Modelling and Analysing Cooperative Adaptive Queueing Networks and their Learning Behaviour

A thesis submitted to the University of Dublin, Trinity College
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

School of Computer Science and Statistics
University of Dublin, Trinity College

May 2010

Dominik Roman Christian Dahlem

Declaration

I, the undersigned, declare that this work has not previously been submitted to this or any other University, and that unless otherwise stated, it is entirely my own work.
I agree that Trinity College Library may lend or copy this thesis upon request.

Dominik Roman Christian Dahlem

Dated: 8th June 2010

To my Dad

Summary

Service networks and the assignment of tasks on such networks are prevalent abstractions in a spectrum of domains spanning manufacturing systems, where a strict dependency on the execution of the tasks exist, load-balancing or routing, where packets are forwarded along links to balance the load and reach an ultimate destination respectively, and e-Commerce applications, where (autonomous) services interact to fulfill a task request. As those networks increase in scale and cross organisational boundaries, centralised control mechanisms pose limitations to scalability. Additionally, these systems exhibit real-time constraints in their queueing behaviour that limit the scope of offline planning and optimisation solutions. Instead, decentralised online control mechanisms are employed that find feasible solutions that may be globally suboptimal, but represent good near-optimal solutions. Decentralised control approaches, however, introduce non-stationary behaviours driven by the dynamic interaction of the autonomous decision-making agents. This requires carefully crafting control mechanisms to avoid selfish behaviour of the agents with a detrimental impact on the other agents such that the whole system suffers and experiences the "Tragedy of the Commons". This thesis is fundamentally an empirical study, and as such the modelling of task networks relies on the mathematical framework of Queueing theory. Queueing theory provides a rich semantics to analyse the performance of interconnected queues based on basic principles of task requests arriving and being serviced according to specific probability distributions. Simulation studies play an ever increasing role in analysing and understanding communication or traffic systems. This is especially the case for complex adaptive systems governed by decentralised control mechanisms, where closed-form solutions do not exist, to address techniques such as online learning for optimising routing behaviour. Importantly, the underlying network structure or topology is treated as a primary evaluation aspect, which is often neglected.

The main goals of the research underlying this thesis are 1) to model large service networks exhibiting specific topological features, 2) to translate the resulting graph-theoretic structure thus derived into a cooperative decentralised optimisation framework, based on multi-agent reinforcement learning, and 3) to find the best multi-agent reinforcement learning settings for a given scenario and analyse the temporal variation of the adaptive forces driven by the individual optimisation tasks over time. To this end, this thesis can be stated as follows: By simulating the decentralised optimisation algorithm with a large variety of network structures, more refined results on the behaviour and performance of the individual agents and the system as a whole can be obtained. In achieving the goals of the thesis thus defined, complex network modelling and analysis is combined with machine learning. The perspective of the behavioural analysis of the adaptive interactions on queueing networks over time enrich the equilibrium analysis usually conducted, and consequently provides a more comprehensive view on multi-agent systems.

This thesis demonstrates that sophisticated statistical techniques based around response surface methodology can be used to guide efficient simulations of network queuing models to explore the relationships between several explanatory variables of the respective learning method and queueing performance measures. The analytical tools of response surface methodology aid in finding globally optimal learning parameters and elucidate the sensitivities of the learning parameters to the respective queueing performance measure. One challenge in reinforcement learning is balancing exploratory and exploitative actions in an uncertain environment, which is exacerbated in multi-agent reinforcement learning. Decentralised control is implemented using SARSA(0) reinforcement learning with neural network function approximators and policies that suitably allow adaptations in a continuously changing environment to study the impact of the underlying task topologies on the learning behaviour. Additionally, the effect of collaborative function approximation is examined. The simulation platform is implemented to utilise high-performance computing infrastructures.

Acknowledgements

As my Ph.D. work draws to a close, inevitably it leads me to a time of reflection on this past experience. One way to express it is M.C. Escher's print "metamorphosis III" which evokes certain parallels to my experience here at Trinity College. This print starts on a plane grey surface... as you move along the print, paths are emerging – leading into what seems a maze, transforming into simple patterns that morph into more intricate patterns only to return to simplicity until finally you reach the – start! In its symbolism, I feel I am part of this print and the process of conducting research originating from a simple playing field leading to ever more complex, sometimes confusing concepts. Research strands continuously blend into each other. Occasionally, without realising, contributing to the process or not. But eventually, you will reach the end and submit when it slowly dawns on you that you are at the beginning again. Only in retrospect would you recognise the beauty of the path, while in the midst of it, it is hard work – but fun.

Without a formidable wife, a lovely family, joyful friends, and helpful colleagues, the gravel on this path would have been rocks, laughter a grunt, and stimulating discussions mere mutterings.

I would like to express my gratitude to everyone who, directly or indirectly, had an influence in helping this thesis come to be. Without some people, this thesis would not have been possible. First and foremost, I would sincerely like to thank both my advisors Jim Dowling, whose enthusiasm and inspiration got me interested into reinforcement learning and inter-disciplinary research, and Bill Harrison, for his great attention to detail and who encouraged me to pursue educational activities in high-performance computing and statistics that at first seemed to take away precious research time, but paid off in the long run. My gratitude extends to Mike Peardon, Geoff Bradley, Richard Timoney, Stefan Sint, Dermot Frost, and David Rohan of the Mathematics Department and the Trinity Centre for High-Performance Computing for welcoming me in their post-graduate courses. Without the analytical skills I learned there, my approach to evaluating this work would have been entirely different. And John Haslett for valuable feedback on Kriging metamodelling. Further afield, I also would like to thank Reinhold Kröger, who gave me the opportunity to move to Dublin.

I acknowledge the Trinity Centre for High Performance Computing and the IITAC program for access to the TCHPC computing facilities. In particular Darach Golden, Jimmy Tang, Paddy Doyle, and Neil Brookes were very helpful in getting me set up and responded swiftly to all parallel computing questions I had.

"Life's greatest happiness is to be convinced we are loved (Victor Hugo, Les Miserables, 1862)." No doubt, my wife Elizabeth showed compassion, love, and patience with me all along and I sincerely appreciate the amount of time she devoted to listening to my research and the valuable comments that contributed to making sense of my thesis. This part of our journey is over now – What's next?

And finally, I would like to thank family and friends, who provide the social net: music, cakes, drinks, football, tennis, and everything that makes life worthwhile. One inspirational person deserves special place in these acknowledgements – my dad. When I was more interested in football than school he made sure that I did my homework and rewarded me with techie toys for as long as I can remember. I belief it is the ATARI computers I had since the mid-80s and the Fischertechnik Computing toolkit I got as a combined Christmas and birthday present at the age of 9 that kept me glued to programming and computer science ever since.

Some more characters complete this journey: Anthony, Tim, Andronikos, Ray, Eamonn, Andy, Dave, Jan, David, Pete, As'ad, Ivana, Gregor, Philip, Moe, Miki, Meike, Felipe, Emma, Marta, and everyone I forgot.

The entire thesis and accompanying software has been produced and developed using freely available software. The text was edited with GNU Emacs and processed with $\LaTeX\,2_\varepsilon$ and the $\mathcal{AMS}$-$\LaTeX$collection among others. The surface plots were generated using gnuplot and the statistical evaluation was conducted using GNU R and the packages igraph, ggplot2, snow, tnet, and plfit by Laurent Dubroca. The response surface methodology was implemented using GNU Octave using the optim package and Monte Carlo errors with less errors by Ulli Wolff. The high-performance simulator was developed using Open MPI, Boost Graph Library, Boost Intrusive containers, and other Boost libraries, GNU Scientific Library, the GNU Compiler Collection, and the GNU Autotools. The quality of the software was assured with CppUnit, valgrind, and GNU Gcov. Without these significant contributions of brilliant developers, my life as a researcher and developer would have been much harder.

Dominik Roman Christian Dahlem
University of Dublin, Trinity College
May 2010

Contents

List of Figures

List of Tables

List of Algorithms

Nomenclature

Capitalisation

a, A	Normal-typed letters represent scalars

Matrices and Vectors

$(\cdot)^T$	Transpose of a matrix
$\mathbf{A} \otimes \mathbf{B}$	Kronecker product of two matrices
$\mathbf{a} \succeq b$	All of the components of $\mathbf{a}$ are greater than or equal to b
$\mathbf{A} \times \mathbf{B}$	Element-wise product of two matrices (Hadamard product)
$\mathbf{A}$	Bold-faced upper-case latin or greek letters represent matrices
$\mathbf{a}$	Bold-faced lower-case latin or greek letters represent vectors
$\mathbf{A}^{(j)}$	Column j of matrix $\mathbf{A}$
$\mathbf{A}_{(i)}$	Row i of matrix $\mathbf{A}$
$\mathbf{A}_{ij}$	Row i and column j of matrix $\mathbf{A}$
$\mathit{Diag}\left[\cdot\right]$	Diagonal matrix
$\|\cdot\|_1$	ℓ_1-norm of a vector, i.e., $\|\mathbf{a}\|_1 = \sum_{i=1}^{n} \lvert \mathbf{a}_i \rvert$
$\|\cdot\|_2$	ℓ_2-norm (Euclidean) of a vector, i.e., $\|\mathbf{a}\|_2 = \sqrt{\mathbf{a}^T \mathbf{a}}$
$\|\cdot\|_\infty$	ℓ_∞-norm of a vector, i.e., $\|\mathbf{a}\|_\infty = \max_i \lvert \mathbf{a}_i \rvert$
$\vec{a}$	Alternative representation of a vector (same as $\mathbf{a}$)

Graph Theory

$\mathbf{A}$	The adjacency matrix of a graph
$\mathbf{s}^{+/-}(i)$	The (out/in)-strength of a node i
$\mathbf{W}$	The weighted adjacency matrix of a graph
$\mathcal{A}$	The set of arcs in a given graph

$\mathcal{V}$	The set of vertices in a given graph
$\upsilon(i)$	First-order (immediate) neighbours of node i
$\mathtt{deg}^{+/-}(i)$	The (out/in)-degree of a node i

Machine Learning

$\langle \cdots \rangle$	A tuple of n elements
$b \in \mathcal{A} \setminus a$	$b \in \mathcal{A}$ without considering element a
AI	Artificial Intelligence
COIN	Collective Intelligence
DAI	Distributed Artificial Intelligence
DEC-MDP	Decentralised Markov Decision Process
DP	Dynamic Programming
FAL	Fair Action Learner
MARL	Multi-agent Reinforcement Learning
MAS	Multi-agent System
MDP	Markov Decision Process
P2P	Peer-to-Peer
RL	Reinforcement Learning
WPL	Weighted Policy Learner

Probability

$\mathbb{E}[X]$	Expected value of a random variable X
$GP_n(\cdot,\cdot)$	Multi-variate Gaussian Process of dimension n
$G(\cdot,\cdot)$	Gamma distribution
$N_n(\cdot,\cdot)$	Multi-variate Normal distribution of dimension n
$P(A)$	Probability of event A occurring
$P(A \mid B)$	Conditional probability of event A occurring, given that event B has occurred
MLE	Maximum likelihood estimation

Statistics

$\bar{(\cdot)}$	Arithmetic mean of a vector
$\hat{(\cdot)}$	Estimator of a variable
$\tilde{(\cdot)}$	Residuals of a regression model
$\hat{y}_{-i}(\mathbf{x}_i)$	Estimator with the observation $y_i(\mathbf{x}_i)$ removed
PRESS	The predictive error sum of squares
R^2	R^2 statistic (coefficient of multiple determination)
IMSE	Integrated mean-squared error statistic
LHS	Latin Hypercube Sampling
MSE	Mean-squared error statistic
OLH	Orthogonal Latin Hypercube
RMSE	Root mean-squared error statistic

— Suppose that you want to teach the "cat" concept to a very young child. Do you explain that a cat is a relatively small, primarily carnivorous mammal with retractible claws, a distinctive sonic output, etc.? I'll bet not. You probably show the kid a lot of different cats, saying "kitty": each time, until it gets the idea. To put it more generally, generalizations are best made by abstraction from experience.

Ralph Philip Boas, (Can we make mathematics intelligible?, American Mathematical Monthly)

1 Introduction

The inspiration for this work originates from my early research into modelling business interactions in peer-to-peer environments [43, 44, 129, 130]. In the "software as a service" paradigm, business services are advertised via Web Services in an open business ecosystem facilitating the combination of a set of Web Services into a more complex business logic. As businesses join in and grow within such networks, some remain niche services, while others become central to the business network. Modelling and analysing those service networks raises three major challenges. First, the topological features of service networks exhibit specific characteristics that play an important role in the way interactions between entities on this network unfold. Second, composed services depend on the availability and correct behaviour of the individual Web Services they use, and they may lack the flexibility to replace a failed Web service with a redundant alternative. Third, as services are generally autonomous in nature, their independent adaptation in the network needs to be analysed in the context of the performance of the whole service network.

The main goals of the research underlying this thesis are 1) to model large service networks exhibiting specific topological features, 2) to translate the resulting graph-theoretic structure thus derived into a cooperative decentralised optimisation framework, based on multi-agent reinforcement learning, and 3) to find the best multi-agent reinforcement learning settings for a given scenario and analyse the temporal variation of the adaptive forces driven by the individual optimisation tasks over time. To this end, this thesis can be stated as follows: **By simulating the decentralised optimisation algorithm with a large variety of network structures, more refined results on the behaviour and performance of the individual agents and the system as a whole can be obtained.** In achieving the goals of the thesis thus defined, complex network modelling and analysis is combined with machine learning. This thesis also demonstrates how sophisticated statistical techniques can be used to guide efficient simulations of network queuing models to measure and illuminate the effects of structure on and the best parametrisation for learning behaviour in multi-agent networks.

1.1 Task Networks

Business ecosystems mentioned in the first paragraph of this chapter motivate the research efforts in this thesis. However, for the purpose of analysis, task networks are generalised as queueing systems which provide a rich mathematical framework to conduct the empirical investigation. To emphasise on the broader applicability of the abstract framework, the application domain is framed as a sequential distributed task assignment problem. Sequential, because tasks arrive sequentially at nodes in the network. Distributed, because each node in the network is considered autonomous, meaning that task requests arrive from the outside of the network to all of the nodes. So each node is concerned with providing the best possible service to its clients. To achieve this, each node employs a learning module to optimise queueing metrics, such as utilisation, delay, number of events in the queues or response time.

A server is responsible to complete a specific local task given an assigned stochastic service time. A task is composed of many sub-tasks each contributing in some way to the overall task. Two different kinds of service requests are distinguished. When an external service request to the task network arrives at a given server, then this server is the starting point of the overall task. Otherwise, an internal request implies that the server is part of an overall task.

For example, assuming a workflow system, S3 in Figure 1.1 receives an external service request from a client. A portion of the task is completed on S3 before the request is forwarded to either S4 or S5. S4 and S5 provide the same service and therefore are competing with each other, but may have different service times to complete the request. They also show different structural embeddings in the task hierarchy, which means that the internal modularity of the respective task is different. While S4 can provide a service independent of others, S5 requires S6 to deliver parts of the service. Simply put, this structural difference is a contributing factor to the overall performance of a task.

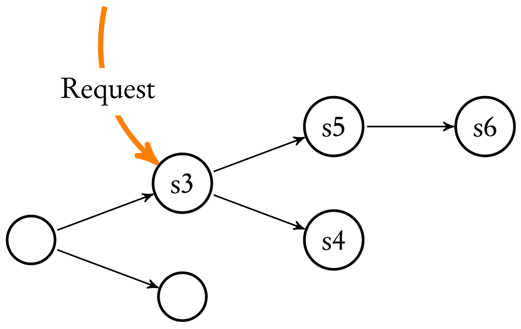


Figure 1.1: Distributed Task Assignment

In the context of this thesis, a few simplifying assumptions are made. These include single class task requests. That means that no differentiation of the actual task request is being made. For example, certain tasks are not prioritised over other ones. Also, tasks arrive stochastically with homogeneous Poisson arrival rates. Real-world queueing networks would account for high-peak and low-peak arrivals that depend on the time of day for example.

1.1.1 A Perspective on Coupling Artificial Intelligence and Complex Networks

Although not the only one, an interesting avenue being used in tackling the decentralised and autonomous control challenges is multi-agent reinforcement learning, which belongs to the broader domain of distributed artificial intelligence techniques. Artificial intelligence draws its inspiration from biological systems that evolved over many millions of years to solve problems through interactions with a dynamic environment. The actors of this environment are systematically transformed as a result of the experiences that unfold within this complex, dynamic, and seemingly chaotic context. Many diverse disciplines such as mathematics, physics, biology, psychology, and computer science try to understand these biological processes that drive the transformations, because of their promise to solve problems relevant in their respective field. The field of psychology in particular has had a great influence on computer science, encouraging to devise algorithms that can enable computers to process data in a way that resembles learning. There are two questions raised in this thesis with respect to modelling and analysing service networks. The first question is how learning modules can be employed such that computers can autonomously reason about their environment to achieve robust service selection as given in the business ecosystem example above? The second is whether learning behaviour is affected by the topological structure of the networks and, further, how it is affected? These two questions focus on and put forth an agenda to coalesce machine learning and complex network modelling and analysis.

Complex network modelling and analysis has its origins in sociology, where the study of relationships among human beings is called social network analysis. Modern social network analysis studies structural features of social phenomena and is generally based on four approaches. Most prominently is the intuition that social actors are linked via ties. The identification of the social actors and their ties is grounded in systematic empirical data. Additionally, it draws heavily on graphical representations of elements of the network and relies on mathematical and computational models. For an historic account on the development of social network analysis from the perspective of sociology see Freeman [56]. However, investigating the interaction among entities is not only confined to human social relationships, but has also been extended to other fields of research. Understanding the structure of social interaction leads to generalising patterns that govern these interactions. Related to those patterns are relatively simple computational models that imply specific structural features. A basis for answering the questions posed above is the establishment of an organisational model of such task or business networks. The formal investigation of networks is a fairly young discipline that was originally shaped by Pál Erdős and Alfréd Rényi who hypothesised that complex networks exhibit random interconnections. However, random graphs could not explain some of the phenomena that were observed by sociologists, such as “six degrees of separation” in the famous Milgram experiment [99]. This lead to more sophisticated network evolution models that incorporate some kind of social structure.

The study of networks is applicable to any domain that is governed one way or another by them. Recent articles in the magazine “Science” put a great emphasis not only on structural features of networks, but also on the interaction patterns of the entities involved [18, 116, 135, 166]. In the following articles, it is argued that the evolution of interaction patterns is as important (if not more so) as the topological features of the networks. Ecological systems are sustained by the interaction among different species and their diverse identities [18]. Some important questions of why certain social-ecological systems collapse and others do not, e.g., through the influence of over-fishing, require the identification and analysis of relationships among the subsystems, including resources, users, and governance [116]. The current economic crisis illustrates the difficulty of predicting or controlling credit and investment networks, trade relationships, and supply chains [135]. Finally, the intersection of technological and human networks whose dynamics and evolution are defined by the interaction with human behaviour [166]. These aspects highlight the importance of a network view on both structure and dynamics and motivate the approach of modelling and analysing large decentralised adaptive queueing systems. In fact, the adaptive behaviour in queueing networks combines both a topological structure and dynamic interaction patterns to provide a compelling basis for applying analysis tools derived from the field of complex networks research.

In pursuit of this research, the theme of cooperative decentralised artificial intelligence techniques coupled with the static and dynamic analysis of complex adaptive networks recurs throughout this thesis. The complex interactions that arise through multiple learning entities and that are organised in some kind of network allow for some fascinating insights. To facilitate a thorough empirical analysis, the application domain of service selection introduced above is generalised using queueing theory. With queueing theory, performance metrics are well defined, both as equilibrium solutions and as part of discrete event simulation. In that sense, the dynamic interaction through learning processes in a service network does not lend itself to analytical solutions, but instead must be simulated.

The task hierarchy of a service network prescribes discrete structures representing pairs of nodes and links connecting them. The dichotomous nature of the static structure in which links between nodes are either present or absent, provides a very basic framework to analyse some of the features of task hierarchies specifically, and graphs more generally. Extensions of this basic framework can be integrated to accommodate more complex scenarios [27]. For example, the links may carry weights, or the nodes may be associated with a real value to give them a richer semantics. In fact queueing theory achieves exactly that. The weights on the links represent routing probabilities and the nodes represent services that exhibit certain performance characteristics. The adaptive forces introduced with artificial intelligence techniques transform the network in a systematic way which expresses itself directly in a change of the weights of the links and indirectly impacts the performance of the nodes in the network.

1.1.2 Empirical Evaluation

One evaluation goal of this thesis is to provide a method for finding the optimal learning parameters of given reinforcement learning methods to be able to compare their utility to modelling cooperative task networks. In a queueing network this may be minimising total delay or mean utilisation, or maximising total throughput. This in itself is a challenging task, because the task of simulating queueing systems is constrained by time and computer resources. Blindly simulating every possible parameter setting may be possible for very small problem domains with few integer or boolean parameters. But real valued parameters or larger simulation studies require more intelligent design of experiments.

A second evaluation goal is the dynamic analysis of learning behaviours with a networks perspective. For that purpose complex network analysis tools can be applied to uncover the learning behaviour that unfolds on complex adaptive networks. This perspective augments the goal of finding optimal results which is mainly concerned with average performance criteria in the limit or once the system has converged. The dynamic analysis of evolving behaviour on complex networks requires an investigation of time slices of the network. This investigation reveals possible fluctuations in learning behaviour that is impossible to attain using equilibrium analysis. While one learning algorithm may produce the optimal long-term queueing characteristics its downside may be huge fluctuations in its impact on the network.

1.2 Why Multi-Agent Reinforcement Learning?

Before delving into the intricacies of the multi-agent part of the question, some introduction of reinforcement learning in general is necessary. Reinforcement learning is a term used to describe learning that takes place by observing cause and effect of actions without an explicit teacher. Moreover, the actions are aligned with single or multiple goals. The central idea is that reappraisal of learning and memory are incorporated into a trial-and-error interaction within a certain context. The acquisition of knowledge is guided by a scalar reinforcement signal that determines the magnitude of the "goodness" of an action. Positive reinforcement signals select and strengthen new behaviour, while aversive stimuli weaken poor behaviour. This is a remarkably simple framework that is able to learn skills from recurrent choice in similar contexts [57]. Therefore, it is attractive to apply reinforcement learning to domains where prior knowledge of the environment is difficult to obtain. However, one might wonder whether reinforcement learning (or more generally any learning method) is the best one in any given environment. Wolpert and Macready [178] proved that there is no learning algorithm that outperforms all others averaged over all possible learning problems, which is famously known as the "no free lunch theorem".

More specifically, this thesis addresses the problem of engineering and evaluating complex adaptive systems using reinforcement learning techniques in order to adapt to a dynamic environment for sequential distributed task assignment problems. Reinforcement learning is used to optimise the selection of alternate paths based on past interactions and future prospects without explicit global knowledge about the system. Each learning module co-located with a node in the task network monitors queueing performance parameters of the interactions and adapts its transition probabilities accordingly to reflect the relative difference in the performance of the available nodes. The inherent stochastic nature of dynamic environments coupled with a learning process allows for autonomous adaptations in order to support optimal decisions during the runtime of the system. The application domain this thesis deals with suggests that multiple reinforcement learners are employed to achieve a common goal, where autonomous control units are generally called agents.

Broadly speaking, this line of research falls into the field of distributed artificial intelligence. But as distributed artificial intelligence maintains its more traditional roots to artificial intelligence as reasoning, understanding, and learning, multi-agent systems emerged as a new concept to emphasise interactions among agents and the learning algorithms they are equipped with [156].

Allowing multiple learners to co-exist in an environment introduces highly dynamic adaptive forces, where system structures can appear without explicit external pressure (exterior control processes), but instead take shape during internal interactions of the components involved within the system. These systems are called self-organising systems. Often those systems show transient phenomena or emergent behaviours that were not designed explicitly. Revealing the underlying processes that lead to the emergence of certain characteristics or predicting behaviours is therefore of particular interest for system engineers.

Now, turning the title of this section around: "if multi-agent learning is the answer, what is the question?"(posed by Shoham et al. [137]). Mannor and Shamma [93] address this question from the engineering perspective. The engineering agenda should include robustness guarantees of the multi-agent system and it should include domain knowledge efficiently. This mirrors Wolpert and Macready [178] "no free lunch theorem" where any engineered system equipped with learning methods should be customised to fit the respective context of the application domain.

Bonabeau et al. [22] also have something to say about engineering large-scale intelligent systems. Their perspective is inspired by social insects and their tremendous success of optimising one of the key organisational elements of complex societies: division of labour. The premise of swarm intelligence is that rich behaviour of social insect colonies arise solely through their interactions mostly governed by simple rules. Retrofitting these ideas to distributed systems has been extremely successful in areas such as routing and load-balancing because the inherent bottom-up approach is amenable to system engineers designing solutions to their specific problems.

To conclude this section, multi-agent reinforcement learning holds a promise to efficiently solve decentralised problem settings. This has to be understood with a caveat though. Applying any learning techniques to decentralised problems is by no means easy nor is there a single best solution. Later chapters will go into detail of how exactly sequential distributed task assignment problems can be modelled. Repeating the question "if multi-agent learning is the answer, what is the question?" leads to the next section, where the networks perspective of multi-agent learning is imposed in order to gain insights into the dynamic adaptation processes occurring in complex networks.

1.3 Why Complex Network Modelling?

The answer to why complex network modelling and analysis is so important can be given in two parts. On a general note, Mitchell [100] points out the importance of "network thinking" to the general field of artificial intelligence. Not only, because it may be an integral part for developing effective decentralised algorithms, but also to assist humans in their network-related tasks. Good examples of synergies between AI and network science are ant-colony optimisation [47] or more generally swarm intelligence [22]. Yet there are two main directions that are relatively unexplored at the intersection of "network thinking" and distributed artificial intelligence.

Firstly, many applications of machine learning approaches in the multi-agent setting, especially with a leaning towards game-theoretic analysis, cover very restrictive classes of problems: either the payoff matrix is given, or stationary stochastic games are assumed, or the number of players is small. To scale multi-agent scenarios up beyond illustrative academic problems one either has to have access to real-world data, which is mostly difficult to obtain, or one needs to develop some form of evolution models that iteratively grow a network. There are a multitude of network growth models, ranging from purely random ones introduced by Erdős and Rényi [51] to regular networks with random rewiring to achieve small-world properties by Watts and Strogatz [171] to preferential attachment models that explain the emergence of scaling by Barabási and Albert [15]. These are seminal articles that attempt to elucidate topological features of real-world systems through simplified network evolution models. However, establishing synergies between network evolution models and queueing networks in particular are still elusive.

Secondly, the analytical tools for investigating certain characteristics of networks, in particular the ones that account for weighted (directed or undirected) networks have received a lot of attention in recent years [5, 6, 98, 109, 114, 133, 168]. Additionally, time-evolving networks have been investigated to uncover, for example, the influential stakeholders in companies [19, 63]. Not only is it fascinating to obtain results from real-world networks that surround our every day lives, but also those techniques are quite useful in artificial intelligence to uncover the dynamics of learning for example. Applying those techniques in multi-agent learning settings is the second goal of this thesis which provides a novel perspective on measuring the quality of concurrent learning processes.

Essentially, these two directions are extensions of "network thinking" for intelligent autonomous multi-agent systems which both provide a different angle on approaching the general problem area. Network evolution models align with the engineering discipline in a sense that large-scale systems need to be set up prior to investigating specially designed learning methods. Analysing weighted networks including evolving networks cover the analytical part of the investigation and may provide a deeper picture of the runtime behaviour of single agents or groups of agents.

1.4 Challenges

Engineering distributed systems in general one encounters numerous challenges that have been present since the first day of their deployment, such as fault-tolerance, performance and scalability, inter-component communication, synchronisation, handling non-deterministic behaviour, etc. An ever more intertwined technological world only emphasises their importance. As Leslie Lamport once famously said:

> A distributed system is one in which the failure of a computer you didn't even know existed can render your own computer unusable.

Cascading failures are very serious, as the case of the power blackout of northeast America in 2003 clearly exemplifies. While a human error as mundane as a failure to cut trees to avoid power-lines to short-circuit, a computer error in an energy management system deprived system operators of audible and visual alerts of crucial changes in the state of the power-grid, which ultimately lead to over 200 power plants being dropped off the grid.

As distributed systems scale up, central management units become harder to maintain, because communication on the one hand and correlating state signals and emanating control instructions on the other hand introduce single-points of failure and bottlenecks. Instead, decentralised control becomes increasingly prevalent, often employing autonomous agents to achieve system goals. In the following paragraphs, a focus is placed on the challenges of these decentralised multi-agent systems.

From an engineering perspective, it is desirable to decompose a global task into sub-tasks, each possibly managed by an agent that can tractably solve the sub-task. The division of labour among a number of agents requires the establishment of communication channels to communicate partial or complete results or to provide sufficient information to each agent in order for it to achieve the task. If agents communicate with each other to contribute to an overall task, coordination mechanisms need to be employed. Ensuring that agents cooperate on the global task turns out to be very difficult, because agents may act selfishly to complete their own task as best as they can. In pursuit of their own goal, agents may loose sight of the agents they (should) cooperate with, which could lead to deteriorating system performances. Within the framework of reinforcement learning, the reward structure needs to reflect the attitude of the agents in the environment. It is by all means possible to design agents with only local reward structures, which is a characteristic of selfish agents, to act in a cooperative manner. Otherwise, global reward structure that apportion credit fairly to the respective agents in solving a task provides an inherent cooperative design. However, if the application domain is not conducive to global reward structures, communication is required to determine who did what and how much an agent contributed. The same is true for selfish agents to encourage cooperative behaviour. Otherwise, selfish agents may fall prey to "the Tragedy of the Commons", which states that selfish behaviour may ultimately lead to the destruction of the common shared resources even though this is in nobody's interest [69]. Game theory, a branch of applied mathematics, and most notably used in economic settings is a formal framework to analyse strategic situations where more than one player (in the terminology of game theory) is involved and their outcome depends on each of their actions. Most of the research in multi-agent learning utilise the game-theoretical setting to analyse the interactions between agents.

From the perspective of learning theory, multi-agent reinforcement learning violates the convergence guarantees of single-agent methods. For example, that single agents convergence to optimal learning policies has been proved for temporal difference methods under certain conditions. As the state space for single-agent scenarios increases, compact representations of the state-action mapping need to be implemented. However, with the introduction of function approximators, convergence proofs become brittle. This is especially the case for non-linear function approximators like feedforward neural networks. There are mixed results with using function approximators in reinforcement learning tasks and consequently, the statistical learning community strives to invent methods that overcome some of the short-comings of neural networks, while providing similar generalisation power and provable convergence guarantees at the same time. However, neural networks did show some remarkable success in some applications, such as the TD-gammon game by Tesauro [149].

Lifting reinforcement learning methods into the multi-agent settings complicates the theory even further, because agents co-adapt alongside each other. So, the learning behaviour of any agent follows a moving target, because other agents in the environment are learning at the same time. Consequently, the environment is not stationary. This means that actions cannot be taken deterministically in any given context. Taking actions greedily embodies the agent's knowledge that this is no doubt the best way to proceed, even though other agents are still learning and thereby introducing changes into the environment. Greedy actions neglect that adaptive forces are still underway in the environment. So, it would be better for the agent to be open-minded to the activity in the environment, which means that actions need to maintain a certain probability of being selected. To illustrate this point, consider a routing network. If an agent believes that one path is the best and consequently forwards all the packets along this way, it might introduce congestion that could lead to cascading failures as in the power-grid example given above. This can be avoided by constantly monitoring the environment and acting accordingly.

Since multi-agent systems do not conform to the formal single-agent convergence guarantees, simulation studies are necessary to investigate their behaviour, both at runtime and the steady-state average performance. However, the specification of simulation studies often reflect the analyst's own bias by choosing amenable input parameters or restricting the set of input parameters to a representative few to illustrate the respective research hypothesis. To overcome this bias standard statistical techniques should be employed to obtain results within a given confidence interval and response surface methodology should be used to allow comparative studies. For instance, if two learning methods are compared and both show completely different characteristics, introducing a bias to simulate a representative set of the learning parameters may potentially miss a region in the design space that optimises the respective behaviour of the simulation. The consequence is that both methods cannot be empirically compared. Moreover, simulation studies may become prohibitively expensive to run unless only small scenarios are tried. This makes it even more important to apply variance reduction techniques and an intelligent design of experiments to achieve a parsimonious and congruent model.

The next section provides an outline of the thesis and points to chapters that address these challenges.

1.5 Outline and Contributions

Chapter 2 is a self-contained chapter describing the details of the evaluation methodology, which is drawn from published literature. Simulation studies play an ever increasing role in analysing and understanding communication or traffic systems. This chapter details how to specify sequential designs of experiments to construct a model of the simulation response given the input data. This allows iterative improvements of an interpolation function to describe the surface and also facilitates a thorough analysis to find the input variables that contribute most to the dynamic behaviour of the output. Essentially, this is the enabling technology to achieve the first goal of this thesis, i.e., finding the optimum settings for a given learning algorithm to solve large-scale sequential distributed task assignment problems.

Chapter 3 describes network growth models adapted for queueing systems. Often, the underlying network structure or topology of simulation studies is treated as a secondary evaluation aspect and often neglected. This chapter stresses the importance of evolving networks with certain topological characteristics. To that end, it introduces two entirely different network growth models, one that leads to a random structure and one that exhibits a more heterogeneous structure akin to social networks. The utilisation of the queues and the waiting times of events in the queues are investigated and their respective sensitivities with respect to the network topologies are studied. It is analytically shown that the two parameters that determine how capacity is boosted on vertices and along the arcs respectively impact the queueing performance of the two different network topologies.

Chapter 4 provides fundamental details of reinforcement learning. It reviews the main learning algorithms relevant for this thesis and explains the limitations of applying those to multi-agent reinforcement learning. This chapter then continues to describe recent research efforts in multi-agent learning, especially covering the basics of game-theory and its influences to this domain.

Chapter 5 models sequential distributed task assignment problems with cooperative multi-agent reinforcement and complex network techniques. It provides the relevant definitions for decentralised Markov decision processes and explains how cooperative behaviour in this setting comes about. The focus is on autonomous multi-agent systems that discover solutions to complex and dynamic tasks online, using learning techniques with roots in dynamic programming and temporal difference reinforcement learning.

Chapter 6 evaluates the sequential distributed task assignment problems with cooperative multi-agent reinforcement using two different policies over two different network structures with different queueing stress levels. The learning methods are calibrated to find their globally optimal learning parameters and their influence on the total event processing time is investigated, thereby achieving the first goal of this thesis.

Chapter 7 investigates the second goal of this thesis of uncovering the non-linear dynamics of learning behaviour on social and random distributed task assignment topologies with different learning policies employed.

Chapter 8 concludes this thesis and gives some areas of future work. It addresses some of the assumptions made and proposes future research directions to expand on the methods used and results obtained.

Table 1.1 depicts the machine learning techniques used in this thesis.

Table 1.1: Machine Learning Techniques used in this Thesis

Supervised Learning	Active Learning	Reinforcement Learning
Function Approximation	Response Surface Improvement	SARSA(0) temporal-difference method

To anticipate some of the results obtained in this thesis, the employed machine learning techniques have proved very successful. It is captivating to witness how a few algorithms actually do learn in uncertain environments and bring about robustness characteristics. Supervised learning techniques map inputs to desired outputs. In this thesis feedforward neural networks are used to establish a mapping between the state space and the value that encodes the "goodness" of an action. Function approximators equip SARSA(0) reinforcement learning agents with the ability to compactly represent learned knowledge. Lastly, active learning plays a role in the statistical framework of improving a response surface of the simulation output by selecting sample points that are likely to improve the response surface.

The contributions of this thesis can be summarised as follows:

- An agent-based simulation platform implementing parallel algorithms for high-performance computing clusters.

- The adaptation of the Erdős-Rényi (ER) random graph and the Barrat, Barthèlemy, and

Vespignani model (BBV) social graph to model specifically queueing networks is a novel application to queueing networks (Chapter 3).

- The reverse effect of the weighted versus the unweighted assortativity metric for both network evolution models is a new insightful result, which emphasises the importance of characterising the underlying graph structure (Chapter 3).
- Incorporating network evolution models into agent-based simulation for queueing networks to analyse learning algorithms is new (Chapter 5).
- Modelling large-scale sequential distributed task assignment problems with SARSA(0) reinforcement learning and a neural network function approximator to generalise over the state-action space is new (Chapter 5).
- To improve scalability of cooperative multi-agent systems, a contribution is the outsourcing of the function approximators which also facilitates faster learning (Chapter 5).
- Employing response surface methodology to find the best parametrisation for reinforcement learning methods and analysing the sensitivities of the learning parameters has to my knowledge not been done before (Chapter 6).
- The metric to measure the distance of learning in Section 7.1 is new.
- The application and results of dynamic network analysis using both queueing metrics and metrics inspired from shareholder networks is new (Chapter 7).

To conclude, the premise of this thesis is to gain a deeper understanding of cooperative decentralised optimisation problems for sequential distributed task assignment. The main building blocks to achieve this goal are the statistical and mathematical framework of response surface methodology, the modelling of large queueing networks with certain topological features, and the behavioural analysis of the individual entities within the context of their relative structural embedding within their network.

— But when a rule is extremely complex, that which conforms to it passes for random.

Gottfried Wilhelm Leibniz, (Discours de métaphysique)

2 Response Surface Methodology for Stochastic Computer Simulations

Computer simulation studies play an ever increasing role, especially when closed-form analytical solutions do not exist. To this end, this chapter presents background material to thoroughly analyse the outputs with respect to the inputs of stochastic computer simulation. This means, that some interpolation approach is needed to estimate the output of computer simulations not yet run. Based on this interpolation function, the analyst is often interested in finding the best settings for a number of design parameters that influence the output of the computer simulation. The metamodelling process depicted in Figure 2.1 shows the four major steps from modelling a real physical system as a computer simulation to attaining an optimal design and finally deploying a real-world system based on the results obtained. Along this process several sources of errors may occur and need to be integrated into an optimal design [142].

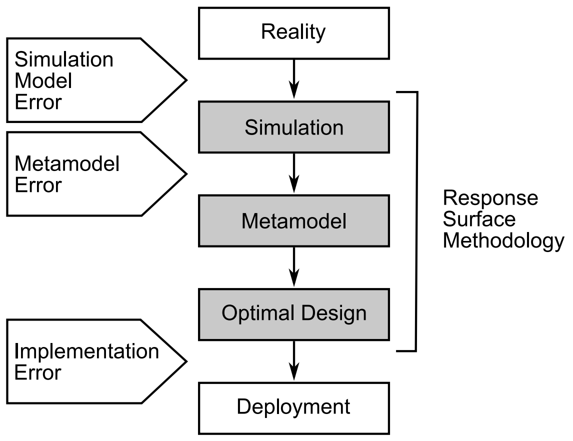


Figure 2.1: The Metamodelling Process

This thesis is concerned with modelling distributed task assignment problems using a simulation-based approach. As such, the focus is not on an accurate representation of a specific real-world system and therefore deployment is not considered. However, great emphasis is placed on an integrated statistical and mathematical methodology, which covers the construction of a metamodel using Kriging coupled with a design of experiments called Latin Hypercube Sampling, and a mathematical analysis of the resulting metamodel.

The analysis of a metamodel may indicate that further improvements of the metamodel are desirable, which can then be incorporated into a sequential design process. Sequential designs are iterative algorithms that use the input/output data acquired from previous iterations to guide the decision for future sample selection. This way a global metamodel can be created that mimics the underlying simulation function, but allows for a much faster computation. A local metamodel on the other hand aims at finding optima, without obtaining a complete picture of the underlying simulation function.

An attempt is made to present the relevant material in a self-contained way. As a consequence most of the mathematical derivations are taken from the respective literature.

2.1 Introduction to Response Surface Methodology

Models need to be established in order to understand and develop appropriate relationships between variables and to predict variables at unknown locations where data have not been collected [163]. The investigation of distributed task assignment problems and their dynamic behaviour driven by different learning algorithms requires a careful analysis of the respective design variables with respect to the simulation output. The ultimate goal is to provide a statistical methodology that is capable of attaining a high-fidelity model in a timely manner and draw meaningful conclusions on the performance of the underlying computer simulation.

A model is a computationally efficient approximation of the input/output function of the underlying computer simulation. Computer simulations show varying degrees of complexity from relatively simple linear regression models to more complicated models that incorporate non-linear relationships between variables. In deterministic scientific phenomena a mechanistic view may introduce an appropriate error term to capture the idea that something is not known exactly. However, computer simulations are often employed when closed-form mathematical expressions do not exist and therefore statistical approaches are necessary to establish a high-fidelity empirical model. Typically, empirical models consist of a first-order or second-order polynomial and an error term. The empirical model is called a response surface model or metamodel, terms which are used throughout this thesis [107].

Section 2.2 gives an overview of spatial data modelling and of establishing a Kriging model in particular. Kriging is a technique for fitting a response surface model that accounts for spatial correlation among the locations. It is a parametric modelling framework that infers model parameters based on the collected data. The inference is computed using Monte Carlo Markov Chain sampling [121]. Kriging modelling consolidates the scientific process of furthering knowledge about a system under investigation. This is facilitated through a sequential process of designing the experiment, constructing and subsequently inspecting and interpreting the response surface. Initial conditions can then be put into context by conducting an analysis of the model and determine for example regions of high uncertainty or regions where solutions do not converge. This can then be taken into account to devise new conjectures and design of experiments leading eventually to a model that can be properly interpreted [24].

Fitting a response surface model to experimental data requires design of experiments to collect the data. A brute-force way of doing this is to create a multi-dimensional mesh to cover the input domain. This is a costly undertaking, because single simulations can take hours or days to complete. In contrast, creating a sparse design may result in insufficient information in certain regions of the design space. Section 2.3 presents a well-known optimal Latin Hypercube Sampling (LHS) technique that reduces the computational cost of collecting data while catering for minimum variance and no bias in estimating the statistical response surface model [97, 185].

The design of experiments and the sequential nature of Kriging can be exploited using high-performance computing approaches to execute the experiments in a parallel fashion. A methodology based on high-performance computing clusters is introduced in Section 2.4. The parallel computing model starts with a pilot design such as Latin Hypercube Sampling [97, 185] to efficiently sample the input domain and then sequentially improves the Kriging model of a response until a desired quality criterion, such as the coefficient of multiple determination, is reached.

Having a model that satisfies some quality criteria assists in further analyses to evaluate the sensitivity of design parameters to the respective response or to find minima, maxima, and stationary points. In other words, if one variable has a steeper slope towards the optimum than other variables, then it is of great interest to identify that variable, because small adjustments to its value will lead to more significant changes in the output of the computer simulation. Canonical or ridge analysis is an approach to find response functions near those points of interest and allows a deeper study of how the independent variables behave in terms of the response [24]. The methodology of canonical and ridge analyses are briefly presented in Section 2.5.

2.2 Overview of Spatial Data Modelling

Increasingly researchers in as diverse fields as climatology, ecology, economics, and computer science are faced with the analysis of data that exhibit multivariate responses with many explanatory variables. For example the application domain presented in this thesis deals with learning algorithms that can be tuned in many ways and their impact on performance characteristics of the queues in the system. As such a research hypothesis might be that one learning method performs better than another. However, it is not obvious how to find the optimal setting, because those parameters could influence each other in some non-linear way. Common explanatory variables include the learning rate which determines how fast a learning entity picks up information; the factor that discounts newly arriving information; a probability of making greedy decisions, etc. Therefore, commonly the task of a researcher is to establish a parametric empirical model of the simulation experiment under study. Statistical inference is the process that estimates the model parameters given the data, i.e., input/output pairs. The advantages of having a model is that predictions of unobserved locations can be done in favour of a computationally expensive simulation.

In spatial data modelling, and in fact in many computer experiments, point-referenced models can be used to classify spatial data sets. These models are characterised as a random vector, $y(x)$, at a d-dimensional location, $x \in \mathbb{R}^d$, where x varies continuously over a design domain $\Omega \subseteq \mathbb{R}^d$. One key aspect of point-level models is the prescription of a covariance structure over the random variables at all locations. More specifically, it is assumed that the covariance between the random variables at two locations depends on the euclidean distance (or some other measure of distance) between the locations. Intuitively, this makes sense in computer experiments, where the random variables in the vicinity of two or more locations show a higher correlation than those that are far apart. This is in contrast to linear regression models that posit a linear relationship among the independent variables and assume, conditional on these variables, that the outcome (responses) are independent. One frequently employed parametric model is the Gaussian autocovariance function, $Cov(y(x_i), y(x_j)) \equiv C(d_{ij}) = \sigma^2 e^{-\theta d_{ij}^2}$ for $i \neq j$, where d_{ij} is the distance between the locations x_i and x_j [14]. Both, σ^2 and θ denote positive parameters known as the partial sill and the decay parameter. Figure 2.2 presents a variogram which measures the average squared difference between each observation and presents this as a function of the distance. The variogram generally assists in Kriging model estimation. It represents the spatial structure of the data under consideration. The structural aspect represents the spatial correlation as a function of the distance between any pair of data points. As the distance increases the variogram values increase until it reaches a maximum at σ^2 (partial sill). The effective range is the distance where the variogram reaches 95% of its maximum which is attained at $\sqrt{3}\frac{1}{\theta}$ for the Gaussian autocovariance function. If at a distance $d \to 0$ a positive variogram value is attained, then this parameter is known as the nugget effect. Given σ^2 the nugget effect, $\tau^2 > 0$, explains additional variability of the data $C(d_{ij}) = Var(y(x_i)) = \tau^2 + \sigma^2$ [14]. Thus, the nugget parameter and the partial sill represent the stochastic aspect of the variogram. Instead of using graphical means to estimate the model parameters of Kriging, statistical methods, such as maximum likelihood estimation or Monte Carlo estimation are frequently employed. Throughout this thesis, Monte Carlo estimation is used to infer the Kriging parameters.

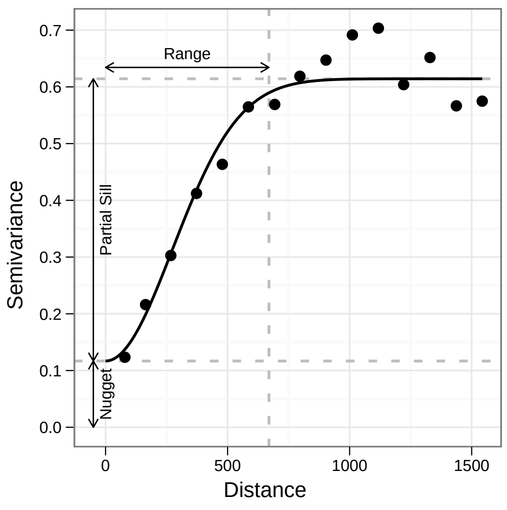


Figure 2.2: Semi-variogram for the Gaussian covariance function

2.2.1 Kriging Modelling

Kriging modelling, first developed by the mining engineer D.G. Krige is an interpolation technique akin to least-squares estimation that relies on the spatial dependence of a response variable [38]. Assuming some prior knowledge of a response variable (i.e., at sampled locations) the Kriging estimator encapsulates its distribution as a function of space. The basic premise is that the Kriging estimator is a linear weighted combination of the known locations. Locations in the vicinity of the predicted value thereby have a higher influence on the estimation than locations that are further remote. Linear models in statistics are important because they are tractable and allow both inference and prediction to be performed.

In contrast to establishing a regular mesh over the input domain, Kriging modelling is a spatial interpolation technique that works best for known locations that are not evenly scattered across the input domain [113].

Traditionally, Kriging is used for rainfall measurements [60, 70, 79] or topsoil concentrations of minerals [106]. More recently, Kriging has also been applied to deterministic [95] and stochastic computer experiments in various fields, such as mechanical engineering [59], aerospace engineering [83], discrete event simulation [159, 160], etc. In the earth and environmental sciences data sets are typically very large in the order of thousands or hundreds of thousands and hence solving the Kriging equations directly involves inverting an $n \times n$ covariance matrix. Matrix inversions are in the order of n^3 which are prohibitively expensive for large data sets and so Kriging becomes the bottleneck of a scientific study. Therefore, parallel implementations of Kriging have been introduced [70, 79]. Because a spatial datum is usually expensive to obtain in computer experiments, sample sizes are generally small and so Kriging is straightforward. Instead of parallelising the Kriging equations, the execution of the stochastic computer experiments is parallelised. The approach presented in Section 2.4 divides Kriging into two phases. Phase one conducts the initial number of experiments over a specified input domain and phase two takes advantage of the iterative nature of Kriging to improve the model.

2.2.1.1 Stochastic Kriging

Kriging modelling is a discipline that originates from the geostatistical discipline and is increasingly used in simulation studies in order to reduce the computer time necessary to achieve enough information about the system under study so that meaningful predictions can be made. The notion of predictions in that sense means the assessment of a random quantity within the design domain that presently is not known exactly. More specifically, simulated input/output data, a finite set, $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, of n scattered d-dimensional points in the domain, $\Omega \subseteq \mathbb{R}^d$, are interpolated, respecting the actually observed output values, $\mathbf{y} = \{y_1, \ldots, y_n\}^T$, in order to be able to predict outputs for unobserved input locations [94].

The mathematical form of a Kriging model is given in equation (2.1) as a multivariate spatial regression model for each location $\mathbf{x}$ comprising an $n \times 1$ response vector, $\hat{\mathbf{y}}$, along with a fixed $n \times p$ matrix of regressors, $\mathbf{F}$.

$$\hat{y}(x) = \sum_{i=1}^{k} \beta_i f_i + Z(x) + \epsilon(x) = \mathbf{F}\boldsymbol{\beta} + \mathbf{Z} + \boldsymbol{\epsilon} \tag{2.1}$$

The first part represents a first-order polynomial linear regression model of the data with k regressors, which models the trend of the hypersurface over the domain. Assuming a constant mean for all independent variables, $\mathbf{F}\boldsymbol{\beta} = \mathbf{1}\mu$, where $\mathbf{1}$ is a vector of ones, is called ordinary Kriging. A more versatile model is achieved by adding additional covariates in $\mathbf{F}\boldsymbol{\beta}$ to the model. These models are called universal Kriging that were developed for spatial interpolation where a drift in the experimental data seems to be justified. The second part defines a $n \times 1$ zero-centered Gaussian random process, denoted $\mathbf{Z} \sim GP_n(0, \mathbf{R}(\cdot, \cdot \mid \boldsymbol{\Theta}))$. This Gaussian process is responsible for the exact interpolation, which means that the mean trend given by $\mathbf{F}\boldsymbol{\beta}$ is pulled through the observed data by accounting for the spatial correlation with mean zero and a Gaussian spatial autocovariance function. The $\boldsymbol{\epsilon}$ denotes an additional error term that traditionally incorporates a positive nugget effect which provides a way of introducing noise into the stochastic process. This way measurement errors can be accounted for, if the mean surface is not exactly smooth and the numerical stability of decomposing the potentially singular covariance matrix can be improved [108]. In this case, the nugget effect is usually a constant added to the diagonals of the matrix $\mathbf{R}(\cdot, \cdot \mid \boldsymbol{\Theta})$ and is independent of the Gaussian process $\mathbf{Z}$. In contrast, stochastic computer simulation outputs have random errors with different variances at the respective design locations. An intuitive extension to traditional Kriging methods is to account for the intrinsic uncertainty of the stochastic simulations as well as the extrinsic one based on the Gaussian process $\mathbf{Z}$. In particular, the intrinsic uncertainty emanating from the stochastic computer simulations can be modelled as a Gaussian process, i.e., $\boldsymbol{e} \sim GP_n(0, \boldsymbol{\Psi}(\cdot, \cdot))$, as well, which is beneficial, if common random numbers are used. In that case, the variances at individual design locations are correlated among themselves and enable more accurate predictions. If independent sampling is assumed, then the matrix is denoted as $\boldsymbol{\Psi} = \mathit{Diag}\left[S^2(x_i)/n_i\right]_{i=1}^{n}$, where $S^2(x_i)$ is the sample variance and n_i is the number of replications of $y(x_i)$ [7, 140].

A product form of the Gaussian autocovariance function with hyperparameters $\boldsymbol{\Theta} = (\sigma^2, \boldsymbol{\theta})$ is given by

$$\mathrm{Cov}(Z(x_i), Z(x_j)) = r(x_i, x_j \mid \boldsymbol{\Theta}) \tag{2.2}$$

$$r(x_i, x_j \mid \boldsymbol{\Theta}) = \sigma^2 c(x_i, x_j \mid \boldsymbol{\Theta}) \tag{2.3}$$

$$c(x_i, x_j \mid \boldsymbol{\Theta}) = \prod_{k=1}^{d} e^{-\theta_k(|x_{i,k} - x_{j,k}|)^2}, \text{ where } \theta_k > 0. \tag{2.4}$$

$$\mathbf{R} = \left[r(x_i, x_j)\right]_{i,j=1}^{n} \tag{2.5}$$

Note, that in equation (2.4) $\boldsymbol{\theta}$ is indexed by k which means that each independent variable correlates with itself in the spatial domain, i.e., this is the reason for the "auto" term. θ_k is associated with each input dimension and represents the length-scale parameter. The length-scale parameter determines by how much y decays given a distance between two locations in the direction of dimension k. Equation (2.3) can be reformulated to emphasise that it only depends on the distance and not on the actual data, i.e., $r(x_i, x_j \mid \Theta) \equiv r(|x_i - x_j| \mid \Theta) \equiv r(\mathbf{d} \mid \Theta)$. Autocovariance functions that solely depend on the distance between locations are called isotropic. In contrast, the hyperparameter σ^2 defines the vertical scale of variation and is assumed stationary for any location x_i. Extensions for deterministic computer experiments do exist that relax the assumption of stationary covariance structures [184]. However, this work is not considered in this thesis.

One important factor to consider is the source of the trend in the data. There are two ways to capture the trend and both are fundamentally different and therefore need careful analysis. One way of absorbing the trend is to include it into the deterministic mean structure of the hypersurface, i.e., by defining regressors for the respective independent variable (first part of equation (2.1)), which is vital to good predictions in ordinary regression. Otherwise, the trend can be incorporated into the spatial autocovariance in the random error variation. A caveat of model building consequently is to include only important independent variables and omit unimportant ones. The goal is to achieve a parsimonious model that explains the relationship of the variables and provides good predictive ability. Assuming autocorrelated errors in the Kriging model random effects of unknown variables are absorbed to give valid, more precise estimation and prediction [163]. It seems counter intuitive to abandon trend modelling in the regressors, but Kriging is able to build good models with very few regressors in the model. In other words, regression aims at filtering out the deviation $\hat{\mathbf{y}}(\mathbf{x}) - \mathbf{F}\boldsymbol{\beta}$ by minimising the sum of squared errors to fit a trend, while Kriging treats these deviations as informative to build a response surface that exactly equals the simulation output. Often a constant trend model is assumed for building computer simulation models. However, it has been shown that higher-dimensional input spaces covered by sparse designs leaves some regions with little information for the Kriging estimation procedure. Incorporating obvious trends in the Kriging model can be beneficial [62] to assume a trend in sparsely covered regions.

Different models can be assessed using the root mean-squared error (RMSE) statistic to investigate the inclusion of regressors into models, which gives an indication of how close the estimates of the model parameters are to the true values and so smaller RMSE indicate a better model.

The best linear unbiased predictor, which minimises the mean-squared error of the prediction is given by

$$\hat{y}(\mathbf{x}) = \mathbf{f}^T(\mathbf{x})\hat{\boldsymbol{\beta}} + \mathbf{r}^T(\mathbf{x})(\mathbf{R} + \boldsymbol{\Psi})^{-1}(\mathbf{y} - \mathbf{F}\hat{\boldsymbol{\beta}}), \tag{2.6}$$

where $\mathbf{r}(\mathbf{x})$ is a column vector of the covariances of the predicted location $\mathbf{x}$ to the existing ones and $\hat{\boldsymbol{\beta}}$ is estimated by its maximum likelihood estimation (MLE) assuming $\boldsymbol{\Theta}$ is known

$$\hat{\boldsymbol{\beta}} = (\mathbf{F}^T(\mathbf{R} + \boldsymbol{\Psi})^{-1}\mathbf{F})^{-1}\mathbf{F}^T(\mathbf{R} + \boldsymbol{\Psi})^{-1}\mathbf{y}. \tag{2.7}$$

Note, that the extrinsic variance, σ^2, cancels out in $\mathbf{r}^T(\mathbf{x})\mathbf{R}^{-1}$. Rearranging equation (2.6) into $\hat{y}(\mathbf{x}) - \mathbf{f}^T(\mathbf{x})\hat{\boldsymbol{\beta}} = \mathbf{r}^T(\mathbf{x})\mathbf{R}^{-1}(\mathbf{y} - \mathbf{F}\hat{\boldsymbol{\beta}})$ clarifies that the predicted residuals at $\mathbf{x}$ are a weighted sum of the residuals observed at the design locations. Additionally, if the Gaussian process is independent at different locations $\mathbf{x}$, i.e., $r(x_i, x_j) = 0$ for $x_i \neq x_j$, then Kriging prediction reduces to the ordinary least square prediction $\hat{y}(\mathbf{x}) = \mathbf{F}\hat{\boldsymbol{\beta}}$.

In traditional Kriging, the best linear unbiased estimator is a frequentist interpretation [141], while Bayesian ones posit a prior belief on the parameters before seeing the data [14, 39]. In that sense, the Kriging prediction, $\hat{y}(\mathbf{x})$, is a posterior to $y(\mathbf{x})$.

In solving equation (2.6) the hyperparameters are assumed to be known a priori. However, this is not the case and therefore needs to be estimated from the set of observations, most commonly done through maximum likelihood estimation. The profile log-likelihood of the model parameters $\boldsymbol{\Theta} = (\sigma^2, \boldsymbol{\theta})$ for stochastic Kriging given is

$$L(\boldsymbol{\Theta} \mid \mathbf{y}) = -\frac{1}{2}\left(n\log(2\pi) + \log\det(\mathbf{R}+\boldsymbol{\Psi}) + (\mathbf{y}-\mathbf{F}\hat{\boldsymbol{\beta}})^T(\mathbf{R}+\boldsymbol{\Psi})^{-1}(\mathbf{y}-\mathbf{F}\hat{\boldsymbol{\beta}})\right), \tag{2.8}$$

where $\hat{\boldsymbol{\beta}}$ is given by equation (2.7). Using Cholesky decomposition to factor matrix $\mathbf{R}+\boldsymbol{\Psi} = \mathbf{L}\mathbf{L}^T$ the profile log-likelihood can be written as

$$\begin{aligned} &\text{maximise } p(\mathbf{y} \mid \hat{\boldsymbol{\beta}}, \sigma^2, \boldsymbol{\theta}) = -\frac{1}{2}n\log(2\pi) - \frac{1}{2}\log\det(\mathbf{L})^2 - \frac{1}{2}\tilde{\mathbf{y}}^T\mathbf{L}^{-1}\tilde{\mathbf{y}} \\ &\text{subject to } \boldsymbol{\theta}, \sigma^2 > 0, \end{aligned} \tag{2.9}$$

where $\tilde{\mathbf{y}} = \mathbf{y} - \mathbf{F}\hat{\boldsymbol{\beta}}$ are the residuals [140].

2.2.1.2 Estimating Kriging Model Parameters

Commonly MLE is used to estimate the Kriging model parameters. For this work Bayesian analysis is used instead, because it allows to visualise the marginal probability distributions and estimate associated uncertainties [94]. The integrals for the Bayesian analysis are computed using an MCMC method. Given a specification of priors on the parameters an adaptive Metropolis-Hastings within Gibbs algorithm [54, 122, 124] shown in Algorithm 2.1 is employed.

In the marginalised model, the response vector and the parameters are distributed as

$$\mathbf{y} \sim \mathcal{N}_n(\mathbf{F}\boldsymbol{\beta}, \mathbf{R}+\boldsymbol{\Psi}) \tag{2.10}$$

$$\boldsymbol{\theta} \sim \mathcal{N}_d(\boldsymbol{\mu}_{\boldsymbol{\theta}}, \boldsymbol{\Sigma}_{\boldsymbol{\theta}}) \tag{2.11}$$

$$\sigma^2 \sim \mathcal{G}(\tau_{\sigma^2}, \mu_{\sigma^2}/\tau_{\sigma^2}), \tag{2.12}$$

where equations (2.11) - (2.12) are centred around their previously accepted respective posterior samples. The draws for logit$(\boldsymbol{\theta})$ are from a multi-variate normal distribution with the variance being adapted to achieve $\sim 23\%$ acceptance rate. σ^2 is log-transformed and the samples are similarly drawn from the Gamma distribution with τ_{σ^2} controlling both the dispersion and the shape of the distribution. Large values for τ_{σ^2} produce priors concentrated around μ_{σ^2}, whereas small values produce vague priors [108]. Following the proposals, the inverse transformations are applied to $\mathtt{logit}(\boldsymbol{\theta})$ and $\log(\sigma^2)$ in order to calculate the Metropolis target density equation (2.9). The logit prior takes into account lower and upper boundary values

$$\text{logit}(\theta_i) = \log(\theta_i - \theta_{i,min}) - \log(\theta_{i,\max} - \theta_i) \tag{2.13}$$

$$\text{logit}(\theta_i)^{-1} = \theta_{i,\max} - \left[(\theta_{i,\max} - \theta_{i,\min})/(1 + e^{\theta_i})\right] \tag{2.14}$$

The posterior distribution of the model parameters given the known observations, y, is denoted as

$$\pi(\hat{\beta}, \sigma^2, \theta \mid y) \propto p(y \mid \hat{\beta}, \sigma^2, \theta)\pi(\sigma^2)\pi(\theta) \tag{2.15}$$

These parameters are estimated using Metropolis-Hastings method of Markov Chain Monte Carlo (MCMC) sampling [121]. MCMC approximates the multi-dimensional posterior distribution in equation (2.15) using the adaptive Metropolis-within-Gibbs Algorithm 2.1.

2.2.1.3 Posterior Predictive Inference

Once an MCMC chain is obtained from $\pi(\mathbf{\Omega} \mid y), \{\mathbf{\Omega}^{(l)} = (\hat{\beta}, \sigma^2, \theta)\}_{l=1}^{L}$ that is sufficiently burnt in and thinned to avoid autocorrelations, predictions, $y^* = [y(s_i)]_{i=1}^{m}$ at locations $\mathbf{X}^*$, can be calculated from either the unmarginalised (2.6) model by sampling the conditional expectations $\mathbb{E}[y^* \mid \text{Data}]^{(l)} = \mathbf{F}^{(l)}\hat{\beta}^{(l)} + \mathbf{Z}^{(l)} + \epsilon$ for $l = 1, \dots, L$, or the marginalised (2.10) likelihood. [1]

Drawing posterior samples from

$$P(y^* \mid \text{Data}) \propto \int \pi(y^* \mid \mathbf{\Omega}, \text{Data})\pi(\mathbf{\Omega} \mid \text{Data})\, d\mathbf{\Omega} \tag{2.16}$$

is straightforward and can be obtained using composition sampling. For each $\mathbf{\Omega}^{(l)} \sim \pi(\mathbf{\Omega} \mid \text{Data})$, draw $y^{*,(l)} \sim \pi(y^* \mid \mathbf{\Omega}^{(l)}, \text{Data})$ to obtain posterior predictive samples $\{y^{*,(l)}\}_{l=1}^{L}$.

So the posterior samples integrate the sources of uncertainties emanating from estimating the model parameters (parametric uncertainty) and using a model to predict unknown locations (structural uncertainty). These uncertainties are intrinsic to the model itself, while the extrinsic uncertainty from the stochastic nature of the computer simulation output can be integrated as well, if the matrix $\mathbf{\Psi}$ is specified. Otherwise, the model reduces to the traditional Kriging approach.

The prediction mean-squared error can be calculated similarly for each element in the MCMC chain.

$$\widehat{\text{MSE}}(\mathbf{x})^{(l)} = \sigma^2 - \sigma^4 \mathbf{r}^T \mathbf{\Sigma}^{-1} \mathbf{r}(\mathbf{x}) + \boldsymbol{\delta}^T \boldsymbol{\delta} \left(\mathbf{F}^T \mathbf{\Sigma}^{-1} \mathbf{F}\right)^{-1}, \quad \text{for each } 1, \dots, L, \tag{2.17}$$

where $\mathbf{\Sigma} = \sigma^2 \mathbf{R} + \mathbf{\Psi}$ and $\boldsymbol{\delta} = \mathbf{f}(\mathbf{x}) - \mathbf{F}^T \mathbf{\Sigma}^{-1} \mathbf{r}(\mathbf{x})\sigma^2$. Equation (2.17) quantifies the structural and parametric uncertainties of the model by applying this function to each value of $\mathbf{\Omega}$ in the burnt-in and sampled Markov chain and taking the mean as the result. It also accounts for the stochastic uncertainty that stems from the computer simulation expressed in the covariance matrix $\mathbf{\Psi}$. This

[1] $\hat{\beta}$ is considered part of the chain, because it is calculated given σ^2, θ and the data

Algorithm 2.1: Metropolis-within-Gibbs

```
Input: $X, y, F, \Psi$, from simulation output
Input: $\theta_0, \sigma_0^2 = n\frac{\sum_{i=1}^{n}(y_i - \bar{y})^2}{n-1}$, initial values for the hyperparameters
Input: $\theta_{min}, \theta_{max}$, lower and upper bounds for $\theta$
Input: $b$, the number of batches
Input: $l$, the batch length
Input: $\mathbf{t}$, the tuning vector
Input: $acc$, target acceptance rate
$\mathbf{b}_{accept} = \mathbf{0}$
$q$ = equation (2.9)
$p = \sum_{i=1}^{d}(\log(\theta - \theta_{min}) + (\theta_{max} - \theta)) + \log(\sigma^2)$
for $j = 1$ to $b$ batches do
    for $i = 1$ to $l$ batch lengths do
        for $\forall\ \boldsymbol{\Theta}_i$ do
            $\boldsymbol{\Theta} = \left\{\text{logit}(\theta), \log(\sigma^2)\right\}$
            $\boldsymbol{\Theta}' = \boldsymbol{\Theta}$
            propose $\boldsymbol{\Theta}_i'$ according to equations (2.11)or (2.12) with $\boldsymbol{\Sigma}_{\theta_i}$ and $\tau_{\sigma^2}$ set to $\mathbf{t}_i$
            respectively
            $q'$ = equation (2.9) with inverse transformations of $\boldsymbol{\Theta}$ accordingly
            $p' = \sum_{i=1}^{d}(\log(\theta' - \theta_{min}) + (\theta_{max} - \theta')) + \sigma^2$
            if rand() $\leqslant ((q' + p') - (q + p))$ then
                $\mathbf{b}_{i,accept}{+}{+}$
                $\boldsymbol{\Theta} = \boldsymbol{\Theta}'$
            end
        end
        Store $\boldsymbol{\Theta}$ in MCMC chain
    end
    for $\forall\ \mathbf{b}_{i,accept}$ do
        if $\mathbf{b}_{i,accept}/l > acc$ then
            increase $\mathbf{t}_i$ for $\theta$
            decrease $\mathbf{t}_i$ for $\sigma^2$ decrease for Gamma prior, otherwise increase $\mathbf{t}_i$
        end
        else
            decrease $\mathbf{t}_i$ for $\theta$
            increase $\mathbf{t}_i$ for $\sigma^2$ increase for Gamma prior, otherwise decrease $\mathbf{t}_i$
        end
    end
end
Output: MCMC chain for $\theta, \sigma^2$
```

approach differs from Kleijnen [81], van Beers and Kleijnen [160], who use a bootstrap sampling of the actual replications to estimate the mean-squared error of Kriging.

2.2.1.4 Kriging Model Assessment

The Kriging model is assessed using two metrics, the root mean-squared error (RMSE) of prediction and the coefficient of multiple determination. The predictive error sum of squares statistic is based on the leave-one-out technique and is given as

$$\mathsf{PRESS} = \sum_{i=1}^{m} (\hat{y}_{-i}(x_i) - y(x_i))^2, \tag{2.18}$$

where $\hat{y}_{-i}(x_i)$ is the estimate of the Kriging model (2.6) with the observation $\hat{y}_i(x_i)$ removed while holding the model parameters constant [102]. Using Currin et al. [40] and Mitchell and Morris [101] re-fitting the Kriging surfaces for each cross validation error is avoided by introducing a computational efficient method.

The estimate of the RMSE using the predictive error sum of squares is then defined as

$$\mathsf{RMSE}_{CV} = \sqrt{\frac{1}{n}\mathsf{PRESS}}. \tag{2.19}$$

Alternatively, the RMSE can be simulated using a number of validation locations and the sampled MCMC chain using equation (2.17). The coefficient of multiple determination, $R^2_{\text{prediction}}$, estimates the amount of variability that the model should explain at a validation set of $m \gg n$ scattered unobserved location

$$R^2_{\text{prediction}} = 1 - \frac{\mathsf{PRESS}}{\mathbf{y}^T\mathbf{y} - \frac{(\sum_{i=1}^{m} y_i)^2}{m}}, \tag{2.20}$$

The denominator is the total sum of squared errors for the m unobserved locations which determines the goodness of fit. As the total sum of squared errors approaches zero, the R^2 statistic approaches 1 indicating a perfect fit.

2.3 Design of Experiments for Kriging Models

In general the design problem is to find or select the locations (inputs) to predict unknown locations efficiently, whereby efficiency is measured in terms of minimum variance. Thus, it is often desirable to construct an optimal design for an experiment in order to avoid unnecessary time-consuming computer experiments. An optimal design of experiments depends on the statistical model (here Kriging) and is assessed using some statistical criterion. Sacks et al. proposed an integrated mean-squared error (IMSE) to assess a design [131]. Given n locations (design points) in d dimensions requires the optimisation of a large system of equations based on the IMSE criterion. One aspect that makes the solution of this optimisation method more challenging is that the parameters of the Kriging model equation (2.6) have to be assumed, because no data has been collected yet. So, the goal of finding the optimal design is a design that performs well over a wide range of unknown model parameters. Currin et al. extended this approach to the Bayesian setting and introduced an entropy criterion to assess the quality of a design [40].

Alternatively, designs can be algebraically constructed independent of the statistical model. In the following sections orthogonal column Latin Hypercubes (OLH) are presented [185], which is used for all design of experiments in this thesis.

In contrast to the approaches in [40, 131], Latin Hypercube Sampling is a systematic sampling technique, where the design space is subdivided into a homogeneous mesh. Each cell in this mesh represents a hypercube and a sample is generated for each segment such that each sample is unique in its row and column (in the two dimensional case). These designs can be extended to be space-filling as well and they are often employed because they improve the interpolation of Kriging. Criteria of space-filling designs are minimising the maximum distance (minimax) and maximising the minimum distance (maximin) between design points. So, an $n \times d$ Latin Hypercube consists of d permutations of the column vector sequence $S_n = \{1, 2, \ldots, n\}$. Orthogonal Latin Hypercubes add the constraint that every pair of the columns of the Latin hypercube has zero correlation, where the correlation between two vectors $\mathbf{u}$ and $\mathbf{v}$ is defined as

$$r = \frac{\sum_{i=1}^{n}(v_i - \bar{v})(u_i - \bar{u})}{\sqrt{\sum_{i=1}^{n}(v_i - \bar{v})^2 \sum_{i=1}^{n}(u_i - \bar{u})^2}}, \tag{2.21}$$

where $\bar{u}$ and $\bar{v}$ are the arithmetic means of the respective vectors.

The construction of the Latin Hypercube for $n = 2^m + 1$ rows and $2m - 2$ orthogonal columns is as follows. Let the top half of the Latin Hypercube be $\mathbf{T}$ (with dimensions $2^{m-1} \times (2m - 2)$). The bottom half is a mirror image of $\mathbf{T}$ with a centre point added to complete the orthogonal Latin Hypercube.

Definition 2.1. *Let the matrix $\mathbf{M}$ denote the matrix with the absolute values of the corresponding entry in $\mathbf{T}$. And let $\mathbf{S}$ be the matrix consisting of entries 1 or -1 to reflect the sign of the corresponding entry in $\mathbf{T}$. Therefore, matrix $\mathbf{T}$ is the element-wise product (Hadamard product) $\mathbf{M} \times \mathbf{S}$.*

The columns of $\mathbf{M}$ are constructed using permutations of $\left[1, 2, \ldots, 2^{m-1}\right]$ and consist of

$$\left\{\mathbf{e}, \mathbf{A}_i\mathbf{e}, \mathbf{A}_{m-1}\mathbf{A}_j\mathbf{e};\ \text{for } i = 1, \ldots, m-1;\ j = 1, \ldots, m-2\right\}, \tag{2.22}$$

where $\mathbf{e} = \left[1, 2, \ldots, 2^{m-1}\right]^T$ and $\mathbf{A}_k$ is defined as

$$\mathbf{A}_k = \underbrace{\mathbf{I} \otimes \cdots \otimes \mathbf{I}}_{m-k-1} \otimes \underbrace{\mathbf{R} \otimes \cdots \otimes \mathbf{R}}_{k}, \tag{2.23}$$

where $\mathbf{I}$ is the 2×2 identity matrix,

$$\mathbf{R} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \tag{2.24}$$

and $\otimes$ is the Kronecker product.

To achieve mutually orthogonal column vectors of the matrix $\mathbf{T}$, matrix $\mathbf{S}$ is constructed in the following way. The columns of $\mathbf{S}$ are defined as

$$\left\{\mathbf{1}, \mathbf{a}_i, \mathbf{a}_1\mathbf{a}_{j+1};\ \text{for } i = 1, \dots, m-1;\ j = 1, \dots, m-2\right\}. \tag{2.25}$$

The vector $\mathbf{a}_k$ is defined as

$$\mathbf{a}_k = \mathbf{B}_1 \otimes \mathbf{B}_2 \otimes \cdots \otimes \mathbf{B}_{m-1}, \tag{2.26}$$

where

$$\mathbf{B}_k = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \quad \mathbf{B}_i = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \tag{2.27}$$

for $i \neq k$. Finally, the columns of matrix $\mathbf{T}$ is given according to Definition 2.1 as

$$\{\mathbf{1} \times \boldsymbol{e}, \mathbf{a}_i \times \mathbf{A}_i\boldsymbol{e}, (\mathbf{a}_1\mathbf{a}_j) \times (\mathbf{A}_j\mathbf{A}_{m-1}\boldsymbol{e}); \\ \text{for } i = 1, \dots, m-1;\ j = 1, \dots, m-2\}. \tag{2.28}$$

Ye proved a few properties of pair-wise orthogonality of the columns of the $\mathbf{T}$ matrix [185]. The orthogonal properties of the independent variables are of particular interest for regression analysis in which one wishes the regression coefficients to be uncorrelated. However, the $\mathbf{T}$ matrix constructed above does not have good space-filling properties. Ye introduced an algorithm that permutes the $\mathbf{M}$ matrix to search for optimal designs based on integrated or maximum mean-squared error, entropy, and minimax or maximin distances [185]. In this thesis the criteria of maximising the minimum distance is used to find an optimal orthogonal Latin Hypercube design. Algorithm 2.2 presents the complete algorithm to establish an optimal orthogonal Latin Hypercube design of experiments and optimises the minimax criterion. The final steps of the algorithm scale the matrix $\mathbf{M}$, such that the columns are within the design region of the simulation experiment.

Algorithm 2.2: Optimal Orthogonal Latin Hypercube algorithm

```
Input: $m$, the constant to construct a $(2^m + 1) \times (2m - 2)$ Latin Hypercube
Input: $r$, the constant to define the $r$ different starting Latin Hypercubes
Input: $\mathbf{lb}$ and $\mathbf{ub}$, the vector for the lower and upper bounds of the design variables
       respectively
$\mathbf{e}$ = randperm($2^{m-1}$)
$\mathbf{M}$ = construct according to equation (2.22) given $\mathbf{e}$
$\mathbf{M}_t = \mathbf{M}$
$cr = \mathtt{minimax}(M)$
for $n = 1$ to $r$ starting Latin Hypercubes do
    for $i = 1$ to $2^{m-1} * (2^{m-1} + 1)$ permutations do
        $\mathbf{e}^1 = \mathbf{e}$
        $[r1, r2]$ = unique integers $(r1, r2) < 2^{m-1}$
        $e^1_{r1} = e_{r2}$
        $e^1_{r2} = e_{r1}$
        $\mathbf{M}$ = construct according to equation (2.22) given $\mathbf{e}^1$
        $cr_i = \mathtt{minimax}(M)$
        if $cr_i > cr$ then remember the best OLH design
            $cr = cr_i$
            $\mathbf{M}_t = \mathbf{M}$
        end
    end
    $\mathbf{e}$ = randperm($2^{m-1}$)
end
$\mathbf{S}$ = construct according to equation (2.25)
$\mathbf{T} = \mathbf{M}_t \times \mathbf{S}$
$\mathbf{L} = \begin{bmatrix} \mathbf{T} \\ 0 \\ -\mathbf{T} \end{bmatrix}$
$\mathbf{L} \leftarrow \mathbf{L} + 2^{m-1}$ move the design matrix into having positive integers
$\mathbf{L} \leftarrow \mathbf{L} * 1/2^m$ normalise the matrix to be unit length
Output: $\mathbf{L} \leftarrow \mathtt{scale}(\mathbf{L}, \mathbf{lb}, \mathbf{ub})$ scale the matrix columns to be within $[\mathbf{lb}, \mathbf{ub}]$
```

2.4 Parallel Kriging

Depending on the problem structure, the computational complexity of Kriging is either dominated by $O(n^2)$ (vector algebra) or $O(n^3)$ (matrix multiplication, matrix inversion, and determinant) operations. Solving equation (2.9) requires the calculation of the least-square estimate in equation (2.7) where the covariance matrix needs to be inverted.

The spatial datum in computer experiments can be very expensive to obtain. Also, non-linear behaviour within the input domain is usually localised in a small area. Therefore the sample size tends to be relatively small compared to modelling of geographic features, such as rain distribution over an entire continent [70]. Consequently, Kriging is relatively straightforward and the mean-squared error and Kriging estimator can be obtained in a timely fashion.

In the following sub-sections a Kriging process for stochastic computer experiments is presented as a fully integrated approach. First a pilot design is conducted (Section 2.4.1) which is then successively improved (Section 2.4.2).

2.4.1 Pilot Design

The activity diagram in Figure 2.3 details the parallel nature of the pilot design. The master and slave processes are on separate computer nodes, where the number of slave nodes can be adjusted to maximise the speedup of the simulation study, e.g., allocating $n \times r$ nodes, where n is the sample size and r is the initial number of replications.

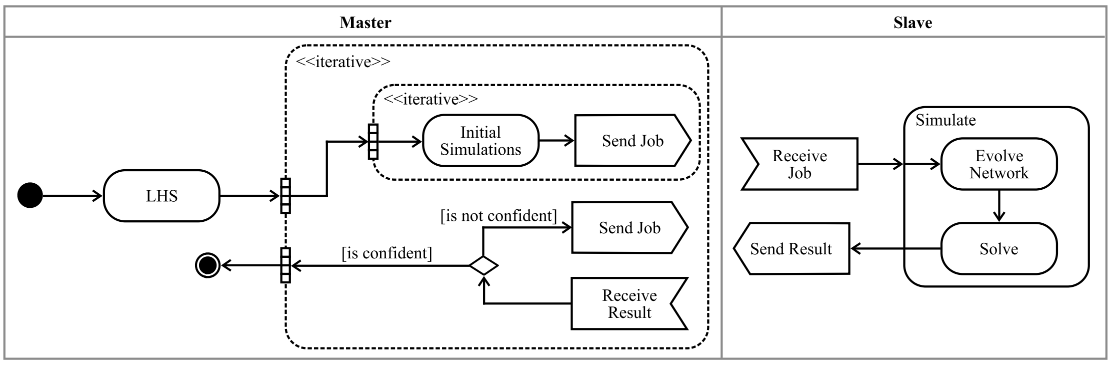


Figure 2.3: Initial Simulation Runs (Phase 1)

Given a specification of the boundaries of the input domain and the sample size, a pilot design using Latin-Hypercube Sampling (LHS) is conducted (Section 2.3). LHS is a stratified sampling techniques with space-filling properties. At each of the sampling points, a number of replications with different random seeds are performed. The master node sends a batch of the simulation parameters to slave nodes using the Message Passing Interface (MPI), which then run the simulation given the input parameters. The result of the simulation is sent back to the dedicated master node. Because a fixed sample size procedure on n replications does not give any control over the confidence interval half-lengths and to avoid running unnecessary expensive computer experiments, a sequential procedure is employed. To obtain an estimate of the mean with specified relative error of γ where $\gamma' = \gamma/(1+\gamma)$ and a confidence level of $100(1-\alpha)$, $\delta(n,\alpha)/|\bar{y}| \leqslant \gamma'$ is evaluated, where $\delta(n,\alpha)$

is denoted as

$$\delta(n, \alpha) = t_{n-1,1-\alpha/2}\sqrt{S^2(n)/n}, \tag{2.29}$$

where $S^2(n)$ is the sample variance and $t_{n-1,1-\alpha/2}$ is the inverted cumulative t-distribution function with $n-1$ degrees of freedom [86]. If and only if $\delta(n, \alpha)/|\bar{y}| \leqslant \gamma'$ then take $\bar{y}$ and the respective confidence interval and stop the simulation. If the objective of the experiment is to measure multiple responses, y_i, then $\delta(n, \alpha)/|\bar{y}_i|$ is evaluated for each one of them and the simulation is replicated until all responses are within the specified relative error and confidence levels.

2.4.2 Iterative Model Improvement

The iterative nature of learning a predictive model can be exploited to adaptively select samples to improve the prediction (also known as adaptive sampling or active learning) [25, 35, 67, 68, 160].

Once the pilot design is completed, the Kriging model is assessed, which determines whether more simulation runs are necessary to improve the coefficient of multiple determination, $R^2_{\text{prediction}}$ (equation (2.20)). The activity diagram in Figure 2.4 illustrates the iterative nature of Kriging model improvement.

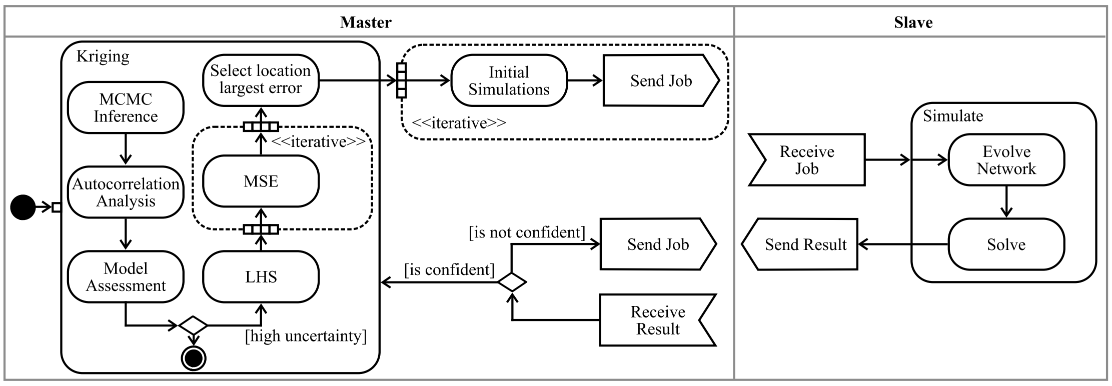


Figure 2.4: Iterative Model Improvement (Phase 2)

An initial interpolation may be done to get an idea on the length of the Markov chain, its mixing and the burn-in characteristics. The parameter of the adaptive Metropolis-within-Gibbs MCMC algorithm (Algorithm 2.1) that controls the dispersion and consequently the acceptance criterion is adjusted to achieve an acceptance rate of $\approx 23\%$ [61]. With this initial Kriging study conservative Markov chain lengths, burn-in threshold, and tuning parameters are used. The Kriging process is then performed on the master node, where the adaptive Metropolis-within-Gibbs MCMC simulation infers the length-scale parameters θ and σ^2 of the autocovariance function (equation (2.3)). The details of the Bayesian estimation approach are given in Section 2.2.1.2. Successive MCMC simulations are seeded with the previous mean values for the θ hyper-parameters. σ^2 is always estimated from the data directly (see Algorithm 2.1).

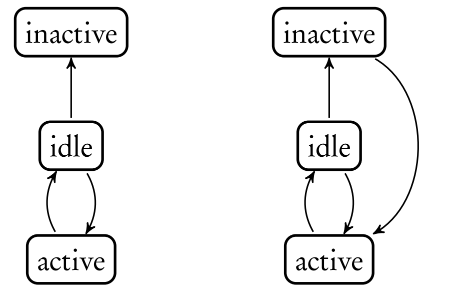
Given a Markov chain the autocorrelation functions are analysed using the approach from the ALPHA Collaboration and Ulli Wolff [176]. The integrated autocorrelation time is used to sample the Markov chain. The chain is further down-sampled to a maximum of 150 elements, if it is too long. The autocorrelation analysis is helpful to visualise the convergence properties of the Markov chains for each inferred parameter offline. The $R^2_{\text{prediction}}$ and the RMSE statistics are then calculated to determine whether more sample points are necessary to improve the prediction quality of the model. A space-filling LHS design with $m \gg n$ validation locations is chosen, if the $R^2_{\text{prediction}}$ statistic is below the defined 0.95 threshold. For each of those sample points the mean-squared error is calculated using equation (2.17) and the location with the largest error specifies the next sample point to be simulated. Similar to the pilot design, an initial number of replications are sent to the slave nodes incrementally evaluating the confidence half-lengths using equation (2.29).

Once the model achieves a good fit, global optima on the surface are determined using simulated annealing [64]. Simulated annealing uses a randomised neighbourhood search strategy to escape local minima, which makes it less likely to fail to converge on difficult functions [80]. Finding these global optima is embedded into the sequential improvement of the Kriging model as well, until the difference between the sampled optima and the interpolated ones are satisfactory.

2.4.3 Parallel Master-Slave Design

Designing a master-slave architecture for parallel computation of computer simulation is relatively straight-forward, but requires some attention on how idle nodes are allocated. Because there is an upfront division of labour, such that each simulation is allocated an initial number of nodes to conduct the replications in parallel, some computer simulations may need a higher number of replications to achieve statistically confident results than others (see equation (2.29)). The pilot design can achieve high speed-ups and efficiency rates, if no single simulation requires significantly more replications than the other ones. However, this can rarely be guaranteed upfront. Hence, an iterative scheme is required that utilises available resources efficiently. Experience has shown that an iterative scheme that schedules a replication one at a time, shows relatively poor parallel performance in terms of efficiency and speed-up. An improvement is achieved, if the replications are scheduled as r replications at a time, where r represents the initial number of replications. This way, the master node waits until all r replications return their result and only then computes the new confidence bands and evaluates whether further replications are required. Since, the master node “knows” that r slave nodes are idle these can immediately be utilised, if further replications are necessary. The disadvantage of this approach is that there is no book-keeping on which slave nodes are idle, because some simulation runs may have reached their desired statistical confidence levels while others may still be computing. If there is no book-keeping then those nodes can be considered inactive, because no jobs will be allocated to them. This degenerative case is depicted in Figure 2.5(a).

The inactive nodes can be recovered by keeping a list in the master node of all inactive nodes. As soon as a set of new replications are scheduled for a particular simulation run, all inactive nodes are included as well. This increases the currently active replication number from previously r (the initial replication number) to $r \leftarrow r + i$, where i is the current size of the inactive list. This progressive design is presented in Figure 2.5(b). As the overall parallel simulation progresses, an increasing number of individual simulation runs finish and put their respective nodes into



(a) Slave-node States (Degenerate)

(b) Slave-node States (Progressive)

Figure 2.5: Slave-node States

the inactive state. This implies that an increasing number of replications are scheduled for the remaining simulations that have not achieved their required confidence levels yet. Other schemes are easily employed, such as dividing the list of inactive nodes up equally among the remaining active simulations or to evaluate the confidence bands and assign the inactive nodes relative to the respective confidence levels, i.e., simulation runs with a higher current confidence level get a smaller number of available resources than those with a lower current confidence level. Graham and Keller [66] propose dynamic communicators for MPI that would be an interesting way of incorporating the progressive assignment of slave nodes to simulation runs, such that the world communicator is split into subworlds for each simulation run. The master node then communicates with the respective subworlds directly, without maintaining separate lists of idle and inactive nodes.

The iterative improvement of the Kriging model does not have the problem of possibly inactive nodes, because a single simulation is carried out, rather than many in parallel. However, this phase can lead to relatively poor speed-ups and efficiency rates, because inferring the Kriging parameters using Markov Chain Monte Carlo can dominate this phase for large data sets.

2.5 Canonical Analysis of Kriging Models

In computer simulations with two inputs and multiple outputs, a geometrical representation enhances the understanding of the simulation outputs, especially by representing each output variable as a contour plot given the inputs. Plotting the contours provides visual feedback on the sensitivity of each design variable with respect to the output. In higher dimensions geometrical representations of the contours becomes more difficult and therefore a more mathematical analysis is necessary. Canonical analysis is a mathematical framework to achieve greater insights into the sensitivities of each design variable in the vicinity of stationary points. In most computer simulation experiments in this thesis a canonical analysis is conducted to provide a more comprehensive view on the simulation results. This section is based on the books by Box and Draper [24], Montgomery [102], Myers et al. [107].

Recall that the purpose of building a model of (stochastic) computer experiments is to find suitable approximations for the true functional relationship between the independent variables and the simulation output. Kriging provides a first-order polynomial term coupled with a spatial correlation of the residuals to express curvature in the response. Least-squares regression without quadratic regressors is not able to capture curvature in the response.

Following the methodology of building and improving a model discussed in the previous sections leads to a response surface with a desired level of certainty within its domain. A canonical analysis can be conducted to characterise the nature of the surface in the vicinity of the stationary point, where the stationary point represents either a point of maximum response, a point of minimum response, or a saddle point. This assumes that the stationary point is within the design domain.

Kriging modelling gives a first-order polynomial function that is then used to determine the region of interest on the hyper-surface spanning the input domain. A second-order response surface

$$y = \beta_0 + \sum_{i=1}^{k} \beta_i x_i + \sum_{i=1}^{k} \beta_{ii} x_i^2 + \sum_{i<j}\sum \beta_{ij} x_i x_j + \epsilon \tag{2.30}$$

is fitted in the vicinity of the stationary point, which, if it exists has partial derivatives $\partial\hat{y}/\partial x_1 = \partial\hat{y}/\partial x_2 = \cdots = \partial\hat{y}/\partial x_k$. A general mathematical solution of the stationary point may be obtained from the fitted second-order model in matrix notation

$$\hat{y} = \hat{\beta}_0 + \mathbf{x}^T\mathbf{b} + \mathbf{x}^T\mathbf{B}\mathbf{x}, \tag{2.31}$$

where

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_k \end{pmatrix} \quad \mathbf{b} = \begin{pmatrix} \hat{\beta}_1 \\ \hat{\beta}_2 \\ \vdots \\ \hat{\beta}_k \end{pmatrix} \quad \mathbf{B} = \begin{pmatrix} \hat{\beta}_{11} & \hat{\beta}_{12}/2 & \dots & \hat{\beta}_{1k}/2 \\ & \hat{\beta}_{22} & \dots & \hat{\beta}_{2k}/2 \\ & & \ddots & \\ \text{sym.} & & & \hat{\beta}_{kk} \end{pmatrix}.$$

$\mathbf{b}$ is the column-vector of first-order regressors and $\mathbf{B}$ is a $(k \times k)$ symmetric matrix whose diagonal consist of the pure quadratic regressors and whose off-diagonal elements consist of the regressors of the interaction terms in equation (2.30).

Taking the derivative of $\hat{y}$ with respect to $\mathbf{x}$ and setting the derivative to zero

$$\frac{\partial\hat{y}}{\partial\mathbf{x}} = \mathbf{b} + 2\mathbf{B}\mathbf{x} = 0 \tag{2.32}$$

and solving for $\mathbf{x}$ defines the stationary point

$$\mathbf{x}_s = -\frac{1}{2}\mathbf{B}^{-1}\mathbf{b} \tag{2.33}$$

and the respective response by substituting equation (2.33) back into equation (2.31)

$$\hat{y}_s = \hat{\beta}_0 + \frac{1}{2}\mathbf{x}_s^T\mathbf{b}. \tag{2.34}$$

Canonical analysis is now concerned with examining the eigenvalues of the coefficient matrix, $\mathbf{B}$, of the second-order model. The canonical form of the model is then given as

$$\hat{y} = \hat{y}_S + \sum_{i=1}^{k} \lambda_i w_i^2, \tag{2.35}$$

where the $\hat{y}_S$ is the estimated point of the stationary point and λ_i are the eigenvalues of the second-order coefficient matrix. The variables w_i are called the canonical variables. The canonical variables refer to a transformed coordinate system with the stationary point at its origin and rotated until the w-axes are parallel to the principal axes of the contour system. For more details on the derivation of equation (2.35) see [24, 102, 107]. The signs of the eigenvalues determine the nature of the stationary point x_S. Mixed signs indicate a saddle point, while all positive or all negative signs indicate a point of maximum or minimum response respectively. Furthermore, the magnitude, $|\lambda_i|$, of the eigenvalues determines the gradient of the w_i direction on the surface, e.g., the w_i direction is steepest for which $|\lambda_i|$ is the greatest. Transforming the coordinate system to make the stationary point its origin is particularly useful if the stationary point is far away from the centre of the design space. Otherwise, a simpler canonical form can be used which is not stated here, but can be found in [24, 102, 107].

2.5.1 Ridge Analysis

Often, in computer simulations with more than one independent variable it is not unusual to encounter more complex response surfaces that cannot be characterised as having a pure maximum, minimum or saddle point. Ridge systems may arise due to some interaction or underlying dependence among the independent variables (or even not considered ones). For example, yield contours for an attenuated maximum in two variables have a banana shape; stationary ridge systems can be understood as having an optimum on a line; a rising ridge system displays a stationary point outside the design region. In three dimensions these systems can easily be investigated by looking at the contour plots. However, as before with pure optima, with more than two independent variables a formal study can be of great assistance.

In the most general case, a ridge analysis can be conducted over a specified region of interest to follow the locus (or path) of a maximum response. The origin of the path can be situated anywhere within the design region and following on the path of maximum response gives a trace of each independent variable. Consequently, variables of high sensitivity can easily be identified independent of the size of the design space. Usually, only the paths of steepest ascent or steepest descent are of interest. However, more paths are easily constructed. In fact, if there are k independent variables, there will be k eigenvalues and $2k$ paths. This can easily be understood geometrically, where each dimension can be followed towards a minimum response or a maximum response. This has the advantage that graphical output can be produced without holding any variables fixed. Additionally, local optima can be investigated as well. However, this procedure has its drawbacks as well, because the respective plots following a defined path are not as easily understood as contour plots, and the analysis depends on the use of a quadratic model.

More formally, the stationary point of equation (2.30) subject to being on a sphere with radius R and centered at a focal point $\mathbf{f}$ with equation

$$(\mathbf{x}-\mathbf{f})^T(\mathbf{x}-\mathbf{f}) \equiv (x_1-f_1)^2+(x_2-f_2)^2+\cdots+(x_k-f_k)^2 = R^2 \tag{2.36}$$

is obtained by considering the Lagrangian function

$$F = \beta_0 + \mathbf{x}^T\mathbf{b} + \mathbf{x}^T\mathbf{B}\mathbf{x} - \mu((\mathbf{x}-\mathbf{f})^T(\mathbf{x}-\mathbf{f}) - R^2), \tag{2.37}$$

where μ is a Lagrangian multiplier. Differentiating equation (2.37) with respect to $\mathbf{x}$ gives

$$\frac{\partial F}{\partial x} = \mathbf{b} + 2\mathbf{B}\mathbf{x} - 2\mu(\mathbf{x}-\mathbf{f}). \tag{2.38}$$

Setting this to zero leads to

$$2(\mathbf{B}-\mu\mathbf{I})\mathbf{x} = -\mathbf{b} - 2\mu\mathbf{f}. \tag{2.39}$$

So, fixing equation (2.36) and maximising equation (2.31) subject to this constraint, a maximum, $\hat{y}(R)$, is defined for any given R. Selecting a value for $\mu \neq \lambda_i$, $(\mathbf{B}-\mu\mathbf{I})^{-1}$ exists and a solution for $\mathbf{x}$ for a stationary point on a sphere of radius R can be obtained by solving

$$\mathbf{x} = -\frac{1}{2}(\mathbf{B}-\mu\mathbf{I})^{-1}(\mathbf{b}+2\mu\mathbf{f}). \tag{2.40}$$

In order to visualise the derivation of the mathematical formulae, consider the Branin function, which is defined as

$$y(\mathbf{x}) = (x_2 - \frac{5.1}{(4*\pi^2)} * x_1^2 + \frac{5}{\pi} * x_1 - 6)^2 + 10*(1-\frac{1}{8*\pi})*\cos(x_1) + 10 \tag{2.41}$$
$$x_1 \in [-5, 10], x_2 \in [0, 10].$$

The function is plotted in Figure 2.6.

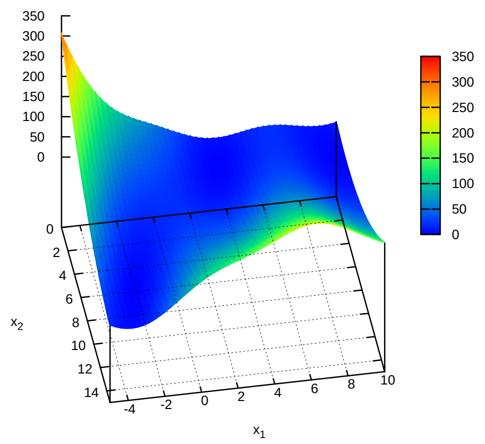


Figure 2.6: Branin Function

The Branin function within the domain specified above has a global maximum at $\mathbf{x} = (-5, 0)$. Because this maximum is attained at the edge of the domain, the stationary point is located outside of the prescribed domain. A suitable tool to analyse the sensitivities of the two variables, x_1 and x_2, is ridge analysis in the vicinity of the global maximum. For this purpose the design domain for the ridge analysis is constrained to be within $x_1 = [-5, 0]$ and $x_2 = [0, 5]$. The focal point is defined to be the centre of experimentation and is set to $f = (-2.5, 2.5)$. Ridge analysis proceeds by constructing a mental sphere with radius R around the focal point and recording the respective independent variables, x, where the sphere intersects with the function. Figure 2.7 provides a visual representation of this process.

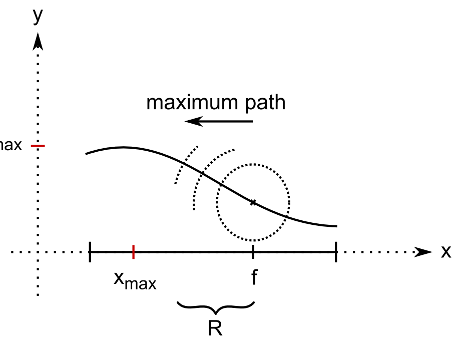


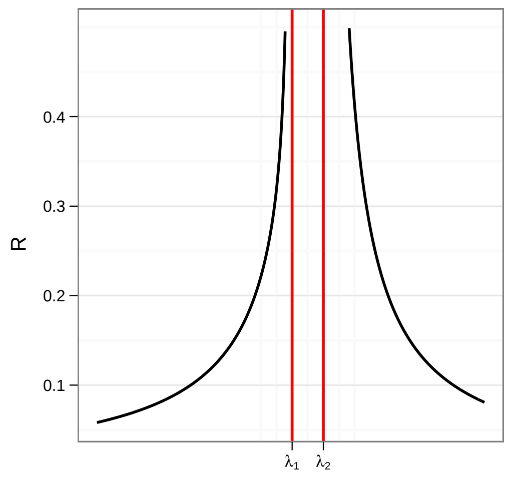
Figure 2.7: Illustration of tracing the maximum path

Further, a path (or direction) is identified by moving away or towards the eigenvalues of the coefficient matrix, $\mathbf{B}$. Both R and x depend on the numerical value of μ. The eigen-spectrum reveals the role of the eigenvalues, λ, to the ridge analysis. From a practical perspective, it reveals the range of the eigenvalues to be substituted into equation (2.40) for μ. Moving μ from $-\infty$ towards the smallest eigenvalue, λ_1, of the coefficient matrix $\mathbf{B}$ and solving for x (equation (2.40)) and R (equation (2.36)) along the way always gives the "minimum $\hat{y}$" path [24] (left-hand side of Figure 2.8(a)). Vice versa, moving μ from the largest eigenvalue, λ_k, to $+\infty$, provides the "maximum $\hat{y}$" path (right-hand side of Figure 2.8(a)).

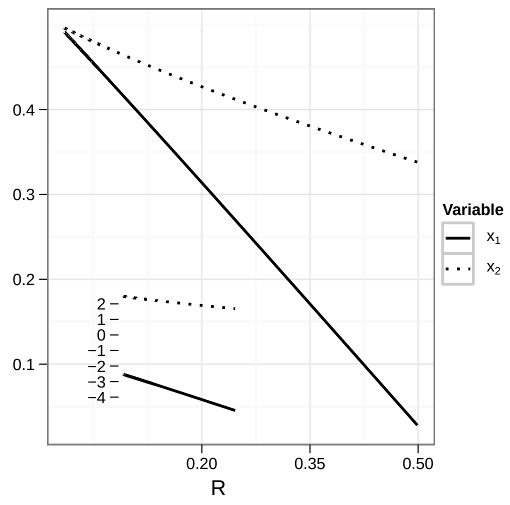
The values of the independent variables x_1 and x_2 along the "maximum $\hat{y}$" path are displayed in Figure 2.8(b) (with the original scale of the independent variables in the inset). The respective change in the response variable is given in Figure 2.8(c) defined as path number 3. For completeness, the minimum ridge path for $\hat{y}$ is also presented. The vertical asymptotes are the respective eigenvalues of the coefficient matrix, $\mathbf{B}$, of this ridge analysis. In order to visualise the effect of the rate of change with respect to all independent variables involved, the range of all variables is scaled to unity. The focal point is placed such that the radius of the sphere expanding towards the global maximum covers half the unit distance for all independent variables within the domain considered for the ridge analysis. One can see that moving from the focal point towards the global maximum, the value of x_1 decreases faster than that of x_2. Moreover, both variables do not enter a stable regime, which in this case implies that both are exactly on the maximum ridge that extends outside the design domain.

This is an example of a simple ridge system in three dimensions. The advantages of conducting this ridge analysis becomes apparent in higher-dimensional spaces where visualising contour plots is more difficult, but an analysis of the sensitivities of the independent variables is necessary.

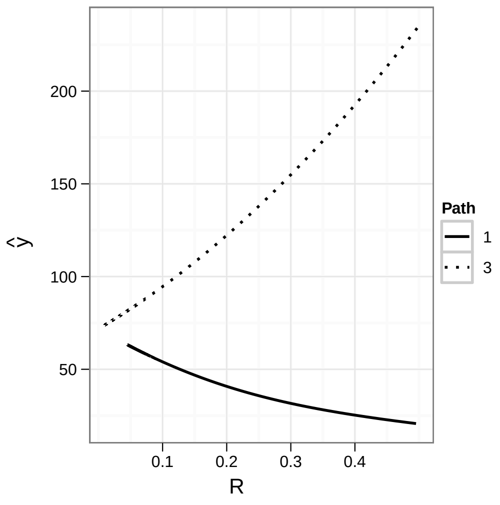
If the ridge analysis is conducted for a model of a real-world system, then it is often desirable to identify stable regimes for the independent variables. In an industrial setting, it is often not possible to operate under exact levels and consequently stable points are preferred over optimal points that



(a) Spectrum of the eigenvalues and their paths



(b) "Maximum $\hat{y}$" path for the independent variables



(c) "Maximum $\hat{y}$" path

Figure 2.8: A sample ridge analysis

may show erratic behaviour when departing from optimal settings [72].

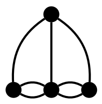
2.6 Summary

As simulation studies are becoming more prevalent in computer science, systematically controlling the experiment or a thorough analysis is often neglected. This chapter presented response surface methodology (RSM) comprising a collection of well-known mathematical and statistical techniques to design, analyse, and optimise simulation experiments based on the books by Box and Draper [24], Montgomery [102], Myers et al. [107]. Within this context, the focus was placed on stochastic computer experiments and as such the tools presented can treat any simulation as a black box. While often graphical investigations are useful for low dimensions a more formal study of computer simulations is of particular interest in higher dimensional design domains. The interpolation of the stochastic simulation outputs or responses is rooted in the realm of geostatistics. Section 2.2 presented an introduction into spatial data modelling and concentrated on a technique called stochastic Kriging. This approach can be understood as an extension to regression models that is capable of integrating a trend surface (as in regression) and spatial correlation of the residuals. That way, non-linear relationship between the independent variables and the response can be captured with a first-order polynomial function. In fact, often this polynomial can be reduced to a constant and therefore force the spatial function to learn the spatial pattern. A fully Bayesian analysis was presented that accounts for all sources of uncertainty, i.e., the stochastic uncertainty of the simulation output, the parametric uncertainty about predicting responses at unobserved locations in the design domain, and model uncertainty that covers the uncertainty resulting from the Markov Chain Monte Carlo inference of the model parameters.

In most computer simulations the independent variables are subject to the control of the scientist. However, in the absence of a frame of reference on the simulation behaviour, a design of experiment can be a great challenge. Section 2.3 presented an optimal and orthogonal Latin Hypercube design of experiments to place locations to be sampled in the design domain. An initial pilot design can reveal a great deal of the underlying response(s) of the simulation experiment. The sequential nature of model building under the Kriging framework is exploited in a high-performance implementation for clustered computing environments introduced in Section 2.4. A technique of active learning is used to find optima on a model surface and select potentially new locations to improve the model.

Once a satisfactory model is constructed, it is often desirable to understand the dependence of the independent variables on the response in the vicinity of the optima. Canonical analysis and ridge analysis presented in Section 2.5 offer a mathematically sound framework to reveal how sensitive the simulation response is to changes in the respective independent variables.

These tools together are frequently used throughout this thesis to gain detailed insights into the simulations under investigation.


Leonhard Euler, 1736 (Seven Bridges of Königsberg)

3 Modelling Distributed Task Assignment Problems

Simulation studies play an ever increasing role in analysing and understanding communication or traffic systems. This is especially the case for complex adaptive systems, where closed-form solutions do not exist, to address techniques such as online learning for optimising routing behaviour [46, 49]. However, the underlying network structure or topology is treated as a secondary evaluation aspect and often neglected. Recent work in this area focused on applying social network analysis to optimise routing behaviour [45], yet the reliance on particular network instances remains.

Starting with the seminal papers on small-world and scale-free networks [15, 53, 171], complex network research has shaped a variety of research agendas across many domains, progressing rapidly to provide new insights, bringing tools well-known to physicists into a domain that was previously dominated by sociologists. Based on those studies the investigation of dynamic evolution and growth of networks attracted increasing interest to explain the emergence of complex structural features [15, 16, 168]. While contributions to the field of complex network analysis have been substantial, similar theoretical results with respect to traffic patterns remain elusive and are in the early stages. Tadić et al. [146], Yin et al. [186] investigated aspects of the functional performance of networks by coupling network topologies and transport processes. Tadić et al. [146] compared a scale-free network topology to a more homogeneous network with respect to traffic dynamics. They showed that flow paths are better for the scale-free network and as a result travel times are significantly reduced. However, network growth models and traffic pattern analysis are still divorced from the analytical framework of queueing systems [21, 76]. A combined approach would contribute significantly to distributed systems studies. On the one hand, network growth models with different structural features can be integrated into large-scale simulation studies and on the other hand the formalisms of queueing theory provides a rich framework for evaluating the system performance.

Queueing theory deals with the mathematical study of queues, which occur whenever demand exceeds the operational capacity to provide a service. Their general applicability and the growing interest in uncovering traffic patterns in complex networks indicate a compelling combination of queueing theory and complex networks research. Section 3.1 introduces some basic analytical results of open and stable queueing systems, i.e., with external traffic sources and destinations and a utilisation rate of smaller than one respectively. The stability criterion is important, because it ensures that analytical steady-state solutions exist. Then (Section 3.2) presents two complementary network evolution models which are applied to establish a stable and open queueing network with directed edges in between the nodes. This is a novel approach providing a framework which is exploited in large-scale simulation studies of complex adaptive communication systems in this thesis. Section 3.2.2 details a model that exhibits a co-evolution of node properties and interaction patterns and scale-free behaviour of the node properties based on the Barrat, Barthèlemy, and Vespignani model (BBV) [16]. BBV identified the need to extend existing evolution models, which generally only take into account the topological structure and not the dynamic interaction with the introduction of new nodes. Real networks, however, rely on crucial time-varying quantities. For example, the number of passengers passing through an airport in the well-studied airline network plays an important role in modelling and analysing epidemic outbreaks [36]. The interplay of the structure of the airport network and the infection dynamics being modelled affect dramatically the outbreak scenarios. BBV recognised that coupling the topological structure and the dynamic interaction yields a more realistic network.

It is of paramount importance to understand the network's functional properties with respect to its structural features. Therefore, as a complementary model, the Erdős and Rényi (ER) random graph model with homogeneous structures is presented in Section 3.2.3.

At the heart of both adapted models are two tuning parameters allowing the adjustment of the stability criterion at the periphery and towards the center of the networks, where stability is preserved if both parameters are $\geqslant 1$. If either of the two parameters is smaller than 1, then the utilisation will exceed 100% at certain nodes which will result in diverging performance metrics.

It seems intuitive to imagine that mean total waiting time will increase linearly with respect to network utilisation. However, it turns out that the linear behaviour can only be observed for utilisation rates below approximately 80%. If utilisation increases above this value, the mean total waiting time increases disproportionately. It is this sensitive regime that is being investigated in this chapter, focusing on how a scale-free network (BBV model) compares against a homogeneous network (ER model). The benefits of this study are the provision of a deeper understanding of queueing systems with respect to functional features of networks. Additionally, complex adaptive systems simulation will be greatly enhanced, if the adaptation is analysed (where possible) with respect to those functional features instead of particular network instances.

3.1 Queueing Systems

Queueing Theory is the mathematical study of a queue's behaviour under certain load dynamics [21]. Generally, it is considered a branch of operations research [118], however it is applicable to a variety of application domains, such as transport [162] and communication networks [182], permitting the derivation of several performance metrics. In the following paragraphs emphasis is placed on the mean total waiting time (delay), W, and the utilisation, $\mathcal{U}$, of a network.

A Queueing Network consists of elementary inter-linked service stations or nodes. Each node in the queueing networks considered for this thesis can only serve a single customer at a time, where customers arrive with Poisson arrival rates, λ, and are being served at an exponential rate, μ. In Kendall's notation this is defined as an $M/M/1$ queue with a First-Come-First-Served (FCFS) queueing discipline, where the M stands for Markovian arrival and service processes respectively. An obvious example is the Internet, where routers represent nodes and a packet leaving one router either proceeds to the next one or leaves the system. Because the operating conditions of each service station dictates the rate at which customer requests are processed, queues are embedded to hold requests that cannot be processed at the time of arrival. Usually, queues present finite memory so that once the queue is full, customer requests are dropped. Packets may also be associated with priorities, allowing service stations to place an incoming request in front of lower priority requests into the queue. Such queueing systems where customers belong to a particular priority class are called multi-class networks. In this thesis, only single-class networks are considered and each queue presents theoretical infinite memory. So, no request is dropped from the queues.

For a steady-state solution to exist, the stability condition $\rho = \lambda/\mu < 1$ has to hold. Otherwise the performance measurements diverge and no meaningful result can be obtained. Since a Queueing System consists of many inter-dependent nodes, a network of queues needs to be reasoned about. In particular open queueing networks (OQN) are considered, where traffic may enter or leave at one or more nodes of the system from an external source. The definition of an open queueing network with Poisson arrival rates and exponential service rates that is used throughout this thesis is given by

Definition 3.1 (Open Queueing Network). *An **Open Queueing Network** is defined by*

- $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_N)$, *the exponential service rates.*
- $\boldsymbol{\lambda}_0 = (\lambda_{01}, \ldots, \lambda_{0N})$, *the vector of external Poisson arrival rates;*
- $\boldsymbol{\lambda}_i$, *the aggregate arrival rate at the ith node;*
- λ, *the overall arrival rate of external jobs to the network, i.e.,* $\sum_{i=1}^{N} \boldsymbol{\lambda}_{0i}$*;*
- $\mathbf{Q}$, *the matrix of routing probabilities,* $\mathbf{Q}_{ij}$, *from node i to node j.*

3.1.1 Analytical Equations

Usually, the vector of aggregate arrival rates, $\boldsymbol{\lambda}$, of an open queueing network is not given and has to be calculated given the external arrival rate, $\boldsymbol{\lambda}_0$, and the routing probabilities, $\mathbf{Q}$. Thus, the aggregate arrival rate not only accounts for the external arrival rates, but also integrates the arrival rates from all other nodes. In statistical equilibrium the rate of departure is equal to the rate of arrival, so the aggregate arrival rate, $\boldsymbol{\lambda}_i$, at node i is calculated as:

$$\boldsymbol{\lambda}_i = \lambda_{0i} + \sum_{j=1}^{N} \boldsymbol{\lambda}_j \mathbf{Q}_{ji}, \quad \text{for } i = 1, \ldots, N, \tag{3.1}$$

for each node in the network. Equations (3.1) are known as traffic equations. These can easily be solved in matrix notation as

$$\lambda = \lambda_0(\mathbf{I} - \mathbf{Q})^{-1}. \tag{3.2}$$

Assuming that the stability criterion holds

$$\rho_i = \frac{\lambda_i}{\mu_i} < 1 \; \forall \, i \in 1, \dots, N, \tag{3.3}$$

then Jackson Theorem is stated as follows.

Theorem 3.1 (Jackson Theorem). *If in an open network ergodicity ($\lambda_i < \mu_i$) holds for all nodes* $i = 1, \dots, N$, *then the steady-state probability of the network can be expressed as the product of the state probabilities of the individual nodes, that is, [76]*

$$\pi(k_1, k_2, \dots, k_N) = \prod_{i=1}^{N} (1 - \rho_i)\rho_i^{k_i}, \tag{3.4}$$

where $\pi(k_1, k_2, \dots, k_N)$ is the state probability of the network given that node i has exactly k_i jobs waiting in its queue. This result, also known as the product-form solution, implies that the number of jobs at any node is independent of the number of jobs at any other node. Because of this independence, equation (3.5) gives the number of jobs in queue i as

$$\mathbf{L}_i = \frac{\rho_i}{1 - \rho_i}. \tag{3.5}$$

Also, Little's Law defined as $\mathbf{L} = \lambda \boldsymbol{W}$, can be applied to derive the mean total waiting time, $\hat{W}$, given in equation (3.6) [21]. The mean utilisation, $\hat{U}$, of the network is given in equation (3.7).

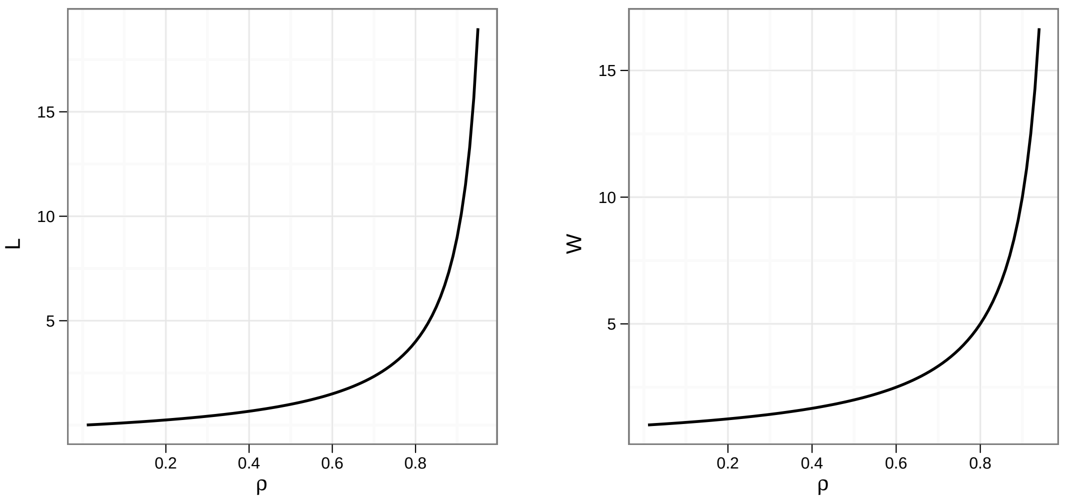
$$\hat{W} = \frac{1}{\gamma'} \sum_{i=1}^{N} \frac{\rho_i}{1 - \rho_i}, \text{ where } \gamma' = \sum_{i=1}^{N} \gamma_i \tag{3.6}$$

$$\hat{U} = \frac{1}{N} \sum_{i=1}^{N} \frac{\lambda_i}{\mu_i}. \tag{3.7}$$

Based on Little's law the mean number of events in an $M/M/1$ queue (equation (3.5)), and the mean response time, $W = L/\lambda = \frac{1/\mu}{1-\rho}$ with $\mu = 1$, for varying utilisations is shown in Figure 3.1(a) and Figure 3.1(b) respectively. Both curves show the non-linear behaviour of queueing performance with respect to the utilisation. As $\rho \to 1$, both the mean number of events and the mean response time grow to infinity and thus the queue tends to get unstable. This trend can be observed for utilisation rates of 80% or higher.

3.1.2 Discrete Event Simulation

Distributed task assignment modelled as decentralised multi-agent adaptive systems using queueing networks imply dynamically changing routing probabilities, because those routing probabilities are subject to the learning algorithm. So, analytical solutions described in Section 3.1.1 do not apply.



(a) Mean number of events vs. utilisation (b) Mean response time vs. utilisation (for $\mu = 1$)

Figure 3.1: M/M/1 queueing performance

Instead discrete event systems are used to measure the queueing performance metrics, where at every step in the simulation the performance measurements are updated accordingly.

A delay in a queue occurs when an event arrives and cannot immediately be serviced because the status of the server is busy. Since the event being serviced has a departure time associated with it, the delay can be calculated as the difference of the departure time of the event being serviced and the event that just arrived at the server. If the queue has already other events waiting, then the departure time of the last event in the queue has to be taken as the basis for the delay calculation enforcing the first come first serve queueing discipline. The *expected average waiting time*, $\hat{w}$, is the running average of all delays that occurred in a queue, i.e.,

$$\hat{w} = \frac{\sum_{i=1}^{n} D_i}{n}. \tag{3.8}$$

Using Welford's algorithm, the mean can be calculated incrementally, without keeping a full history of all observed delays [173].

The estimate of *the average number of events* in a queue during the simulation can be expressed as

$$\hat{q} = \frac{\int_0^T Q(t)dt}{T}, \tag{3.9}$$

where $Q(t)$ is the current number of events at time t, and T is the stopping time of the simulation [86]. Equation (3.9) calculates the continuous average of $Q(t)$ as the simulation progresses through time and needs to be evaluated instantaneously after events arrive to and depart from a server.

To measure the utilisation, u, of a server, which is the proportion of time the server is busy, a "busy-function" is defined as

$$B(t) = \begin{cases} 0 & \text{if the server is busy at time } t \\ 1 & \text{if the server is idle at time } t. \end{cases} \tag{3.10}$$

The *estimated utilisation* as a continuous-time average is then calculated as [86]

$$\hat{u} = \frac{\int_0^T B(t)dt}{T}. \tag{3.11}$$

For all events leaving the system at node i, the estimate of the *average event processing time* can be computed as

$$\hat{s} = \frac{\sum_{i=1}^{n} r_i}{n}, \tag{3.12}$$

where r_i is the response time of the event leaving the system at node i. The event processing time is the duration of an event in the system, measured as the difference of the time the event leaves the system and the arrival time.

3.2 Network Evolution Models

Based on the results of the Queueing Theory presented in Section 3.1, this section's focus turns towards two complementary network evolution models, "social" and "random", that meet the criteria of openness and stability, i.e., traffic events can enter and leave the queueing network and the utilisation is below 100% respectively. First, however, some background on algebraic graph theory is given in Section 3.2.1. Some node properties, such as degree, strength, assortativity, and the cluster coefficient for weighted directed networks are given in order to characterise the network evolution models in terms of their structural features.

3.2.1 Algebraic Graph Theory

In this section, some of the necessary algebraic graph theoretical concepts are introduced.

Definition 3.2 (Directed graph). *A **directed graph (digraph)** is a tuple $\mathcal{G} = \langle \mathcal{V}, \mathcal{A} \rangle$, where $\mathcal{V}$ is a finite set of vertices and $\mathcal{A}$ is a finite set of arcs $\mathcal{A} \subseteq \mathcal{V} \times \mathcal{V}$. For an arc $a = (u, v) \in \mathcal{A}$, u is called the head vertex and v is called the tail vertex of a.*

If $(u, v) \in \mathcal{A} \; \forall \; (v, u) \in \mathcal{A}$, then the graph is undirected. Further, if the graph does not have any self-loops, i.e., arcs of the form (u, u) for $u \in \mathcal{V}$, or multiple arcs with the same head and tail vertex, then the graph is simple. Throughout this thesis the simple graph $\mathcal{G}$ represents the communication links between the server nodes in the distributed task assignment network. The connectivity of such a graph is captured by the adjacency matrix $\mathbf{A}$ defined by:

Definition 3.3 (Adjacency Matrix). *The **adjacency matrix** of a directed graph $\mathcal{G}$ without self-loops, denoted as $\mathbf{A}$, is an asymmetric square matrix with dimension $|\mathcal{V}| \times |\mathcal{V}|$ defined as follows:*

$$\mathbf{A}_{ij} = \begin{cases} 1 & \textit{if } (u_i, u_j) \in \mathcal{A} \textit{ and } u_i, u_j \in \mathcal{V} \\ 0 & \textit{otherwise,} \end{cases} \tag{3.13}$$

where $\mathbf{A}_{ij}$ denotes the matrix element at row i and column j.

Definition 3.4 (Degree of a Vertex). *The **in-degree** of a vertex* u *is the number of arcs that have* u *as their head vertex. The in-degree is then given as:*

$$\deg^{-}(u_i) = \sum_{j=1}^{N} \mathbf{A}_{ij}, \text{ where } u_i \in \mathcal{V}. \tag{3.14}$$

Analogously, the out-degree of a vertex u_i *is defined as*

$$\deg^{+}(u_i) = \sum_{i=1}^{N} \mathbf{A}_{ij}, \text{ where } u_i \in \mathcal{V}. \tag{3.15}$$

In the context of distributed task assignment networks and their connectivity definition above, a number of servers participate in the completion of a task. The servers that are involved in the completion of a task form a directed path from the external arrival of a request to the completion. More formally:

Definition 3.5 (Directed path). *Given a directed graph* $\mathcal{G}$ *without self-loops,* $P = (u_1, u_2, \dots, u_k)$ *is a **directed path** in* $\mathcal{G}$*, if for every* $1 \leqslant i < k \; \exists \; (u_i, u_{i+1}) \in \mathcal{A}$*.*

Additionally, most networks are intrinsically weighted, which means that an arc (in the directed case) connecting node i to node j can provide a richer qualitative measure than just a binary value as in the adjacency matrix. For example some social relationships may be stronger than others. Or in the case of distributed task assignment networks more transactions per time unit may be routed down one path compared to another. Not only is this additional information necessary to analyse queueing systems where the interpretation of an arc carries a probability of routing traffic along this path, but also from a topological perspective where different semantics for the degree, cluster coefficient, etc. of a node exist.

Likewise to representing a graph as an adjacency matrix (see Definition 3.3) a weight matrix has the weights as entries instead of simply a binary value. Since, the weights of queueing networks are probabilistic in nature, their respective weight values are calculated as

$$\boldsymbol{W}_{(i)}^{(\text{actual})} = \boldsymbol{\lambda}_i * \mathbf{Q}_{(i)} \tag{3.16}$$

$$\boldsymbol{W}_{(i)}^{(\text{pot})} = \boldsymbol{\mu}_i * \mathbf{Q}_{(i)} \tag{3.17}$$

$\forall \; i \in \mathcal{V}$ where $\mathbf{Q}$ are the routing probabilities of the queueing network and $\boldsymbol{\lambda}$ is the vector of aggregated arrival rate as given in equation (3.2). The actual weights $\boldsymbol{W}_{(i)}^{(\text{actual})}$ for arcs leaving node i represents the amount of traffic that travels along the respective paths in steady-state, while the potential weights $\boldsymbol{W}_{(i)}^{(\text{pot})}$ represent the maximum possible amount of traffic flow that node i can handle without becoming unstable.

The strength or weighted degree generalises the degree definition for unweighted networks (equations (3.14) and (3.15)) as

$$s_j^- = \sum_{i=1}^{N} W_{ij} \qquad (3.18) \qquad\qquad s_i^+ = \sum_{j=1}^{N} W_{ij}. \qquad (3.19)$$

In queueing networks the in-strength represents the traffic inflow from within the network, while out-strength represents the traffic leaving the respective node.

Another important aspect is the idea of how entities in a network couple with one another. Networks can be characterised by the way choices are made about who to connect to. In that sense, some connections are more probable than others. In social networks, assortative mixing is often observed. This occurs when nodes with a high degree have a tendency to connect to neighbours with high degrees [110]. In contrast, economic or technical networks exhibit disassortative mixing. This is when the nearest neighbours of nodes with a high degree have a low degree. For directed networks the degree-degree correlation can be defined as

$$k'_{nn}(i) = \frac{1}{\deg^-(i)} \sum_{\forall\, j \in \upsilon(i)} \deg^+(j), \qquad (3.20)$$

where $\upsilon(i)$ denotes the set of first-order neighbours of node i [98]. Taking weights and the directional attribute of edges into account this can be extended to

$$k^w_{nn}(i) = \frac{1}{s^-(i)} \sum_{\forall\, j \exists (i,j)} W_{ij} \deg^+(j) + \sum_{\forall\, j \exists (j,i)} W_{ji} \deg^+(j). \qquad (3.21)$$

Equation (3.21) correlates the in-degree of node i with the out-degree of its first-order neighbours. The extension given in equation (3.21) puts a connection weight on the respective neighbours' out-degree value. The weight matrix W is not symmetric, because the networks considered in this thesis are directed as well as weighted.

It is often of interest to take the average over nodes with the same in-degree to compare the degree-degree correlation

$$\bar{k}_{nn}(k) = \frac{1}{NP(k)} \sum_{\deg^-(i)=k} k_{nn}(i), \qquad (3.22)$$

where $NP(k)$ is the number of nodes with in-degree $\deg^-(i) = k$. For $\bar{k}^w_{nn}(k) > \bar{k}_{nn}(k)$ the edges with the larger weight are directed to the neighbours with larger degrees, and for $\bar{k}^w_{nn}(k) < \bar{k}_{nn}(k)$ the edges with the larger weight are directed towards neighbours with lower degrees [98].

A metric of great interest to networks researchers is the clustering coefficient that expresses the tendency to cluster into small groups [171]. More specifically, picking three nodes i, j, and k in a network and letting i be connected to j and k, one may ask what the probability is that j and k will also be connected. Evidence in social and technical networks suggests that this probability is greater than random [109, 171]. In the unweighted and undirected case the global clustering coefficient is a ratio between the number of closed triangles and the number of total triangles which include open and closed ones, where triangles are three nodes either connected with two (open) or three edges (closed).

$$C = \frac{\sum \tau_\Delta}{\sum \tau}, \tag{3.23}$$

where $\sum \tau$ represents the total number of triangles and $\sum \tau_\Delta$ denotes the subset of these triangles that are closed. Equation (3.23) cannot be applied to directed networks. A related measure is called transitivity defined as

Definition 3.6 (Transitivity)**.** *The triangle involving nodes* i, j *and* k *is transitive, whenever arcs* (i, j) *and* (j, k) *imply* (i, k)*. If either of the two arcs,* (i, j) *and* (j, k)*, is not present, then the triangle is termed vacuously transitive. Vacuously transitive triangles are neither transitive nor intransitive [139].*

Opsahl and Panzarasa extended the cluster coefficient according to Definition 3.6 to include weights as well [114]. In their calculations a triangle carries a value, ω, as either the arithmetic mean, geometric mean, maximum, or minimum of the weights covered by the triangle.

$$C_\omega = \frac{\sum_{\tau_\Delta} \omega}{\sum_\tau \omega}, \tag{3.24}$$

where the numerator does not include the vacuously transitive triangles.

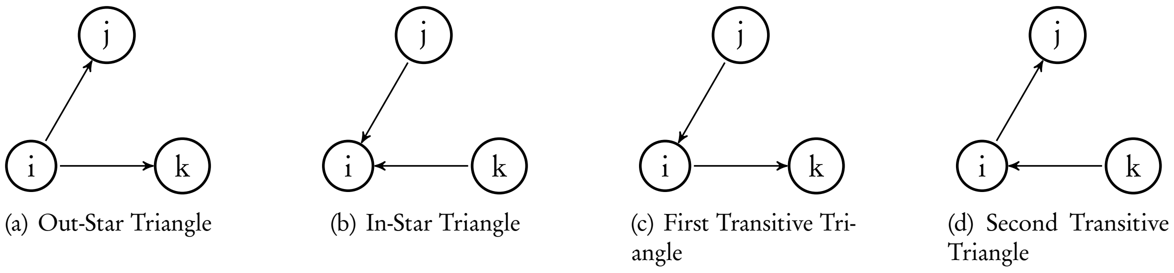


Figure 3.2: Triangles for transitivity in digraphs with no loops

Figure 3.2 presents the four possible triangles for transitivity. Figure 3.2(a) and Figure 3.2(b) are vacuously transitive and are therefore not considered in the cluster-coefficient calculation.

3.2.2 Social Model

Understanding and characterising network structures in communities as diverse as Ecology, Biology, and Computer Science brought about activities in network research to interpret complex topological properties, such as small-world and scale-free behaviours. The dynamic evolution and growth of networks plays an important role in understanding the complex topological features. While some network models focused only on the topological structure, such as the Barabási-Albert model [15], Barrat et al. [16] (BBV) also characterised how networks grow using two quantities: node strength and edge intensity or weight. In an airline network, the node strength corresponds to the number of passengers passing through an airport and the edge intensity corresponds to passengers travelling from one airport to a destination airport. The co-evolution of node properties and interaction patterns of the BBV model are particular interesting and is in the following treatment adapted for the evolution of queueing networks.

In its original setting to model airline networks, at each evolutionary step of the BBV model a new node is added, preferentially attaching to existing nodes with a high strength. Since the addition of a new node presents passengers to the network, variations to the nodes' strength attribute are introduced. In BBVs model the strength attribute of a node i is a derived quantity which is calculated as the sum of the edge weights of all adjacent nodes j connecting to node i. In contrast to their model, our adaptation introduces the time-varying changes to the nodes' strength directly, without changing the interaction dynamics. It is, however, necessary to cast network evolution models into a queueing-theoretic context. This way, the weight attribute remains a probabilistic feature given as the probability, $\mathbf{Q}_{ij}$, of routing traffic from node i to node j. For example, adding a new node i to a traffic network not only triggers a change in the traffic intensity due to the contributions of the new node at node j it is connected to, but it also has the incidental effect of percolating through the network causing updates to the respective edge's weight along the way.

More formally, the BBV model employs a strength-driven preferential attachment model, where a new node i is connected to an already existing node j with probability

$$\prod_{i \to j} = \frac{s_j}{\sum_{n \in \mathcal{V}} s_n}, \tag{3.25}$$

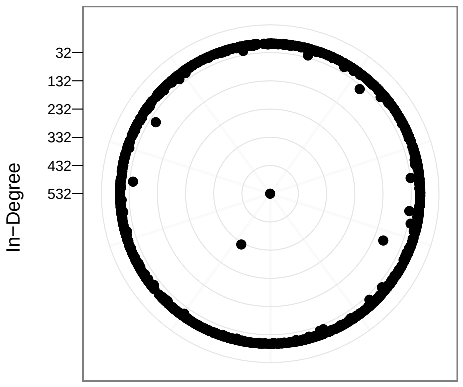
where $\mathcal{V}$ is the set of all existing nodes. The strength of a node is an attribute of its capability of servicing the incoming traffic, i.e., the exponential service rate $\boldsymbol{\mu}_i$. So, as new nodes are added to the network, the service rate of all nodes reachable from the new node are updated.

The adaptation of the model dynamics to suit queueing networks start with a single node. At each time step a new node with Poisson arrival rate λ_{0i} is created. This arrival rate could be fixed to a particular value or it could be stochastic. Here, a uniform random rate assignment with an upper bound of $\lambda_{\max}$ is used. The node is connected with $d_{\max}$ edges to already existing nodes in the network according to equation (3.25) not allowing parallel edges, where $d_{\max}$ is drawn from a uniform distribution (though other schemes are easily employed). Once the connections are established, then the new edges are assigned an equal weight of $1/\mathrm{deg}^+(i)$, where $\mathrm{deg}^+(i)$ denotes the out-degree of the new node i. Then the traffic contributions, $\boldsymbol{\lambda}_i$, of the new node i are "induced" using two rules

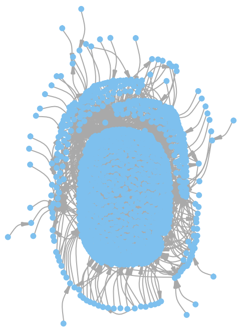
$$\mu_{i0} = \lambda_{0i}\delta_V \tag{3.26}$$

$$\boldsymbol{\mu}_j = \mu_{j0} + \sum_{i \exists (i,j)} \boldsymbol{\lambda}_i \mathbf{Q}_{ij} \delta_E. \tag{3.27}$$

Equation (3.26) assigns a service rate to the new node using a constant factor δ_V. If $\delta_V > 1$, then the new node is stable, i.e., $\boldsymbol{\lambda}_i < \boldsymbol{\mu}_i$ (equation (3.3)). Equation (3.27) determines to what degree the traffic stimulates the occurrence of variations through the network emanating from the newly created node. $\boldsymbol{\lambda}_i * \mathbf{Q}_{ij}$ determines the amount of traffic being routed from node i to node j. If $\delta_E \geqslant 1$, then the stability condition is maintained. The second rule is applied iteratively in a breadth-first-search manner. The role of both δ_V and δ_E is to adjust the utilisation rate at the new node being connected and the already existing nodes in the network respectively. Consequently, by generating networks with different values for δ_V and δ_E it is possible to examine and analyse the performance metrics of the queueing network with respect to the utilisation rate at each node. A radial plot of the graph is presented in Figure 3.3(a). It shows that the graph is defined in its periphery with many nodes having only a small in-degree, while very few nodes have a high in-degree. Figure 3.3(b) presents the actual graph structure layout using Kamada-Kawai algorithm [77]. This representation clearly shows that certain vertices in the periphery of the network with no or small in-degree, while the ones in the centre of the network show a large in-degree. The vertices in the centre are the ones that are added early in the evolutionary process, so the network exhibits a rich-get-richer semantics, which is expected from preferential attachment growth models.



(a) Radial Plot of the Node In-Degree



(b) Kamada-Kawai Layout

Figure 3.3: BBV model

Figure 3.4 shows the corresponding in-degree distribution, which follows a power-law as $P(k) \sim k^{-\lambda}$ with $\lambda = 2.16$. Power-laws have become a focus of attention when analysing probability distributions. The principle reason for this is that power-laws characterise a staggering number of natural phenomena exhibiting quantities that vary over many orders of magnitudes, such as city population [111] or frequency of word usage in many languages [188]. In many such natural phenomena the exponent of the distribution is close to -2. That is the quantity of interest with popularity P approximately scales as P^{-2}, which is the case for the in-degree distribution shown in Figure 3.4 as well. Power-laws are also called Zipf's law according to his pioneering work of unravelling the semi-universal quadratic scaling law for word frequencies and city populations [188].

Such networks are called scale-free and exhibit the important characteristic of a few nodes with an in-degree that exceeds the average in-degree significantly. Those nodes are often referred to as hubs and obtain a special role in real-world network, such as the Internet. In other words, these networks exhibit a long tail, which is the part of the distribution representing large but rare events. Fitting power-laws in this thesis is based on the statistical framework of Clauset et al. [34].

3.2.3 Random Model

The most commonly studied random graph model is the one developed by Erdős and Rényi (ER), denoted $G(N, p)$, where a graph consists of N nodes and every possible edge is established with

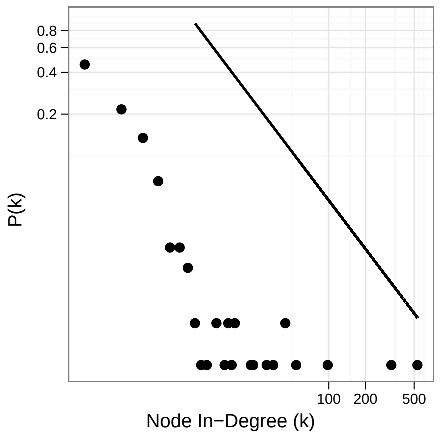
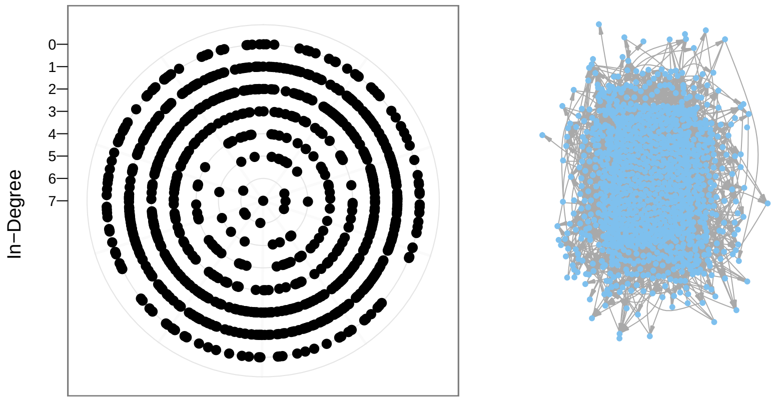


Figure 3.4: Power-Law of the Node In-Degree - BBV model

probability p. Within the context of this thesis, this model's purpose serves as a useful comparison to the BBV model, since it does not exhibit scale-free behaviours of node and edge properties.

Since the inception of the random graph model by Erdős and Rényi, this model and related models have mainly been used in theoretical analysis of asymptotic behaviours, such as the probability of connectedness as the size of the network grows very large. Our goal, however, is to compare the sensitivity of mean total waiting time versus the utilisation in social graphs. Hence the ER model needs to be adapted in a similar fashion as the BBV one to frame it into a stable Open Queueing Network. To achieve this task, the closely related model $G(N, M)$ is used, where M represents the number of edges to be established. As a result, graphs can be generated with comparable high-level features as in the social model, i.e., number of edges and number of nodes.



(a) Radial Plot of the Node In-Degree (b) Kamada-Kawai Layout

Figure 3.5: ER model

Once the ER graph is created all cycles need to be removed and the Poisson arrival and Exponential service rates need to be fixed. This is achieved by sorting $\mathcal{G}$ topological and applying equations (3.26) and (3.27) in reverse topological order at each step. The result of this model can be examined in Figure 3.5(a) and Figure 3.5(b). In contrast to the previous model, the ER evolution model has a much smaller range of in-degree values in the graph and the distribution of those is much more homogeneous. Additionally, the Kamada-Kawai layout algorithm is not able to discern structural differences between groups of vertices.

3.3 Evaluation

3.3.1 Methodology

The network evolution models are controlled systematically within a response surface modelling framework called kriging presented in Chapter 2. The input domain for the parameters that control the stability criterion within the evolution models are $\Omega = (\delta_v, \delta_E) \subseteq \mathbb{R}^2$. The graph size is fixed to 1000 nodes, with a maximum out-degree $d_{max} = 3$ for the social model, which will create approximately 2000 edges. Using the ER model $G(N, M)$, M is fixed to 2000 to provide comparable measurements. A Latin Hypercube Sampling design of experiments is employed to sample the input domain, Ω, with an initial sample size of 15 including the boundary values. At each sampled location, $x_i = (\delta_{v,i}, \delta_{E,i})$, a number of networks are evolved with different random number seeds and their mean total waiting time, W, and mean utilisation, U, are evaluated. A metamodel is then fitted with the expected values of those performance metrics. The model quality is assessed with the $R^2_{\text{prediction}}$ statistic, because the sum of squared errors of the standard R^2 statistic is typically not available in kriging models. $R^2_{\text{prediction}}$ is a normalised quantity that ranges usually from 0 to 1 with 1 indicating a perfect fit. If $R^2_{\text{prediction}} < 0.95$ then a validation set of scattered locations over the domain is sampled and their mean-squared error, MSE, evaluated. The location with the highest mean-squared error is chosen as the new location where the BBV or ER networks are evolved and added to the data set. Fitting another response surface, this procedure repeats until $R^2_{\text{prediction}}$ shows the desired level of accuracy.

As a result, a response surface for the mean total waiting time, W, and mean utilisation, U, for both BBV and ER models will be generated, which allows a qualitative analysis.

3.3.2 Structural Properties

This section evaluates the structural properties of both network models using ten replicas for each evolution model generated with different random number seeds. First the unweighted topological properties are presented in Table 3.1. [1]

Table 3.1: Summary of the unweighted Structural Properties

	BBV Model		ER Model	
	Mean	Std. Dev.	Mean	Std. Dev.
Diameter	6.4	1.07	58.9	7.06
Average Path Length	1.7	0.12	14.55	1.0
Cluster Coefficient	0.0078	0.0008	0.0026	0.00081

[1]Calculated with: Csardi G, Nepusz T: The igraph software package for complex network research, InterJournal, Complex Systems 1695. 2006. Version 0.52. Retrieved on 22.09.2009 from http://igraph.sf.net

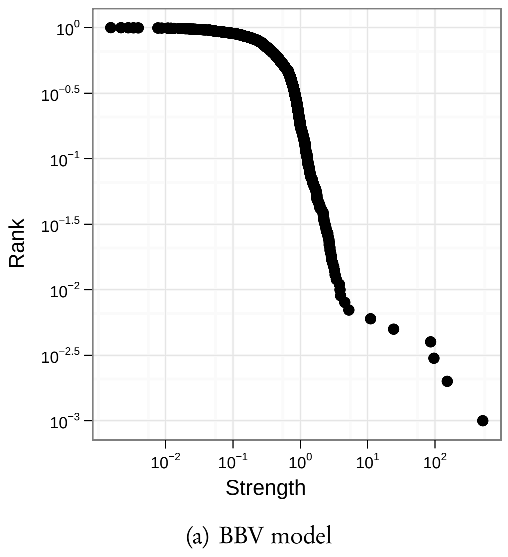
As expected the BBV model exhibits some small-world properties of a low diameter and a low average path length. Also, the clustering coefficient is higher compared to the ER model.

Importantly, the weighted structural properties presented in Table 3.2 integrate the weights of both evolved networks. [2] The overall results obtained this way for the BBV model mirror the ones presented in Table 3.1, while the ones for the ER model show a smaller diameter and average path length.

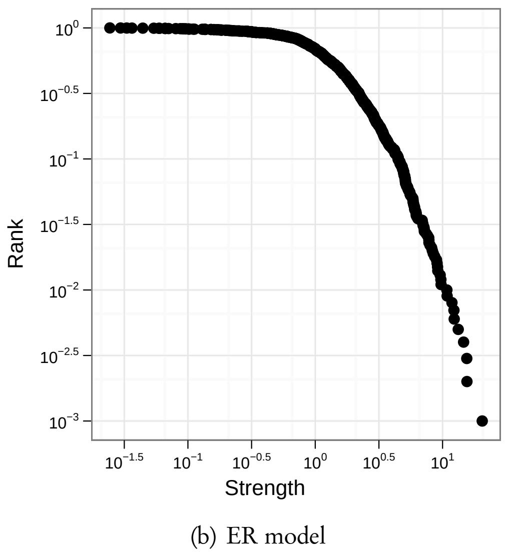
Table 3.2: Summary of the weighted Structural Properties

	BBV Model		ER Model	
	Mean	Std. Dev.	Mean	Std. Dev.
Diameter	6.59	0.34	11.68	0.433
Average Path Length	1.86	0.1	3.31	0.12
Cluster Coefficient	0.34	0.04	0.0012	0.0006

In the remainder of this section power-law relationships between the different topological quantities presented in Section 3.2.1 are shown. Evidently, power-law relationships have been discovered in many social and technical networks. Typically, the quantity under study ranges on a large spectrum of values, having a heavy-tailed distribution. Figure 3.6 gives the distribution of inbound strength for each node in the network. In Figure 3.6(a) the Power-law was found to be $P(s_{in}) \sim s_{in}^{\gamma_{in}}$ with $\gamma_{in} = 2.59$ and a Kolmogorov-Smirnov goodness of fit of 0.095. For the random model the Power-law exponent is $\gamma_{in} = 3.86$ (0.044) and the strength is an order of a magnitude lower compared to the BBV model.



(a) BBV model



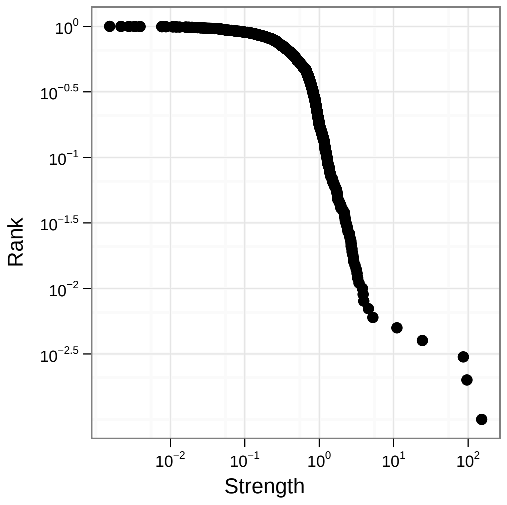
(b) ER model

Figure 3.6: In-strength Distribution

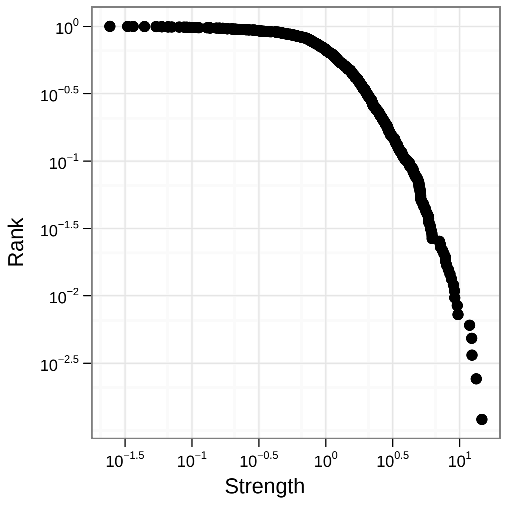
The distribution of the out-strengths presented in Figure 3.7 show a similar quality as before. In queueing networks modelled in this thesis no in-coming transactions are dropped, meaning that all

[2]Calculated with: Opsahl, T. (2007). tnet: Software for Analysis of Weighted and Longitudinal networks, version 0.1.1. Retrieved on 10.09.2009 from http://opsahl.co.uk/tnet/

transactions are processed. Only the leaf nodes in the network finalise a transaction and therefore do not have any out-strength. The Power-law exponents for the BBV model and the ER model respectively, are $\gamma_{in} = 2.72$ (0.07) and $\gamma_{in} = 2.81$ (0.051).



(a) BBV model



(b) ER model

Figure 3.7: Out-strength Distribution

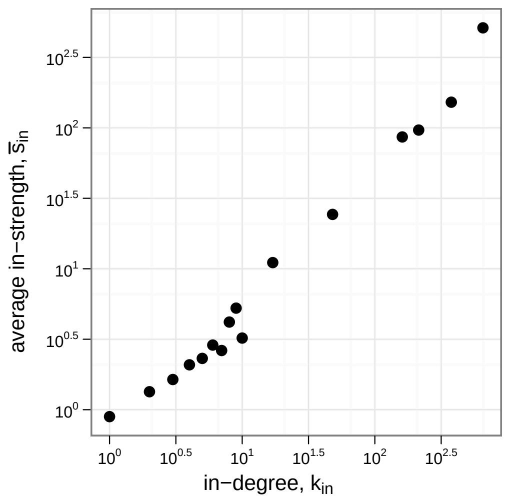
Interestingly, analysing the intensity of the in-bound and out-bound transactions in Figure 3.8 presents some distinguished features. For both, the BBV model and the ER model there is a reverse relationship between the in-strength to in-degree versus the out-strength to out-degree. This can be explained by the absence of re-wiring and loops in the network. The network models employ a rich get richer semantics. So, nodes that entered the network early have only a low out-degree, but gain in in-degree over time. Nodes that are added late in the evolution process are likely to have a higher out-degree, but only a very small strength.

Finally, the in-degree versus out-degree correlations is investigated and presented in Figure 3.9. Interestingly, for both network evolution models there is a marked assortativity mixing for the weighted metric (equation (3.21)) and a disassortative mixing in the unweighted case (equation (3.20)). This means that edges with stronger weights are directed towards neighbours with larger degrees while nodes with large in-degrees have mainly neighbours with lower degrees.

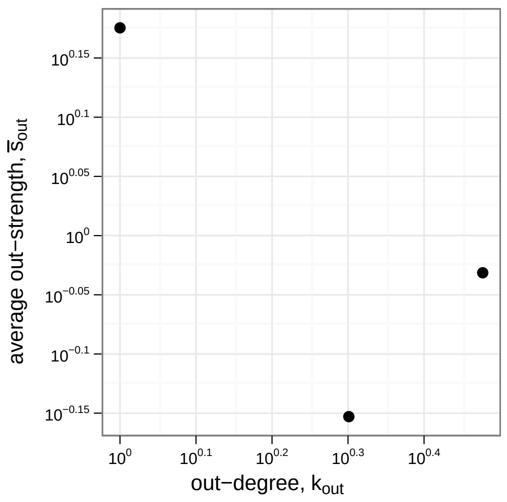
3.3.3 Sensitivity Analysis

Analysing the sensitivities of mean total waiting time with respect to the system utilisation provides valuable insights in how a system operates under load. The experiments for the sensitivity analysis were conducted with two degrees of freedom, δ_v and δ_E, where the generated networks were replicated with different random number seeds to produce ten instances of each design point in the domain $\Omega = (\delta_v, \delta_E) \in (1, 1.1]^2$. The mean value of the respective queueing performance metric was then used to perform the sensitivity analysis.

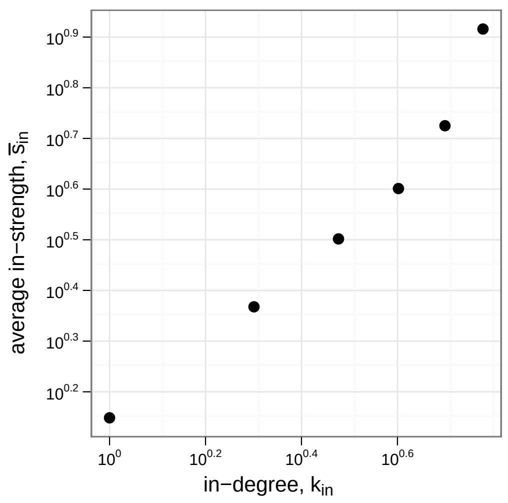
For the ER model this domain shows that the waiting time spikes in a small region of the input space and then levels off very quickly in both directions. The response surface of the utilisation represents a perfect plane as shown in Figure 3.10(a).



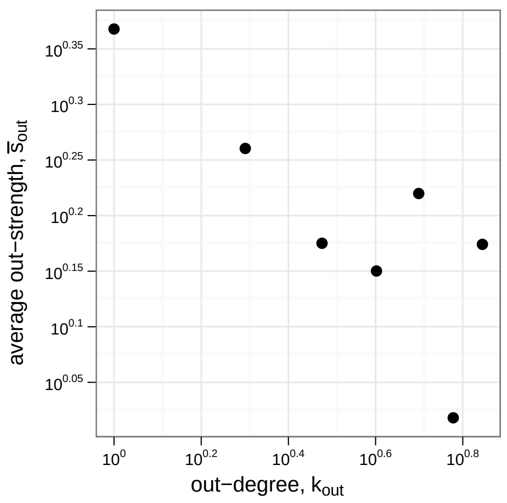
(a) In-degree versus in-strength for the BBV model



(b) Out-degree versus out-strength for the BBV model



(c) In-degree versus in-strength for the ER model

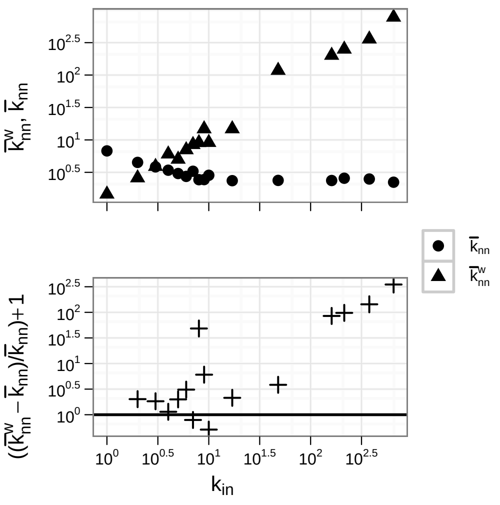


(d) Out-degree versus out-strength for the ER model

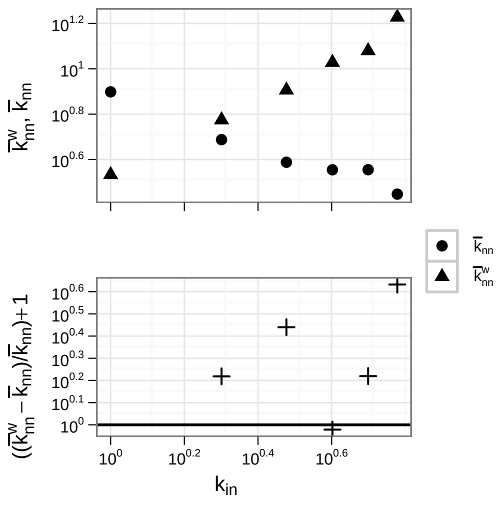
Figure 3.8: Structural Relationships

In contrast to the ER model, Figure 3.10(b) shows that the mean total waiting time for the BBV model levels off in the δ_V input dimension, while the δ_E parameter does not present a significant contribution to the waiting time behaviour. Interpreting this result in light of the underlying structure provides an explanation of this behaviour. The density of the BBV model is concentrated in the periphery of the network (see Figure 3.5(a)) and the average path length is very low. 68% of the nodes in the graph have an in-degree of 0 (83% with in-degree of 1 or less). Therefore, the parameter δ_E has little impact on a change in utilisation, because boosting the edge intensity only accounts for few paths towards the centre of the network. This is unlike the ER model, where 14% of the nodes have an in-degree of 0 (49% with an in-degree of 1 or less).

Conducting a canonical analysis of the response surface in the small domain $[1.01, 1.09]$ further elucidates the magnitude of the parameters relative contribution to the response (for a detailed treatment of canonical analysis see [107] and Section 2.5). Further, the analysis gives eigenvalues

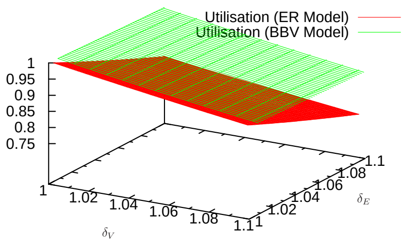


(a) BBV model

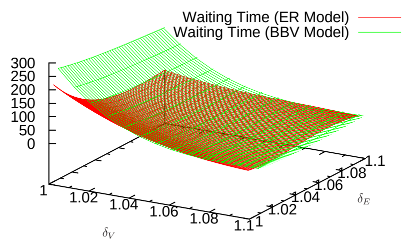


(b) ER model

Figure 3.9: In-degree - out-degree correlations



(a) System Utilisation for Both Models



(b) Total Mean Waiting Times for Both Models

Figure 3.10: Queueing Performance for Both Models

(summarised in Table 3.3) of the second-order response surface with which the nature of the surface can be characterised.

The eigenvalues of the second-order response surface λ_1 and λ_2 correspond to the δ_V and the δ_E parameters respectively. Since the stationary point is outside of the design domain and both eigenvalues are positive, it suggests a rising ridge system with a maximum stationary point, which is easily confirmed visually by consulting Figure 3.10(b). Comparing the nature of the response surfaces, one can see that the relative importance of the parameters to the mean total waiting time in both models expose different scales. While δ_V is the most dominant parameter in the ER model with a factor of two higher than δ_E, the δ_v parameter in the BBV model dominates δ_E by almost an order of a magnitude. δ_E in the BBV model shows little influence on the waiting time (see Table 3.3 for the respective eigenvalues).

Table 3.3: Summary of the Canonical Analysis

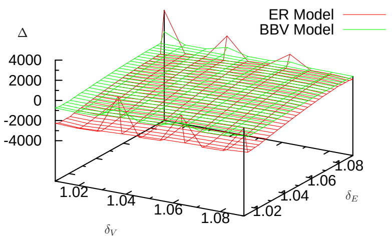
	BBV Model	ER Model
λ_1	4602	13241
λ_2	35917	24232

The interaction of waiting time and utility is important to understand, because the behaviour of this function is only linear up to a certain point. For $\mathrm{U} \gtrapprox 80\%$ both models exhibit a different non-linear quality of the curve, which is investigated taking the first and second derivative of W w.r.t. U, i.e.,

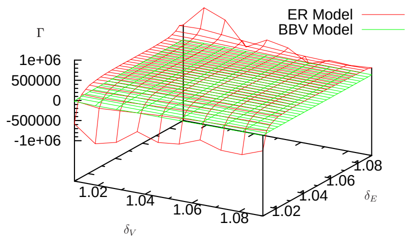
$$\Delta = \frac{\mathrm{d}W}{\mathrm{dU}} \tag{3.28}$$

$$\Gamma = \frac{\mathrm{d}^2W}{\mathrm{dU}^2}. \tag{3.29}$$

Equation (3.28) gives the rate of change, while equation (3.29) represents the curvature. Both, Δ and Γ can easily be derived within the input domain using the best linear unbiased predictor of the kriging model [94].



(a) Δ for Both Models



(b) Γ for Both Models

Figure 3.11: Sensitivity Surfaces of the Network Models

The BBV model in both Figure 3.11(a) and Figure 3.11(b) is less sensitive to changes in the system utilisation. Γ, the curvature of the waiting time with respect to the utilisation is almost flat, and Δ exhibits a significant lower rate of change compared to the ER model.

Understanding the different performance characteristics for both models would be a benefit to communication or traffic studies that rely on network structures. The two proposed models can easily be plugged into such studies to provide a more comprehensive analysis with respect to the underlying topological features. This is particularly useful where real-world network instances are unavailable.

3.4 Conclusion and Future Work

This chapter has shown how network evolution models can be cast into queueing-theoretic systems known from operations research. Focusing on stable Jackson networks, it was demonstrated that two parameters that boost the capacity at vertices and along arcs respectively, give rise to different queueing behaviours for the different network evolution models. In particular, the evaluation concentrated on the mean total waiting time with respect to the mean utilisation of the network. Interestingly, the rate of change and the curvature, i.e., the first and second derivative of this function respectively, indicate that the presented adaptation of the scale-free BBV model is less sensitive to changes in utilisation than the ER model. Additionally, the curvature, Γ, of the BBV model within the critical regime of $U \gtrapprox 80\%$ is less pronounced. This result can be put into perspective of the adapted model and its two tuning parameters, which have a direct impact on the utilisation. The δ_E parameter does not contribute much to a difference in queueing characteristics in the BBV model, because the density of the network is concentrated in its periphery. This means that upon preferential attaching new nodes to already existing ones, a relatively short path to the leaf nodes exists. This property is one of the features of small-world networks. In contrast the homogeneous nature of the ER model implies that both parameters have an approximately equal contribution to queueing performance characteristics.

Both models are basic in that the node strength evolution is only triggered when new nodes enter the network. In order to further improve analysis, future work requires the dynamic evolution of existing nodes through a re-wiring mechanism. Coupled with a continuous evolution of the network structure, the external arrival rate could be a time-varying quantity as well, which would yield more realistic networks. For example, the process that generates the external arrival rates can take into account a time-varying non-homogeneous Poisson processes [125].

— *Experience is simply the name we give our mistakes.*

Oscar Wilde

4 Fundamentals of Reinforcement Learning

The exposition of the fundamentals of reinforcement learning (RL) in this chapter starts with the formulation of Sutton and Barto's book [145]. It details the theoretical foundations of reinforcement learning and shows under which assumptions they hold. Lifting the reinforcement learning algorithms into a multi-agent learning setting introduces challenges, in particular because multi-agent reinforcement learning does not abide to the theoretical guarantees given in the single-agent case. So Section 4.4 presents some recent developments in this setting that synthesise game-theory and reinforcement learning, which places an emphasis on the interaction patterns of multiple agents learning concurrently. This thesis uses some of the methods introduced in this chapter to model sequential distributed task assignment problems.

One of the central issues of learning concerns how choices are made in the face of alternatives. Through repeated exposure to the same or similar situation, knowledge is acquired and the process of selecting an action with this information can results in learning complex skills or behaviours. Without an explicit teacher, learning agents repeatedly select actions and observe information about the cause and effect of their actions. For this thesis the definition of an agent by Russell and Norvig is adopted, in that agents are entities that operate under autonomous control, perceive the environment, and adapt to change [128]. Agents are called rational, if they act to achieve the best expected output in an uncertain environment. The ability to reason about an uncertain environment provides important steps towards achieving a goal. It is a remarkable result that combining basic principles of learning into a computational approach arrives at achieving complex tasks.

One such learning approach is called reinforcement learning, which focuses on goal-directed learning from interactions with an environment without the need of an explicit teacher. As the name suggests, this method is based on the idea that well-performing actions are positively reinforced, while badly performing actions are negatively reinforced [144]. The consequence of reinforcement is that the tendency to selecting actions again is affected such that an improvement in the learning

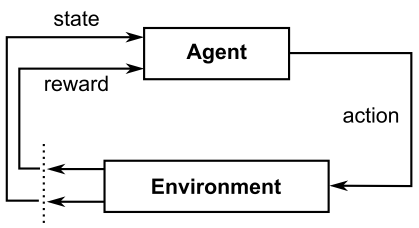


Figure 4.1: Agent-Environment Interface

objective can be attained. The formulation of reinforcement includes three aspects: sensation, action, and goal. At each time step the agent receives a state signal (sensation) from the environment and chooses an action to perform. The quality of the action is encoded in a reward signal (goal). Figure 4.1 illustrates the agent-environment interface, where the boundary between the agent and the environment reflects the limits of control rather than physical features (such as in a robot's physical body). This implies that everything out of the agent's control belongs to the environment including the computation of the reward signal. For example in a soccer playing robot, the agent senses the location of the ball and the goal as part of the environment. The motors and mechanical limbs also belong to the environment. However, the agent operates the mechanical limbs to defend a ball, run towards a goal, and kick the ball. All these complex actions generate reward signals. If the robot fails to deliver a ball to one of the players in his team a negative reward is received and the agent learns how to avoid those negative rewards by operating the limbs such that the reward signal is maximised.

So, the reinforcement learning agents learns what to do in order to maximise a numerical reward signal. By interleaving acting in an (uncertain) environment and reinforcement computation, i.e., updating the belief about the goodness of the actions, agents perform an online search in the state space. Since the agent might be unaware of how the environment reacts to actions taken, the principle issue to online search is that of exploration or trial-and-error [128]. The agent must try as many actions in as many states as possible in order to learn how to behave. The reward provides a qualitative statement (positive or negative feedback) about an action taken in the past. Action can have far-reaching impacts, not only affecting the immediate reward, but also future rewards. These two characteristics of trial-and-error and delayed reward are the two most important distinguishing features of reinforcement learning. Compared to other machine learning approaches, the reward signal is less informative than supervised learning, which provides the correct signal (see [28] for an empirical comparison of supervised learning techniques), and unsupervised learning, which does not provide an output signal at all.

To be more precise about the objective of learning, the agent seeks to maximise the expected return, R_t, where the return is defined as some specific function of the reward sequence. For episodic tasks or tasks with a final state, the sum of the rewards, $R_t = r_{t+1} + r_{t+2} + \cdots + r_T$, where T is the final time step, can be used. However, most tasks are continuous in nature, such as the distributed task assignment problem considered in this thesis. Using the sum of the rewards in such cases implies that $T = \infty$ and consequently the return could itself be infinite. To address this issue in a mathematically convenient way the concept of discounting is used. The agent then chooses actions to maximise the expected discounted return denoted as

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}, \tag{4.1}$$

where $0 \leqslant \gamma < 1$ is called the discount rate. If $\gamma = 0$, then the agent is myopic, i.e., only concerned with immediate rewards. In contrast, as γ approaches 1 the agent becomes more farsighted taking future rewards more strongly into account. Additionally, the discount rate is a mathematical trick to bound the sum of the future rewards that otherwise would grow infinitely, which would complicate the mathematical derivation of the solutions.

4.1 Markov Decision Processes

Within the framework of reinforcement learning introduced in this chapter agents make their decision as a function of the state received from the environment. Importantly, the effects of the actions are Markovian, i.e., a state signal retains all relevant information without a history of the past states, and current actions can have a delayed impact. These two facts together have significant practical value, because RL solutions can be implemented in a memory efficient way.

The state signal can be an arbitrary vector, where each vector element has a certain meaning for the agent. For example, the immediate queue length or the average delay in queue of a server in a queueing network could be candidates for state signals to the agent, whereby past signals should not be included in this vector.

Let $s_t \in \mathcal{S}$ be the state of the system at time t. At every time step the agent picks an action $a_t \in \mathcal{A}(s)$. Consider the dynamics of the environment

$$\begin{aligned} P\left\{s_{t+1} = s', r_{t+1} = r' \mid s_t, a_t, r_t, s_{t-1}, a_{t-1}, r_{t-1}, \ldots, r_1, s_0, a_0\right\} = \\ P\left\{s_{t+1} = s', r_{t+1} = r' \mid s_t, a_t\right\}. \end{aligned} \tag{4.2}$$

If equation (4.2) is true then the right-hand side represents a one-step dynamics, which enables the prediction of the next state and next expected reward given the current state and action equally as well as with the dynamics that take the complete history into account. It follows that the best policy of choosing an action as a function of a Markov state is as good as the best policy for choosing an action as a function of complete histories. Note, that a policy, denoted as $\pi(s, a)$, is a mapping from states to probabilities of selecting each possible action.

Based on the Markov property, Markov decision processes (MDPs) provide a mathematical framework for modelling the decision-making in reinforcement learning problems.

Definition 4.1 (MDP). *An MDP [73] can be defined as a tuple* $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$, *where*

- $\mathcal{S}$ *is a finite set of states;*
- $\mathcal{A}$ *is a finite set of actions;*
- $\mathcal{P}^a_{ss'} = P\{s_{t+1} = s' \mid s_t = s, a_t = a\}$ *is the transition probability given a current state* s *and taking action* a *to state* s'*;*

- $\mathcal{R}^{a}_{ss'} = \mathbb{E}\{r_{t+1} \mid s_t = s, a_t = a, s_{t+1} = s'\}$ *is the expected value of the next reward taking action* a *in state* s *and transitioning to the current state* s'.

4.2 Value Functions

Given the return functions (total sum or discounted sum) to be maximised an algorithm needs to be specified to find the policy with the maximum return. A naive approach would be to sample returns for each possible policy and then choosing the policy with the largest expected return. This becomes prohibitively expensive, because the policy space can be extremely large. Instead, most reinforcement learning algorithms employ value functions to estimate the goodness to be in given states or given actions. The notion of how good the estimate is is expressed in terms of expected returns which depends on the actions an agent is going to take. Consequently, value functions are defined with respect to particular policies, i.e., the value of a state s (and taking action a) under a policy π, denoted as $V^{\pi}(s)$ (or $Q^{\pi}(s,a)$), is the expected return when starting in s (, taking action a,) and following the policy π thereafter. Formally, the value functions for MDPs can be defined as

$$V^{\pi}(s) = \mathbb{E}_{\pi}\{R_t \mid s_t = s\} = \mathbb{E}_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s\right\}, \tag{4.3}$$

$$\begin{aligned} Q^{\pi}(s,a) &= \mathbb{E}_{\pi}\{R_t \mid s_t = s, a_t = a\} \\ &= \mathbb{E}_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s, a_t = a\right\}. \end{aligned} \tag{4.4}$$

The value functions given in equations (4.3) and (4.4) can be estimated incrementally while the agent is interacting with the environment. For example, the agent can maintain averages while following policy π for each encountered state of the actual returns that followed those states. As the states are visited infinity often, the average will converge to the state's value, $V^{\pi}(s)$. Conversely, keeping averages for each action taken in a state, then these averages will converge to the action values, $Q^{\pi}(s,a)$. The technique of keeping averages over all states (and actions) is called Monte Carlo estimation. However, these techniques become prohibitively expensive for large state (and action) spaces and therefore a more compact representation of the value functions is sought. This can be achieved by discretising the state space (assuming finite actions) and representing the function in tabular form. Or otherwise parameterised function approximators, such as neural networks, can be employed.

A fundamental property of value functions is a recursive relationship between the value of a state and the values of its successor states. This relationship is captured in the Bellman equation for V^{π}

$$\begin{aligned}
V^{\pi}(s) &= \mathbb{E}_{\pi}\{R_t \mid s_t = s\} \\
&= \mathbb{E}_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s\right\} \\
&= \mathbb{E}_{\pi}\left\{r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} \mid s_t = s\right\} \\
&= \sum_{a} \pi(s,a) \sum_{s'} \mathcal{P}^{a}_{ss'} \left[\mathcal{R}^{a}_{ss'} + \gamma \mathbb{E}_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+2} \mid s_{t+1} = s'\right\}\right] \\
&= \sum_{a} \pi(s,a) \sum_{s'} \mathcal{P}^{a}_{ss'} \left[\mathcal{R}^{a}_{ss'} + \gamma V^{\pi}(s')\right], \qquad (4.5)
\end{aligned}$$

where actions are taken from the set $\mathcal{A}(s)$ and the next states s' are taken from the set $\mathcal{S}$. The Bellman equation can be understood as looking ahead at all possible successor states given the available actions and averages over all those possibilities, weighting each by its probability of occurring. It captures the fact that an agent's reward depends not only on its immediate reward but also on its future (discounted) rewards.

So the aim of reinforcement learning is to find the optimal policy that maximises the reward over the long run. For finite MDPs, a policy π can be defined to be better than or equal to a policy π' if its expected return is greater than or equal to the one of π' for all states, or more formally $V^{\pi}(s) \geqslant V^{\pi'}(s)$ for all $s \in \mathcal{S}$. The optimal policy that is better or equal to all other policies is considered to be the optimal policy π^*. Thus, the optimal state-value function, denoted V^*, is defined as

$$V^*(s) = \max_{\pi} V^{\pi}(s) \;\forall\; s \in \mathcal{S}. \qquad (4.6)$$

Similarly, the optimal action-value function can be defined as follows

$$Q^*(s,a) = \max_{\pi} Q^{\pi}(s,a) \;\forall\; s \in \mathcal{S} \text{ and } a \in \mathcal{A}(s). \qquad (4.7)$$

Because V^* is a value function, the Bellman equation (4.5) must hold and therefore the optimal state-value function can be rewritten without a reference to the optimal policy. This follows intuitively, because the value of a state under an optimal policy must equal the expected return for the best action from that state. This expression is called the Bellman optimality equation, which is defined as

$$\begin{aligned}
V^*(s) &= \max_{a \in \mathcal{A}(s)} Q^{\pi^*}(s,a) \\
&= \max_a \mathbb{E}_{\pi^*} \{R_t \mid s_t = s, a_t = a\} \\
&= \max_a \mathbb{E}_{\pi^*} \left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s, a_t = a \right\} \\
&= \max_a \mathbb{E}_{\pi^*} \left\{ r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} \mid s_t = s, a_t = a \right\} \\
&= \max_a \mathbb{E}_{\pi^*} \{ r_{t+1} + \gamma V^*(s_{t+1}) \mid s_t = s, a_t = a \} \\
&= \max_a \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V^*(s') \right] . \qquad (4.8)
\end{aligned}$$

The Bellman optimality equation for Q^* is

$$\begin{aligned}
Q^*(s,a) &= \mathbb{E} \left\{ r_{t+1} + \gamma \max_{a'} Q^*(s_{t+1}, a') \mid s_t = s, a_t = a \right\} \\
&= \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma \max_{a'} Q^*(s', a') \right] . \qquad (4.9)
\end{aligned}$$

A unique solution of the Bellman equation (4.8) independent of the policy employed can be obtained for finite MDPs. The Bellman equation can be turned into a system of N linear equations with N unknowns given N states. If the transition probabilities, $\mathcal{P}^a_{ss'}$, and the reward function, $\mathcal{R}^a_{ss'}$, are given, then the equations of the system can be solved using any standard dynamic programming method. Once the V^* is obtained, at least one action in any state exists for which a maximum of the Bellman equation is attained. With this insight any greedy policy that assigns non-zero probabilities only to these actions with respect to the optimal value function is an optimal policy. Interestingly, with optimal one-step actions the expected long-term return is maximised, because V^* encodes for each current state all possible future rewards. In contrast to this, the agent does not need to perform a one-step ahead search for Q^*, because the state-action pairs are locally and immediately available. Consequently, the agent can choose optimal actions without knowing the possible successor states. This implies that no knowledge about the dynamics of the environment, i.e., the transition probabilities, $\mathcal{P}^a_{ss'}$, and the reward function, $\mathcal{R}^a_{ss'}$, is required.

Solving the Bellman optimality equations (4.8) and (4.9) explicitly is akin to performing an exhaustive search starting from the current state, enumerating all possible paths (choosing an action a and ending up in state s'), computing their probability of occurrence, and computing their desirability in terms of expected reward. This approach relies on three assumptions which are rarely true in practice:

1. the dynamics of the MDP are known exactly, i.e., the transition probabilities, $\mathcal{P}^a_{ss'}$, and the reward function, $\mathcal{R}^a_{ss'}$;

2. enough resources to compute the solution;

3. and the Markov property.

As a consequence, reinforcement learning uses approximations to estimate the dynamics of the environment from actual experience. This implies that a reinforcement learner builds a predictive model of the environment through taking random actions. The agent is able to perceive the state of the environment and receives stimuli that indicate how well (or badly) this action performed. Without such feedback on its actions, reinforcement learning agents will have no grounds for which actions to take in the future. Over time, trial-and-error actions coupled with their respective feedback will lead to an optimal, or near optimal policy for the environment. The elementary solution methods that are the foundations of reinforcement learning are presented in the next section.

4.3 Elementary Solution Methods

4.3.1 Dynamic Programming

Dynamic Programming (DP) combines a set of methods to solve for optimal policies given a perfect model of the environment as a finite Markov Decision Process. While their use in reinforcement learning have limited value, they are important theoretically.

The first method considered here is called policy evaluation, which computes the state-value function V^π for an arbitrary policy π. Given the exact specification of the environment equation (4.5) can be solved using a system of $|\mathcal{S}|$ linear equations in $|\mathcal{S}|$ unknowns. The iterative policy evaluation method uses the Bellman update rule

$$\begin{aligned} V_{k+1}(s) &= \mathbb{E}_\pi\{r_{t+1} + \gamma V_k(s_{t+1}) \mid s_t = s\} \\ &= \sum_a \pi(s,a) \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V_k(s')\right] \;\; \forall\, s \in \mathcal{S} \end{aligned} \tag{4.10}$$

to obtain successive approximations for V^π (4.5) starting from an initial guess V_0 and is guaranteed to converge to V^π as $k \to \infty$. The successive approximations of V_{k+1} from V_k are produced by iterating through all states and replacing the old value of state s with a new estimate. This estimate is calculated from the old values of the successor states of s, and the expected immediate return, along with all the one-step transitions possible under the policy being evaluated. This operation is referred to as performing a full backup, which means that every state is backed up once to produce a new estimate V_k per iteration. Algorithm 4.1 details the iterative policy evaluation method, which updates the values "in-place", that is values are used in subsequent calculations as soon as new values become available. It has been shown that this algorithm converges faster than waiting for the sweep through the state space to be finished and then updating the new values. However, the order in which states are visited has a significant impact on the rate of convergence.

An important aspect of this algorithm is the stopping criterion after each sweep through the state space, usually chosen to be $\max_{s\in\mathcal{S}} |V_{k+1}(s) - V_k(s)| < \varepsilon$, where ε is set to a sufficiently small value.

Algorithm 4.1: Iterative Policy Evaluation

```
Input: $\pi$, the policy to be evaluated
$V(s) = 0, \ \forall\ s \in \mathcal{S}$
repeat
    $\Delta \leftarrow 0$
    foreach $s \in \mathcal{S}$ do
        $\upsilon \leftarrow V(s)$
        $V(s) \leftarrow \sum_a \pi(s,a) \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V(s')\right]$
        $\Delta \leftarrow \max(\Delta, |\upsilon - V(s)|)$
    end
until $\Delta < \varepsilon$ (a small positive number)
Output: $V \approx V^\pi$
```

The policy iteration algorithm considered the policy fixed and finds the respective value function. Once the value function V^π is known the underlying policy can be improved by selecting an action $a \neq \pi(s)$ for some state s and following the policy thereafter. The value of this behaviour is given as

$$\begin{aligned} Q^\pi(s,a) &= \mathbb{E}_\pi\{r_{t+1} + \gamma V^\pi(s_{t+1}) \mid s_t = s, a_t = a\} \\ &= \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V^\pi(s')\right]. \end{aligned} \tag{4.11}$$

Theorem 4.1 (Policy Improvement Theorem). *Let π and π' be any pair of deterministic policies such that, for all $s \in \mathcal{S}$,*

$$Q^\pi(s, \pi'(s)) \geqslant V^\pi(s). \tag{4.12}$$

Then the policy π' must be as good as, or better than, π, i.e.,

$$V^{\pi'}(s) \geqslant V^\pi(s)\ \forall\ s \in \mathcal{S}. \tag{4.13}$$

The following proof expands the Q^π side of the equation (4.12) recursively using equation (4.11) and applying equation (4.12) until $V^\pi(s) \leqslant V^{\pi'}(s)$.

Proof.

$$
\begin{aligned}
V^{\pi}(s) &\leqslant Q^{\pi}(s,\pi'(s)) \\
&= \mathbb{E}_{\pi'}\{r_{t+1} + \gamma V^{\pi}(s_{t+1}) \mid s_t = s\} \\
&\leqslant \mathbb{E}_{\pi'}\left\{r_{t+1} + \gamma Q^{\pi}(s_{t+1},\pi'(s_{t+1})) \mid s_t = s\right\} \\
&= \mathbb{E}_{\pi'}\{r_{t+1} + \gamma \mathbb{E}_{\pi'}\{r_{t+2} + \gamma V^{\pi}(s_{t+2})\} \mid s_t = s\} \\
&= \mathbb{E}_{\pi'}\left\{r_{t+1} + \gamma r_{t+2} + \gamma^2 V^{\pi}(s_{t+2}) \mid s_t = s\right\} \\
&\leqslant \mathbb{E}_{\pi'}\left\{r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \gamma^3 V^{\pi}(s_{t+3}) \mid s_t = s\right\} \\
&\vdots \\
&\leqslant \mathbb{E}_{\pi'}\left\{r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \gamma^3 r_{t+4} + \cdots \mid s_t = s\right\} \\
&= V^{\pi'} \qquad (4.14)
\end{aligned}
$$

□

Extending this to change all states and all possible actions is called policy improvement. Policy improvement is achieved by setting up a new policy that improves on an original one, and letting it be greedy (or nearly greedy) with respect to the value function of the original policy. Consider the new greedy policy π', given by

$$
\begin{aligned}
\pi'(s) &= \arg\max_a Q^{\pi}(s,a) \\
&= \arg\max_a \mathbb{E}\{r_{t+1} + \gamma V^{\pi}(s_{t+1}) \mid s_t = s, a_t = a\} \\
&= \arg\max_a \sum_{s'} \mathcal{P}^a_{ss'}\left[\mathcal{R}^a_{ss'} + \gamma V^{\pi}(s')\right]. \qquad (4.15)
\end{aligned}
$$

By construction, the policy improvement theorem (Theorem 4.1) holds for equation (4.15). Suppose the new greedy policy, π', is not better, but as good as the original one, then $V^{\pi'}(s) = V^{\pi}(s)$ (from equation (4.13)) and the value function for $V^{\pi'}$ is equal to the Bellman optimality condition in equation (4.8). This implies that π and π' must be optimal policies. While this line of argument followed the derivation of the formulas for the deterministic policy, it is generally applicable to stochastic policies as well.

Algorithm 4.2: Policy Improvement

```
Input: $V^{\pi}$, value function for an arbitrary deterministic policy
foreach $s \in \mathcal{S}$ do
    $\pi(s) \leftarrow \arg\max_a \sum_{s'} \mathcal{P}^a_{ss'}\left[\mathcal{R}^a_{ss'} + \gamma V(s')\right]$
end
Output: $\pi' \geqslant \pi$
```

Policy iteration combines the two approaches mentioned above, i.e., policy evaluation (Algorithm 4.1) and policy improvement (Algorithm 4.2), into an iterative scheme to obtain a sequence of monotonically improving policies and value functions.

Policy iteration, however, is prohibitively expensive, because the policy evaluation only converges in the limit to V^π and has to be performed at each iteration. Truncating the policy evaluation after just one sweep of the state space is a special case and the algorithm is called value iteration. A simple backup operation combining the policy improvement and truncating the policy evaluation is given as

$$\begin{aligned} V_{k+1}(s) &= \max_a \mathbb{E}\{r_{t+1} + \gamma V_k(s_{t+1}) \mid s_t = s, a_t = a\} \\ &= \max_a \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V_k(s')\right] \ \forall\, s \in \mathcal{S}. \end{aligned} \tag{4.16}$$

Equation (4.16) is a backup rule for the Bellman optimality equation (4.8) and therefore is guaranteed to converge to V^* for arbitrary starting values V_0. Equation (4.16) is identical to the policy evaluation backup equation (4.11) except that it requires the max operator taken over all actions.

Algorithm 4.3 presents the algorithm of value iteration with the same stopping criterion as Algorithm 4.2 to avoid an infinite number to converge exactly to V^*.

Algorithm 4.3: Value Iteration

```
$V(s) = 0,\ \forall\, s \in \mathcal{S}$
**repeat**
    $\Delta \leftarrow 0$
    **foreach** $s \in \mathcal{S}$ **do**
        $v \leftarrow V(s)$
        $V(s) \leftarrow \max_a \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V(s')\right]$
        $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
    **end**
**until** $\Delta < \theta$ *(a small positive number)*
**Output**: $\pi(s) = \arg\max_a \sum_{s'} \mathcal{P}_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V(s')\right]$
```

Both algorithms, policy iteration and value iteration converge to the optimal policy for discounted finite MDPs. Also, they exhibit a pattern that describes almost all reinforcement learning methods. This pattern consists of two simultaneous, interacting processes, one to evaluate the value function with the current policy, and the other to turn the policy into a greedy one with respect to the current value function. This pattern is called generalised policy iteration and is abstract enough to cover different schemes of granularity. For example in policy iteration these two processes alternate, each completing before the other one begins. In contrast, value iteration, only a single iteration of the policy evaluation is performed before performing policy improvement. This iteration process eventually stabilises, meaning that no further improvement of the current policy and the current value function are attained and also that optimal policies and optimal value functions share the same optimal solution.

Compared to direct search methods in the policy space dynamic programming is exponentially faster providing provable polynomial worst-case guarantees as a function of the states and actions.

For finite MDPs solutions can be found in polynomial time even though the number of deterministic policies is $|\mathcal{A}|^{|\mathcal{S}|}$. However, dynamic programming suffers from the curse of dimensionality, the fact that the number of states grows exponentially with the number of state variables, in the same way as other solution methods.

4.3.2 Monte Carlo Methods

Dynamic programming requires complete knowledge of the dynamics of the environment and the reward function which makes those techniques impractical in many real scenarios. Monte Carlo techniques, on the other hand, only learn from online experience and do not require prior knowledge of either the model of the environment or the reward function. The online estimation is based on sampling the environment and observing the consequences of choosing certain actions in certain states. Strikingly, learning from online experience can lead to optimal behaviours. Monte Carlo methods take averages of sample returns in episodic tasks. That means that the estimates are done on a well-defined horizon and consequently involves incremental episode-by-episode averages of returns. Here first-visit Monte Carlo are considered, which take the average of all returns following the first occurrence of s within an episode. So for all first visits to a state s the return is calculated for that state and appended to a list. By the end of the episode the list of visited states contains a list of returns for all states that occurred first in the episode. This estimation converges to $V^{\pi}(s)$ as the number of first visits to state s approaches ∞. By the law of large number as the number of averages of these estimates increases, the sample mean will tend to approach (and stay close to) the expected value. Each average is itself unbiased and the standard deviation of its error falls as $1/\sqrt{n}$, where n is the number of returns averaged.

One of the fundamental differences to dynamic programming is that Monte Carlo methods are n-step transitions, where n is the number of transitions within an episode, while dynamic programming includes only one-step transitions. Another difference is that Monte Carlo methods do not bootstrap due to the independence of the estimates for each state. An advantage over dynamic programming is the fact that Monte Carlo estimates can be performed on a subset of all states.

If a model of the environment, i.e., the transition probabilities and the reward function, are not available, then it is useful to estimate action values rather than state values, because all action values must be estimated in order for the values to be useful in suggesting a policy. The policy evaluation problem for action values is to estimate $Q^{\pi}(s, a)$, the expected return when starting in state s, taking action a, and thereafter following policy π. So, the first-visit Monte Carlo scheme estimates the returns following the first time in each episode that the state was visited and the action was selected. One of the biggest problem with Monte Carlo, in fact most estimation methods, is that deterministic policies do not admit to covering the full action space, because only one of the actions in each state will be observed. In order to learn to control an unknown environment, two opposing objectives have to be combined. On the one hand, the environment must be sufficiently explored in order to make qualitative decisions. Exploration is necessary to gain knowledge. On the other hand, the acquired knowledge must be exploited, for example to avoid bad exploratory actions in the past. The fundamental concepts of exploration and exploitation are generally opposing in that exploration seeks to minimise learning time, while exploitation seeks to minimise cost.

Generally, the smaller the learning time, the larger the costs, and vice versa [153]. However, purely exploring the environment while maximising knowledge gain does not necessarily minimise the cost, because irrelevant parts of the environment are explored. To overcome this complication continuous exploration and simultaneous exploitation of the action space has to be assured. One way to achieve this is to set non-zero probabilities for selecting actions in a given starting state of each episode.

Monte Carlo control is concerned with approximating optimal policies and fits into the pattern of generalised policy iteration, where approximate policies and approximate value functions are maintained and iteratively improved. Under the assumption of exploring starts and infinite episodes, Monte Carlo methods will compute Q^{π_k} exactly for arbitrary π_k using policy evaluation. Policy improvement is achieved by turning the policy greedy with respect to the action-value function. The corresponding greedy policy that chooses deterministically an action with maximal Q-value is denoted

$$\pi(s) = \arg\max_a Q(s, a), \; \forall \; s \in \mathcal{S}. \tag{4.17}$$

Constructing each π_{k+1} this way as the greedy policy with respect to Q^{π_k} implies that the policy improvement theorem (Theorem 4.1) holds for all $s \in \mathcal{S}$,

$$\begin{aligned} Q^{\pi_k}(s, \pi_{k+1}(s)) &= Q^{\pi_k}(s, \arg\max_a Q^{\pi_k}(s, a)) \\ &= \max_a Q^{\pi_k}(s, a) \\ &\geqslant Q^{\pi_k}(s, \pi_k(s)) \\ &= V^{\pi_k}(s). \end{aligned} \tag{4.18}$$

Since the policy improvement theorem holds, convergence to the optimal policy and optimal value function is guaranteed. For Monte Carlo control, this means that no prior knowledge about the environment is necessary given only observations from sample episodes. The assumption of infinite episodes can be removed analogue to value iteration in dynamic programming. Since, Monte Carlo methods are based on episodes, it is natural to alternate between policy evaluation and improvement on an episode-by-episode basis. Concretely, after each episode, the observed returns are used for policy evaluation to move the value function towards Q^{π_k}, and then the policy is improved at all states visited in this episode.

4.3.2.1 On-Policy Monte Carlo Control

On-policy Monte Carlo methods are used to overcome the assumption of exploring starts in order to ensure that all actions are selected infinitely often. On-policy thereby refers to methods that evaluate or improve the policy that is used for decision making. Additionally, on-policy methods are generally soft, that is all action probabilities for all states are non-zero, i.e., $\pi(s, a) > 0, \; \forall \; s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$. ϵ-soft policies assign a probability to selecting each action in each state of at least $\epsilon/|\mathcal{A}(s)|$.

One such policy is called ϵ-greedy. With probability $1 - \epsilon$ the action with highest Q-value is chosen and the other ones are randomly selected (with uniform probability) for a proportion $\epsilon > 0$.

$$\pi(s) = \begin{cases} 1 - \epsilon + \frac{\epsilon}{|\mathcal{A}(s)|}, & \text{if } a = \arg\max_a Q(s,a) \\ \frac{\epsilon}{|\mathcal{A}(s)|}, & \text{if } a \neq \arg\max_a Q(s,a) \end{cases} \tag{4.19}$$

ϵ-greedy policies are the closest to greedy policy. Their disadvantage is that among the remaining non-greedy actions, each is selected with equal probabilities.

Another ϵ-soft policy, called the Boltzmann policy, overcomes this limitation by varying the probabilities according to a gradient function of the estimated value

$$\pi(s) = \frac{e^{Q(s,a)/\tau}}{\sum_{b \in \mathcal{A} \setminus a} e^{Q(s,b)/\tau}} \tag{4.20}$$

The Boltzmann policy accounts for a τ parameter (temperature) which allows action selections to be equal ($\tau = \infty$) or greedy ($\tau = 0$). This, however, makes the Boltzmann policy sometimes difficult to tune, because one has to have an idea on the range of the Q-values.

Constructing a generalised policy iteration scheme for Monte Carlo control without exploring starts but with an ϵ-greedy policy requires the policy to be moved towards a greedy policy. The policy improvement theorem (Theorem 4.1) assures that any ϵ-greedy policy, π', with respect to Q^π is an improvement over any ϵ-soft policy π.

$$\begin{aligned} Q^\pi(s, \pi'(s)) &= \sum_a \pi'(s,a) Q^\pi(s,a) \\ &= \frac{\epsilon}{|\mathcal{A}(s)|} \sum_a Q^\pi(s,a) + (1-\epsilon) \max_a Q^\pi(s,a) \\ &\geqslant \frac{\epsilon}{|\mathcal{A}(s)|} \sum_a Q^\pi(s,a) + (1-\epsilon) \sum_a \frac{\pi(s,a) - \frac{\epsilon}{|\mathcal{A}(s)|}}{1-\epsilon} Q^\pi(s,a) \\ &= \frac{\epsilon}{|\mathcal{A}(s)|} \sum_a Q^\pi(s,a) - \frac{\epsilon}{|\mathcal{A}(s)|} \sum_a Q^\pi(s,a) + \sum_a \pi'(s,a) Q^\pi(s,a) \\ &= V^\pi(s). \end{aligned} \tag{4.21}$$

The corresponding algorithm realising the ϵ-soft on-policy Monte Carlo control is given in Algorithm 4.4.

4.3.2.2 Off-Policy Monte Carlo Control

In contrast to on-policy methods, off-policy Monte Carlo control does not estimate the value of a policy while using it. Instead, off-policy methods employ two policies, the behaviour policy to select the actions, and the estimation policy that is evaluated and improved. Both policies may be unrelated, a fact which can be used to exploit a greedy policy for evaluation and improvement, while using a stochastic (non-zero probabilities) policy to sample all possible actions. Algorithm 4.5 presents an off-policy Monte Carlo control algorithm based on the generalised policy iteration

Algorithm 4.4: ϵ-soft on-policy Monte Carlo control algorithm

```
$\left.\begin{array}{l} Q(s,a) \leftarrow \text{arbitrary} \\ Returns(s,a) \leftarrow \text{empty list} \\ \pi \leftarrow \text{an arbitrary } \epsilon\text{-soft policy} \end{array}\right\} \quad \forall\, s \in \mathcal{S},\ a \in \mathcal{A}(s)$
repeat
    Generate an episode using $\pi$
    foreach pair $s, a$ appearing in the episode do
        $R \leftarrow$ return following the first occurrence of $s, a$
        Append $R$ to $Returns(s,a)$
        $Q(s,a) \leftarrow average(Returns(s,a))$
    end
    foreach $s$ appearing in the episode do
        $a^* \leftarrow \arg\max_a Q(s,a)$
        forall $a \in \mathcal{A}(s)$ do
            $\pi(s) \leftarrow \begin{cases} 1-\epsilon+\frac{\epsilon}{|\mathcal{A}(s)|}, & \text{if } a = a^* \\ \frac{\epsilon}{|\mathcal{A}(s)|}, & \text{if } a \neq a^* \end{cases}$
        end
    end
until forever
```

principle for computing Q^*. The behaviour policy π' is maintained as an arbitrary soft policy, while the estimation policy π is the greedy policy with respect to an estimate of Q^π.

4.3.3 Temporal Difference Methods

Temporal difference (TD) methods are a significant extension to the previous approaches presented in Section 4.3.1 and Section 4.3.2 in that TD methods combine the best of both approaches. Like Monte Carlo methods TD methods do not require prior knowledge of the environment. They are able to learn from experience alone. Also, they employ concepts of dynamic programming in that TD methods do not rely on the final outcome to improve estimates, i.e., they bootstrap. More concretely, TD methods form a target at time $t+1$ and make an update using the observed reward r_{t+1} and the estimate $V(s_{t+1})$. The most simple form of a TD update rule is

$$V(s_t) \leftarrow V(s_t) + \alpha\left[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)\right]. \tag{4.22}$$

Because the update rule (4.22) uses the differences in state-values between successive states, these methods are commonly called temporal difference (TD) equation. In order to illustrate the subtle difference between Monte Carlo methods and TD methods, consider the Bellman update rule to estimate the value function $V^\pi(s)$:

Algorithm 4.5: An off-policy Monte Carlo control algorithm

```
Q(s, a) ← arbitrary                              ⎫
N(s, a) ← 0   (Numerator of Q(s, a))             ⎪
D(s, a) ← 0   (Denominator of Q(s, a))           ⎬  ∀ s ∈ 𝒮, a ∈ 𝒜(s)
π ← an arbitrary deterministic policy            ⎭
repeat
    Select a policy $\pi'$ and use it to generate an episode
    $\tau \leftarrow$ latest time at which $a_\tau \neq \pi(s_\tau)$
    foreach pair s, a appearing in the episode at time τ or later do
        $\tau \leftarrow$ the time of first occurrence of s, a such that $t \geqslant \tau$
        $\omega \leftarrow \prod_{k=t+1}^{T-1} \frac{1}{\pi'(s_k, a_k}$
        $N(s, a) \leftarrow N(s, a) + \omega R_t$
        $D(s, a) \leftarrow D(s, a) + \omega$
        $Q(s, a) \leftarrow \frac{N(s,a)}{D(s,a)}$
    end
    foreach $s \in \mathcal{S}$ do
        $\pi(s) \leftarrow \arg\max_a Q(s, a)$
    end
until forever
```

$$
\begin{aligned}
V^\pi(s_t) &= \mathbb{E}_\pi\{R_t \mid s_t = s\} && (4.23)\\
&= \mathbb{E}_\pi\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s\right\}\\
&= \mathbb{E}_\pi\left\{r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} \mid s_t = s\right\}\\
&= \mathbb{E}_\pi\{r_{t+1} + \gamma V^\pi(s_{t+1}) \mid s_t = s\} && (4.24)
\end{aligned}
$$

Monte Carlo methods use a sample return to estimate the real expected return in equation (4.23). Dynamic programming, on the other hand, use the current estimate $V_t(s_{t+1})$ in place of $V^\pi(s_{t+1})$, while the expected value (the whole expression of (4.24)) is assumed to be provided by the model of the environment. TD methods combines the sampling of Monte Carlo methods with the bootstrapping of dynamic programming by sampling the expected value in equation (4.24) and using the current estimate of V_t in place of V^π. Algorithm 4.6 realises the procedure of TD(0).

TD methods have a number of advantages over the Monte Carlo methods and dynamic programming. One of the most obvious and already discussed is the ability to learn without having a model of the environment, of its rewards, and next-state probability distributions. Further, TD methods are online, fully incremental algorithms, where learning is not delayed until the end of an episode (like Monte Carlo methods), but instead only observed state transitions are used to improve the estimate of V^π. This is unlike policy evaluation in dynamic programming (equation (4.10)), which involves all possible successor states to update the value function. This difference can be

Algorithm 4.6: Tabular TD(0) for estimating V^{π}

```
Initialise $V(s)$ arbitrarily, $\pi$ to the policy to be evaluated
repeat
    Initialise $s$
    repeat
        $a \leftarrow$ action given by $\pi$ for $s$
        Take action $a$, observe reward, $r$, and next state, $s'$
        $V(s) \leftarrow V(s) + \alpha [r + \gamma V(s) - V(s)]$
        $s \leftarrow s'$
    until for each step of episode until $s$ is terminal
until for each episode
```

understood as sampling the current environment to make the adjustments to the value function. At the expense of computation time, TD methods could include such simulated state transitions. The online, fully incremental features of TD methods are of particular interest, where episodes are very long or the learning application faces continuous tasks. Finally, for any fixed policy π, the TD method given in Algorithm 4.6 has been proved to converge to V^{π} given a constant learning rate α under the conditions that it is sufficiently small. A stronger convergence guarantee can be obtained (with probability 1), if the learning rate decreases according to the stochastic approximation (Robbins-Monro) conditions

$$\sum_{k=1}^{\infty} \alpha_k(a) = \infty \quad \text{and} \quad \sum_{k=1}^{\infty} \alpha_k^2(a) \leqslant \infty, \tag{4.25}$$

where $\alpha_k(a)$ is the k-th learning rate for action a. The first condition ensures that initial conditions are overcome, while the second one guarantees that the steps become small enough to assure convergence.

4.3.3.1 SARSA-Learning

Instead of learning state-value function as with the TD(0) algorithm discussed previously, SARSA learns an action-value function. In particular, SARSA estimates $Q^{\pi}(s, a)$ for the current behaviour policy π and for all states s and all actions a. As such it is an on-policy method and conceptually similar to the TD(0) update rule given in equation (4.22) and formally the same convergence theorems apply:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha [r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]\,. \tag{4.26}$$

The name SARSA refers to the use of a quintuple of events, $\langle s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1}\rangle$, that qualify a transition of the Markov chain from one state-action pair to the next state-action pair. Algorithm 4.7 presents the SARSA control algorithm. The update rule (4.26) is done after every transition from a non-terminal state s_t.

Algorithm 4.7: SARSA: An on-policy TD control algorithm

```
Initialise $Q(s, a)$ arbitrarily
repeat
    Initialise $s$
    Choose $a$ from $s$ using policy derived from $Q$ (e.g., $\epsilon$-greedy)
    repeat
        Take action $a$, observe reward, $r$, and next state, $s'$
        Choose $a'$ from $s'$ using policy derived from $Q$ (e.g., $\epsilon$-greedy)
        $Q(s, a) \leftarrow Q(s, a) + \alpha [r + \gamma Q(s', a') - Q(s, a)]$
        $s \leftarrow s'$
        $a \leftarrow a'$
    until for each step of episode until $s$ is terminal
until for each episode
```

4.3.3.2 Q-Learning

Q-learning is an off-policy algorithm that learns the optimal state-action function, Q^*, independent of the policy being followed [170]. The one-step Q-learning update rule is defined as

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right]. \tag{4.27}$$

Q-learning is model-free, meaning that it does not require knowledge of the transition or reward functions. The learning rate, α, specifies the contribution of the update target $r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)$ to the current estimate of $Q(s_t, a_t)$. If the learning rate $\alpha = 1$, then the update target overwrites the current estimate and if $\alpha = 0$ then the update target is cancelled out.

Theorem 4.2 (Q-learning convergence). *The Q-learning algorithm given by the update equation* (4.27) *converges to the optimal Q^* values with probability* 1 *if [75, 155, 170]*

1. *the state and action spaces are finite;*
2. $\sum_{t=1}^{\infty} \alpha_t(s, a) = \infty$ *and* $\sum_{t=1}^{\infty} \alpha_t^2(s, a) \leqslant \infty$ *uniformly over* s *and* a *with probability* 1*;*
3. $\mathrm{Var}\{r_s(a)\}$ *is finite.*

Algorithm 4.8 presents the algorithm for Q-learning.

4.3.3.3 Generalisation over State Spaces

So far previous sections discussed finite MDPs and their practical implementations of solution methods. Once large or continuous state (or action) spaces enter the application domain, tabular representations of the state-value (or action-value) functions become prohibitively expensive in terms of memory due to the curse of dimensionality. Therefore, generalisations are sought that provide a parameterised functional form to approximate the respective value functions. In this sense function approximators are often employed to achieve generalisation, which include neural networks among others. These kind of techniques are generally instances of supervised learning,

Algorithm 4.8: Q-Learning: An off-policy TD control algorithm

```
Initialise $Q(s, a)$ arbitrarily
repeat
    Initialise $s$
    repeat
        Choose $a$ from $s$ using policy derived from $Q$ (e.g., $\epsilon$-greedy)
        Take action $a$, observe reward, $r$, and next state, $s'$
        $Q(s, a) \leftarrow Q(s, a) + \alpha [r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$
        $s \leftarrow s'$
    until for each step of episode until $s$ is terminal
until for each episode
```

where the training data consists of pairs of input vectors and desired outputs. In the case of neural network function approximators, the state is represented as input neurons and the output represents the value function to be approximated. The generalisation capabilities of function approximators eliminate the need to visit every state in the state space infinitely often, because the functional representation is able to capture states that are "close" to each other. That means that something can be said about states not being visited, if previously nearby states have been explored.

A neural network comprises a network of primitive functions, where each primitive function is represented as a neuron that integrates the signals from each input channel. Usually, the input channels have an associated weight. This simply means that the scalar input x_i is multiplied by the weight w_i. The primitive function can be chosen arbitrarily, but it is common to choose a sigmoidal squashing function. An abstract neuron is depicted in Figure 4.2.

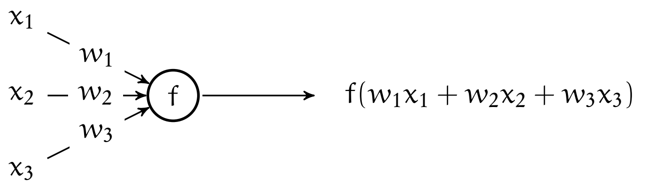


Figure 4.2: An abstract neuron

A network of interconnected primitive functions is shown in Figure 4.3, which exhibits two input signals, one hidden layer with four neurons, and one output unit.

The neural network function approximator is a compact mapping, $Q : \mathbb{R}^n \rightarrow \mathbb{R}$, from an n-dimensional real input $(s_1, s_2, \ldots, s_n)$ (here the state representation) to a real output $Q(s)$ (here the state-action value). For a systematic introduction into the biological paradigm that underlies neural networks, their theoretical foundations, and uses see Rojas [123].

The essential character of such networks is that similar input vectors are mapped to a similar output (many inputs are mapped to one output in the case of approximating value functions). This allows networks to make reasonable generalisations over input vectors that have not exactly been seen before [127]. Additionally, the generalisation capabilities scale very well with the size of the state space, because the parameters of neural networks represented as a weight vector, ω, are usually much smaller than the number of state variables. The prediction methods described in previous sections back up an estimated value function in a particular state, s, towards this backed-up value.

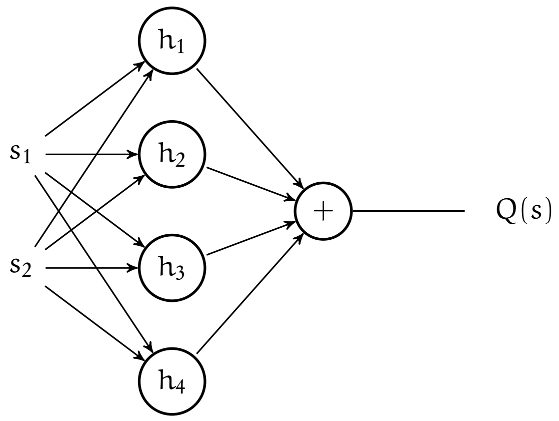


Figure 4.3: An artificial Neural Network

The notation $s \rightarrow \upsilon$ represents the state s being backed-up by a target υ and expresses that an estimate for state s should be moved towards υ.

Neural network (in fact as employed for this thesis multi-layer perceptrons) seek to minimise the mean-squared error of the estimated output (state-value of state s) and the actual output (target υ). One way of achieving this, is to perform a steepest descent on the surface in weight space, where the height corresponds to the error. The standard back-propagation training algorithm is one of the training algorithms that minimises the mean-squared error and adjusts the weights accordingly to move closer to the target value. For detailed derivation of the formulas see [127].

To facilitate a compact representation of the state space standard backpropagation feedforward neural networks with one hidden layer are generally expressive enough to learn non-linear function estimates of the Q-values for each action given a state vector [127]. So the training samples have the form $s_t, a_t \rightarrow \upsilon_t$. Alternatively, each agent could employ a neural network with one output neuron for each action. The disadvantage of this approach is that the utility of one action affects the other ones as well and hence results in poorer performance compared to using neural networks for each action separately [3]. Equation (4.26) can then be expressed as the general gradient-descent update rule for neural network training as

$$\Delta\omega_{t+1} = \alpha[\upsilon_{t+1} - Q(s_t, a_t)]\nabla_{\omega_t} Q(s_t, a_t), \tag{4.28}$$

$$\upsilon_t = r_t + \lambda Q(s_t, a_t), \tag{4.29}$$

where $\nabla_{\omega_t} Q(s_t, a_t)$ is the vector of partial derivatives of the value function $Q(s_t, a_t)$ with respect to the weight vector ω_t.

So far, non-linear function approximation, such as neural networks, poses challenges to convergence proofs. While convergence for on-policy TD methods have been brought forward, off-policy methods, such as Q-learning, have shown unstable behaviours, if the behaviour policy is not sufficiently close to the estimation policy. Only recently have convergence proofs for non-linear function approximators been established [92], but there have been excellent examples of their use in practical applications, such as TD-gammon [149], hybrid RL for resource allocation tasks [152], or motor control [37].

Neural networks suffer from ill-conditioning, where weights show different sensitivities towards the error function (mean-square error of input/output pairs). When a global learning rate is applied to each weight in the weight vector, ω, then the different scales of sensitivities are not addressed. Several solutions have been proposed, but it is unclear whether these algorithms are generally applicable [13]. This implies that more empirical analysis of neural network function approximations are necessary to investigate their practical value in different scenarios.

4.4 Multi-agent Systems

In the previous sections it is generally assumed that the environment is stationary and that a single agent learns a policy that optimises the learning objective. The single agent semantics, however, become a bottleneck in environments that are inherently decentralised. Not only do central authorities introduce single points of failure in large environments, but also the correlation of dispersed information and the dissemination of control instructions become prohibitively expensive. While it is conceivable to devise single-agent strategies in for example load-balancing tasks, where a system designer is responsible for the maintenance and control of the entire system, many decentralised system are heterogeneous in nature that require multiple agents to accomplish desired goals. Such systems that consist of multiple autonomous agents that can potentially interact with each other are called multi-agent systems (MAS). It is an area of distributed artificial intelligence (AI) studying the joint behaviour and the complexities that arise through their interactions.

Multi-agent systems can broadly be classified as being either cooperative or competitive [71]. The cooperative nature arises when the agents pursue a common goal which expresses itself in benign behaviour towards each other to achieve this goal, i.e., there is no inherent conflict in the system design of the interactions. In contrast competitive agents act selfishly to maximise the reward of their actions. They do so without consideration of their counterparts in the environment. Mostly, the selfish behaviour arises through heterogeneous goal structures, meaning that agents in the environment follow different goals or even maintain conflicting goals in their own decision process. The heterogeneity comes about in scenarios where the control over agents is dispersed among competing companies or entities involved in trading and allocation of resources. Naturally, the competitive environments can be encompassed by game-theoretic terms which is a longer established field than autonomous agent technology.

However, the distinction between those classifications is not always as clear-cut. The emergence of different behaviours may tend towards temporary cooperation in competitive settings, if it suits the respective agents. Or, an agent may monopolise a resource with adverse effects to its cooperative environment [71]. Additionally, a game-theoretic framework can be specified in such a way that cooperative behaviour emerges. Section 4.4.1 reviews some of the basic principles of game theory and their instantiation as normal-form games. The state of the art in multi-agent reinforcement learning is then presented in Section 4.4.2 to cover cooperative and competitive approaches.

4.4.1 Introduction to Game Theory

Relying on the mathematical framework of Markov Decision Processes (see Definition 4.1) and blindly extending those to the multi-agent setting poses some serious challenges. Recall that unique solutions for the Bellman optimality equations (4.8) and (4.9) exist for finite MDPs. This means that, for any MDP, there is an optimal policy assuming that the policy is stationary and deterministic. A stationary policy does not change over time and deterministic action selection implies that an agent in state $s \ \forall \ s \in \mathcal{S}$ always chooses the same action. These assumptions are violated in multi-agent systems and need to be taken care of. Consider two interacting agents whose individual decisions affect the total payoff. This setting involves interdependence and allows for cooperative or competitive interpretations. These kind of interactions constitute a game with strategies and payoffs/utilities as their fundamental representation. Strategies and payoffs/utilities correspond to the terms policies and rewards in reinforcement learning respectively. So game theory mathematically captures the behaviour of strategic situations where the decisions of two or more parties impact all.

Assuming deterministic policies as in the single-agent case to solve the MDP can quickly lead to problems in finding optimal policies. For example the classic game of "rock, paper, scissors" allows agents with a deterministic policy to be consistently defeated. Consequently, multi-agent systems often require a probabilistic policy. This requirement stems from the interdependence of the agents decisions and the resulting uncertainty of an opponent's action selection.

In order to understand how selfish or utilitarian agents handle different strategic situations the most simple form of games with two agents and two actions are considered [71]. These games can be represented using a payoff matrix which defines the utility each agent will receive given their actions. The example game matrices given in Table 4.1 to Table 4.4 are also known as normal form matrices. It is assumed that the actions are taken simultaneously by each player. Both players will then receive a payoff, or utility value, based on their joint action. In these simple games it is also assumed that all players have a common knowledge about the payoff matrix. Based on the utility values prescribed in the payoff matrix, different situations arise. As such, games can broadly be classified as either trivial, games of no conflict, games of complete opposition, or as games of partial conflict.

Trivial games arise in situations where agents can act independently of the other agents. This implies that the expected reward of an agent is also independent.

Table 4.1: A trivial game

	A2	B2
A1	2,2	2,2
B1	2,2	2,2

Table 4.1 gives the payoff (or reward) matrix for a trivial game. Each cell in this matrix details the payoff structure for a joint action, i.e., the row player selects an action followed by the column player. The total payoff is the sum of the individual numeric values within a cell. Since the individual utilities are equal no matter which actions are selected, both utilitarian and selfish players cannot be distinguished. In game theory a strategy π is defined to be the set of actions all players take. For example, a strategy of $\pi = (A1, A2)$ assigns player 1 and player 2 a utility of 2 each.

No-conflict games also offer little distinction between the selfish and the utilitarian player shown in Table 4.2. Note that for both individual reward and group payoff the unambiguous strategy $\pi = (A1, A2)$ is preferred. If either player was to deviate from this strategy, both would be worse off. Unless the row player aims to maximise the relative utility, which is the case for choosing action B1, there is no incentive to change the strategy. If the joint action was ambiguous, i.e., $\pi = (B1, B2)$ would have payoff 4, 4, then utilitarian agents face the problem of choosing the action that maximises the group payoff, because the decisions have to be made individually. So, in the absence of consensus among the agents over which actions to perform, cooperation is problematic.

Table 4.2: A no-conflict game

	A2	B2
A1	4,4	2,3
B1	3,2	2,2

Games of complete opposition, or zerosum games, are challenging for both selfish and utilitarian players. These games fall into the category of adversarial contests, such as “rock, paper, scissors”, in which one player wins and the other one loses as a consequence. One player’s gain in terms of the payoff must come at the other player’s loss. An example is given in Table 4.3. For a selfish player the best strategy is a random one with equal probabilities for each action. In the face of zero payoff for joint actions, utilitarian players also tend to the random strategy. Strategies that assign probabilities to their action choices are mixed strategies. This is in contrast to pure strategies, which always select the same action, i.e., probability 1, and corresponds to the greedy policy introduced in Section 4.2.

Table 4.3: A game of complete opposition

	A2	B2
A1	0,0	2,-2
B1	1,-1	-3,3

As the name suggests, partial conflict games allow for profitable actions, but the selfish agents prefer different solutions. In the example given in Table 4.4 a selfish row player prefers joint action $\pi = (B1, B2)$, while the selfish column player prefers $\pi = (A1, A2)$. However, if the row agent selects action B1 and the column agent selects A2, a joint action is chosen that neither prefers. In contrast, the utilitarian players unanimously prefer strategy $\pi = (B1, B2)$.

Table 4.4: A partial conflict game

	A2	B2
A1	2,7	-1,-10
B1	1,-5	10,1

Given a game specification, one is still interested which strategy to use. Are there best strategies that can be followed in any given game? It turns out the answer is not that simple. Morgenstern and Von Neumann [104] proposed a conceptual solution in which an agent always takes the action which maximises the worst possible utility it could get. This is known to be the maxmin strategy. In a two-player game, player i’s maxmin strategy is given by

$$\pi_i^* = \max_{\pi_i} \min_{\pi_j} u_i(\pi_i, \pi_j), \tag{4.30}$$

where u_i is the utility given both players' strategy. So player i assumes that player j always selects the action that is worst for player i. Thus, given this assumption player i takes the best possible action. Von Neumann proved that a strategy that minimises the maximum loss can always be found for all two player zerosum games. However, it has been shown that this strategy is not stable in the general case, which means that if player i knows that the opponent follows the maxmin strategy, then player i prefers to deviate.

Another approach that offers unequivocal benefits are dominant strategies which says that agent i is best off following the dominant strategy regardless of the opponent's strategies. Formally, a pure strategy is dominant for player i if

$$\forall_{\pi_{-i}} \forall_{a_i \neq \pi_i} \; u_i(\pi_{-i}, \pi_i) \geqslant u_i(\pi_{-i}, a_i), \tag{4.31}$$

where π_{-i} represents the strategies of all agents except i.

From the cooperative perspective, a social welfare strategy can be formed that maximises the sum of everyone's payoff denoted as

$$\pi^* = \arg\max_{s} \sum_{i} u_i(\pi). \tag{4.32}$$

Following the discussion of no-conflict games as in Table 4.2, utilitarian and selfish agents settle for a strategy that benefits both similarly. However, if joint actions are ambiguous, then the selfish agent tends to maximise its own benefit, even though the benefit of the group could be much worse off. So, social welfare strategies might not be stable. Additionally, the payoff structure might only benefit one agent, while others get almost nothing.

The Pareto optimal strategy, however, solves the unfairness problem. It states that any unilateral deviations of agent i from a strategy π to improve its payoff is impossible without making the opponents worse off. So the set of Pareto optimal strategies can be defined to be the set

$$\left\{\pi \mid \neg\exists_{\pi' \neq \pi}(\exists_i u_i(\pi') > u_i(\pi) \wedge \neg\exists_{j \in -i} u_j(\pi) > u_j(\pi'))\right\}. \tag{4.33}$$

While Pareto optimal strategies are highly desirable from a utilitarian perspective, solutions can be unstable, if multiple Pareto optimal strategies exist. In the face of choice selfish agents will maximise their own payoff to the disadvantage of others.

John F. Nash solved the stability problem with a strategy called Nash equilibrium. A Nash equilibrium constitutes strategies where no agent is better off by changing its strategy unilaterally while the opponents keep theirs fixed. Formally, the set of all Nash equilibrium strategies is defined by

$$\left\{\pi \mid \forall_i \forall_{a_i \neq \pi_i} \; u_i(\pi_{-i}, \pi_i) \geqslant u_i(\pi_{-i}, a_i)\right\}. \tag{4.34}$$

Nash proved that all game matrices have at least one Nash equilibrium. Nash equilibria become problematic when many of them exist which is often the case. In this case, some Nash equilibria might be better for some agents than others. Resolving ambiguity requires communication between agents to find an agreed upon strategy that benefits all agents involved.

Repeated games can change the game dynamics significantly. Especially for selfish agents, repeated games opens opportunities that are unavailable in playing a game once. One way to take advantage of repeated games is to model the behaviour of the opponents and based on their bias to choose certain actions. The learning problem is then two-fold: first learn the strategy of the opponent and then learn the best counter-strategy to achieve the highest possible reward.

A famous game to study cooperative behaviour in repeated games is the Prisoner's Dilemma whose payoff matrix is given in Table 4.5. This game is governed by three intricate rules. Each player has a choice of cooperating or defecting. A reward $R = 3$ is payed for mutual cooperation. However, the player is tempted to defect which gives a reward of $T = 5$. It is assumed that $T > R$, so that it pays off to defect, if the other player cooperates. However, if the other player defects and the player in question cooperates, a sucker's payoff $S = 0$ is rewarded to the other player. If both defect, then both are punished with $P = 1$. The other assumption is that $P > S$ in order to entice the other player to defect. So, it pays off to defect. The dilemma is established through the fact that mutual defection pays less than mutual cooperation [11]. In a tournament conducted by Axelrod and Hamilton, the tit-for-tat strategy turned out to be the most successful one. The gist of this strategy is very simple: on the first move cooperate, and then respond in kind to the opponent's move.

In a single-shot game, the dominant strategy is to always defect, despite the higher reward for both cooperating. In the iterated version of the game, the risky joint action is the dominant strategy, while it guards against exploitation. If the opponent defects, then the next round starts off with defection and the opponent responds with defect. This indicates to the tit-for-tat player a cooperatively minded opponent and so he reverts to cooperate again [11].

Table 4.5: A Prisoner's Dilemma game

	C2	D2
C1	3,3	0,5
D1	5,0	1,1

This fact has been formalised into the folk theorem. Informally, it states that a feasible equilibrium strategy is one that is not Pareto dominated by another strategy and where each player's payoff is at least as high as the amount he would receive when his opponents adopted the maxmin strategy (4.30).

More formally,

Definition 4.2 (Enforceable)**.** *A payoff profile* $\mathbf{r} = \{r_i\}_{i=1}^{N}$ *is enforceable if* $\forall\ i \in N, r_i \geqslant \pi_i^*$ *[138].*

Definition 4.3 (Feasible)**.** *A payoff profile* $\mathbf{r} = \{r_i\}_{i=1}^{N}$ *is feasible if there exist rational, non-negative values* α_a*, such that for all* i*, we can express* r_i *as* $\sum_{a \in \mathcal{A}} \alpha_a u_i(a)$*, with* $\sum_{a \in \mathcal{A}} \alpha_a = 1$ *[138].*

Then formally, the folk theorem is given as

Theorem 4.3 (Folk Theorem)**.** *Consider any n-player normal form game G and any payoff profile* $\mathbf{r} = \{r_i\}_{i=1}^{N}$ *[138].*

1. *If* $\mathbf{r}$ *is the payoff profile for any Nash equilibrium* π *of the infinitely repeated game G with average rewards, then for each player* i, r_i *is enforceable.*

2. *If* $\mathbf{r}$ *is both feasible and enforceable, then* $\mathbf{r}$ *is the payoff profile for some Nash equilibrium of the infinitely repeated game G with average rewards.*

So, the maximum attainable payoff player i can guarantee itself when assuming that the opponent always selects the action that is worst for player i is given through the maxmin strategy (4.30). So, the folk theorem says that any joint payoff $\mathbf{r} = (\mathbf{r}_1, \mathbf{r}_2)$, such that $\mathbf{r}_1 \geqslant \pi_1^*$ and $\mathbf{r}_2 \geqslant \pi_2^*$ can be sustained, meaning it is enforceable, if one player deviates with adverse effects on the agents, and it is feasible, meaning a strategy exists that holds for the inequalities given above.

For multi-agent learning, this theorem has profound implications in that agents should thrive for long-term rather than short-sighted (myopic) benefits. This, however, is a formidable task since agents generally do not know their counterparts and their preferences. The folk theorem provides a minimum utility an agent should receive through the maxmin strategy and also indicates that frequently agents will be able to do better than that, if agents would coordinate.

Before framing the multi-agent systems domain into the context of reinforcement learning, cooperative and competitive approaches and some of their application domain are explored from existing survey literature [26, 71, 117]. The next chapter's analysis demonstrates the utility of the integrated modelling approach, only online techniques based on reinforcement learning to learn value functions are investigated. So other techniques, such as evolutionary game theory, auctions, mechanism design, etc. are not covered.

4.4.2 Multi-agent Reinforcement Learning

The game-theoretic concepts introduced in the previous section all have a specification of the payoff matrix. However, many multi-agent applications do not know the payoffs a priori, but instead the payoffs need to be acquired through repeated and exploratory interactions with the environment. Thus, the formalisms of the MDP (see Definition 4.1) can be generalised to account for multiple agents as

Definition 4.4 (DEC-MDP). *An n-agent continuous state* ***DEC-MDP*** *of a queueing network is defined by a tuple* $\mathcal{M} = \langle N, \mathcal{A}, \mathcal{S}, \mathcal{P}, \mathcal{R}, \Omega, \mathcal{O} \rangle$, *where*

- N *is the number of agents in the environment.*

- $\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2 \times \cdots \times \mathcal{A}_N$ *are finite action sets for each agent* i.

- $\mathcal{S}$ *is finite set of states.*

- $\mathcal{P}_{ss'}^{\vec{a}} = P\{s_{t+1} = s' \mid s_t = s, \vec{a}_t = \{a_1, \dots, a_i\}\}$ *is the transition probability of state* s' *when the actions* $\{a_1, \dots, a_i\}$ *have been taken in state* s.

- $\mathcal{R}_{ss'}^{\vec{a}} = \mathbb{E}\{r_{t+1} \mid s_t = s, \vec{a}_t = \{a_1, \dots, a_i\}, s_{t+1} = s'\}$ *is the expected value of the next reward taking actions* $\{a_1, \dots, a_i\}$ *in state* s *and transitioning to the current state* s'.

- Ω *is the set of all observations for each of the agents.*

- $\mathcal{O}$ *is the observation function.* $O\left(s, \{a_1, \dots, a_i\}, s', \{o_1, \dots, o_i\}\right)$ *is the probability of agents 1 to* i *seeing observations* $\{o_1, \dots, o_i\}$ *after selecting actions* $\{a_1, \dots, a_i\}$ *in state s.*

While the DEC-MDP definition is quite general, it expresses the decentralised setting of multi-agent reinforcement learning in the following way. First, it is assumed that the global state space is not attainable by each individual agent. If the state representations for the individual agents are non-overlapping, then the state space can be factored.

Definition 4.5 (Factored state). *A **factored** n-agent DEC-MDP is a DEC-MDP such that the states can be factored into n components,* $\mathcal{S} = \mathcal{S}_1 \times \dots \mathcal{S}_n$.

Additionally, an agent's local decisions may only be based on its locally fully observable state.

Definition 4.6 (Locally fully observable). *A factored, n-agent Dec-MDP is said to be **locally fully observable** if* $P(s_i|o_i) = 1 \; \forall \; i, \dots, N$.

Definition 4.6 renders Ω and O in Definition 4.4 redundant which simplifies the definition further and reduces the problem space.

Generally, the transition of a state s to a state s' for an agent i depends on the actions of the other agents in the environment. This is expressed in the transition probability of Definition 4.4. The reward function, however, embodies the stance of the agents towards each other. If the action vector $\vec{a}$ in $\mathcal{R}^{\vec{a}}_{ss'} = \mathbb{E}\{r_{t+1} \mid s_t = s, \vec{a}_t = \{a_1, \dots, a_i\}, s_{t+1} = s'\}$ contains all actions of the agents forming a team, then the MARL is said to be cooperative. Otherwise, if the reward function only depends on the action of the individual agents then agents tend to be competitive.

4.4.2.1 Cooperative Multi-agent Reinforcement Learning

Engineering large-scale decentralised systems, whether it be load-balancing in server farms or swarm intelligence in robots, etc, is a huge challenge. Manually designing each entity and hard-wiring interaction patterns is difficult and often does not account for complex environments that are large, open, dynamic and unpredictable. Machine learning techniques have proven popular in solving multi-agent systems, because agents can be equipped with learning abilities in order to be able to learn complex behaviours through repeated trials with the environment.

The approaches of cooperative multi-agent systems can be broadly classified as team learning and concurrent learning [71, 117]. The former nominates a single agent to learn a set of behaviours for the whole team. The advantage of team learning approaches is that standard convergence proofs from the single-agent literature hold. However, such agent architectures suffer from an explosion of the state space on the one hand and scalability issues on the other. In complex domains with a large number of agents and a set of states $\mathcal{S}$ in the environment, the state space of the whole team of N agents might scale as $|\mathcal{S}|^N$. This explosion of the state space is both problematic for model-free and model-based reinforcement learning algorithms in terms of the memory required to store and explore the state space efficiently. Other techniques, such as evolutionary computation exist that can cope with the size of the state space. Additionally, single agents that accrue and correlate dispersed, possibly out of reach state information exhibit challenges to the communication infrastructure.

Depending on the problem domain, cooperative multi-agent systems can be divided into homogeneous and heterogeneous approaches. Homogeneous approaches drastically reduce the search space of the learning problem by assigning identical behaviours to all agents. In contrast, heterogeneous team learning a single learning authority learns different behaviours of the agents to the benefit of the team. The heterogeneous setting allows for task specialisation to emerge at the cost of an increased search space compared to homogeneous team learning.

Alternatively to the team learning approach concurrent learning is the most natural cooperative learning technique equipping each agent with learning capabilities. As agents learn behaviours given their own partial information of the state space the most fundamental assumptions of machine learning techniques are violated. In single-agent learning scenarios a learner explores its environment through repeated interaction to improve its behaviour. Since concurrent learning implies that multiple learners are co-evolving in a dynamic environment, learnt behaviour may become obsolete as a consequence of the other agents changing their behaviour. In reinforcement learning tasks the formalisms of the MDP needs to be extended to capture the dependence of an agent's decisions. The reward function is no longer solely defined over an agent's state and actions, but instead must also include all the agents' actions. Intuitively speaking, a cooperative task is rewarded upon completion and based on all the agents involved in achieving it. Any decisions by agents along the way of completing the task impact the reward.

The completion of a task through team effort raises the question of how to apportion the reward to the respective agents. The confounding signals a dynamic environment produces can be attributed to each agent's past actions. However, the actual source of a reward could have been either through the exploitative action to reach a goal or an exploratory action to reduce the uncertainty associated with the knowledge about an environment. The uncertainty about an environment is further compounded by the co-adaptation of other agents. Thus, the assignment of credit is a complicated issue. If the problem definition bestows a global utility function that specifies the overall performance criteria to the participating agents, then the multi-agent reinforcement learning approach to modelling a decentralised domain offers a scalable cooperative solution. For instance total throughput or total response time are natural performance criteria in queueing networks for global utility functions. If the problem specification does not accommodate a global utility function, the designer needs to construct one that aggregates all preferences efficiently and encourages robust cooperative behaviour [156, Chapter 1]. The most simple solution is to split the team reward evenly among the agents within the team. Such global and monolithic reward functions are often difficult to compute in a decentralised environment, because individual and dispersed reward signals cannot easily be aggregated. Also, an even split of the reward signal does not account for heterogeneous agent behaviours. Some agents might have been more involved than others to complete the task. If the intensity of involvement varies significantly, then evenly splitting the reward invites to laziness. In contrast to the global reward function, an agent could be assigned credit based on its actual contribution to the task. This is called local reward. Unfortunately, local reward functions encourage competitive rather than cooperative behaviours, because the agent is solely interested in maximising its local reward. Consequently, reward functions need to address the heterogeneous nature of credit assignment. Matarić [96] proposed a shaped reinforcement which calculates continuous rewards based on sensory input of the environment to estimate the progress of the current goal and local states in the foraging robot domain. Yet another approach is the "wonderful life utility" proposed by Wolpert and Tumer [179] that introduces a metric of how a task would have fared without the respective agent.

Section 4.3.3.2 presented the Q-learning algorithm which governs an agent's learning behaviour to simultaneously act and learn in uncertain environments. The learner successively improves its knowledge about the environment, however, to guarantee that the agent has found the optimal policy, the agent will need to visit every state and every action. Q-learning has also been applied in the multi-agent setting. Claus and Boutilier [33] applied the Q-learning algorithm to n-player cooperative repeated games, such as the following simple example

Table 4.6: A two-agent stage game

	A2	B2
A1	x	0
B1	0	y

If $x = y > 0$, then the two agents are facing ambiguous actions, because neither agent prefers any action over the other one. Lacking any means of coordinating the agents may choose suboptimal actions. Claus and Boutilier modelled the multi-agent problem using independent learners (IL) that are unaware of the other agent's action selection and joint action learner (JAL) that keep a Q-function for each joint action. Careful tuning of the exploration strategy and decreasing the exploration to zero over time ensures that an equilibrium is reached and that the agents do not escape the equilibrium. However, optimal equilibria can only be achieved by the JAL learners because joint Q-values are maintained.

Kapetanakis and Kudenko [78] concentrated on independent learners and introduced a Frequency Maximum Q-value heuristic (FMQ) that estimates the value for a given action a as

$$EV(a,t) = Q(a) + c * \frac{\int_0^t R(t,a)}{n_a} * \max \mathcal{R}(a), \tag{4.35}$$

$$R(t,a) = \begin{cases} 1 & \text{if } \max \mathcal{R}(a) \text{ obtained at time } t \\ 0 & \text{otherwise,} \end{cases}$$

where $\max \mathcal{R}(a)$ is the maximum reward encountered so far and the fraction calculates the number of times the maximum reward for action a has been attained over the number of executing action a in total. c denotes a weight that controls the importance of the FMQ heuristic. For $c = 0$ the expected value reduces to the standard Q-learning problem. As a result, the FMQ heuristic integrates how often a given action produces its maximum attainable reward. For deterministic games, this heuristic has shown to be quite successful converging almost surely. However, in fully stochastic games the application of this heuristic is problematic.

To accommodate stochastic games with noisy payoffs and model-free multi-agent reinforcement learning, Wang and Sandholm [169] proposed optimal adaptive learning (OAL), an algorithm that converges to a Pareto optimal Nash equilibrium with probability 1. For optimal adaptive learning, virtual games (VG) are constructed for each state of the game in order to eliminate suboptimal Nash equilibria. Given a state s and a joint action $\vec{a}$ the virtual game value is calculated as

$$VG(s, \vec{a}) = \begin{cases} 1 & \text{if } \vec{a} = \arg\max_{\vec{a}' \in \mathcal{A}} Q^*(s, \vec{a}') \\ 0 & \text{otherwise.} \end{cases} \tag{4.36}$$

Additionally, Wang and Sandholm introduced an algorithm that biases agents towards recently selected actions. They showed that this guarantees convergence to a coordinated optimal joint action for the virtual game and as a consequence also to a coordinated action for the underlying stochastic game.

To conclude this section, synthesising game-theoretic concepts with multi-agent reinforcement learning does not eschew the emergence of cooperative behaviour. Cooperative game theory allows for agents to enter joint agreements on which actions to perform. In many multi-agent scenarios multiple Nash equilibria exist, but one may only be Pareto optimal. This has profound implications for multi-agent systems, because any team strategy that is stuck in one Nash equilibrium could benefit all if only they could all agree in adopting different strategies.

4.4.2.2 Competitive Multi-agent Reinforcement Learning

Unlike cooperative multi-agent reinforcement learning, competitive settings are governed by rational or selfish agents. These are identified as pure local utility maximisers, which means that they only act in their own self-interest. As such, the employed reward function discerns their own contribution on the overall team effort. The selfish agent tries to maximise solely its contributions to the task. That does not preclude the fact that cooperative behaviour may emerge. Even non-aligned goals or employing competing goal structures within a decision making agents may bring about cooperative behaviour, if it suits the respective agents in achieving their goals. Therefore, some of the algorithms below, in particular [1, 2, 74], can be designed to work in cooperative settings as well.

Hu and Wellman [74] extended the familiar Q-learning algorithm to suit multi-agent general-sum stochastic games. The goal of their algorithm (Algorithm 4.9) is to find the best strategy relative to how other agents play in the game. In particular, a Nash equilibrium point is sought. The Nash equilibrium point is defined as:

Definition 4.7 (Nash Equilibrium Point). *A tuple of n policies $\langle \pi_1^*, \pi_2^*, \ldots, \pi_n^* \rangle$ such that for all $s \in \mathcal{S}$ and $i = 1, \ldots, n$,*

$$\forall_{\pi_i \in \Pi_i} v_i(s, \pi_1^*, \ldots, \pi_n^*) \geqslant v_i(s, \pi_1^*, \ldots, \pi_{i-1}^*, \pi_i, \pi_{i+1}^*, \ldots, \pi_n^*), \tag{4.37}$$

where $v_i(s, \pi_1^*, \ldots, \pi_n^*)$ is the total reward that agent i can expect to receive starting from state s and assuming that agents use policies $\langle \pi_1^*, \ldots, \pi_n^* \rangle$ Vidal [167].

Thus, the Nash equilibrium point covers a set of policies (one for each agent) and expresses that no one agent i will gain anything through unilaterally changing its policy. Hu and Wellman [74] showed that such an equilibrium point exists. Their NashQ-learning algorithm converges to the Nash equilibrium point under the assumptions that:

1. All agents use the NashQ-learning algorithm;

2. No agent has anything to lose, if the other agents change their policies;
3. A social welfare solution exists, where all agents receive highest possible reward.

Each agent maintains a Q-value function for each other agent in the population. So, the Q-value function for a multi-agent environment becomes $Q_i(s, \vec{a})$ indexed by the target agent. The traditional Q-value to determine future rewards is replaced by a NashQ-value that encodes the future Nash equilibrium payoffs. This requires knowledge about the opponents' actions and rewards.

This algorithm provably converges, albeit under certain restrictions mentioned above and which are only satisfied in a small class of problems. In single-player games (i.e., Markov decision processes), the NashQ function is a simple maximisation function which reduces to the Q-learning algorithm, which is known to converge to optimal values [170]. In zero-sum games, the NashQ function is a minimax function and the update rule reduces to minimax-Q, which is also known to converge to optimal values [90]. In this case this is the Nash equilibrium.

Algorithm 4.9: NashQ-Learning algorithm

```
$Q^j_t(s, \vec{a}) = 0 \; \forall \; s \in \mathcal{S}$ and $a_j \in \mathcal{A}_j, \; j = 1, \ldots, n$ arbitrarily
Initialise s
repeat
    foreach agent $i = 1, \ldots, n$ do
        Choose $a_i$ from $s_i$ using policy derived from $Q_i$ (e.g., $\epsilon$-greedy)
        Take action $a$, observe reward, $r_1, \ldots, r_n$, joint action, $a_1, \ldots, a_n$, and next state, $s'$
        foreach $Q_j \; \forall \; j = 1, \ldots, n$ do
            $NashQ_j(s') = \pi_1(s') \cdots \pi_n(s') Q_j(s')$
            $Q_j(s, \vec{a}) \leftarrow (1 - \alpha) Q_j(s, \vec{a}) + \alpha \left[r_j + \gamma NashQ_j(s')\right]$
            $s \leftarrow s'$
        end
    end
until for each step of the game
```

Littman [89] introduced the friend-or-foe Q-learning (FFQ) algorithm that relaxes the assumptions made by the NashQ-learning algorithm. If an agent is able to distinguish between a benevolent agent and an adversarial agent, respective Q-functions can be employed. More specifically, each agent maintains a set of k friends with action sets $X_1, \ldots, X_k$ and a set of l foes with action sets represented by $Y_1, \ldots, Y_l$. The update rule for friends is ordinary Q-learning, because the agent attempts to maximise the long-run discounted reward. Otherwise, the update rule is minimax-Q. FFQ learning converges to a Nash equilibrium.

Theorem 4.4 (FFQ Convergence). *Foe-Q learns values for a Nash equilibrium policy if the game has an adversarial equilibrium and a Friend-Q learns values for a Nash equilibrium policy of the game has a coordination equilibrium. This is true regardless of opponent behaviour [89].*

Algorithm 4.10: Friend-or-Foe Q-Learning algorithm

```
$Q^j_t(s, \vec{a}) = 0 \; \forall \; s \in \mathcal{S}$ and $a_j \in \mathcal{A}_j, \; j = 1, \ldots, n$ arbitrarily
Initialise s
**repeat**
    **foreach** *agent* $i = 1, \ldots, n$ **do**
        Choose $a_i$ from $s_i$ using policy derived from $Q_i$ (e.g., $\epsilon$-greedy)
        Take action $a$, observe reward, $r_1, \ldots, r_n$, joint action, $X_1, \ldots, X_k, Y_1, \ldots, Y_l$, and next state, $s'$
        $NashQ_i(s') = \max_{\pi \in \Pi(X_1 \times \cdots \times X_k)} \min_{y_i, \ldots, y_l \in Y_1 \times \cdots \times Y_l}$
            $\sum_{x_1, \ldots, x_k \in X_1 \times \cdots \times X_k} \pi_1(s') \cdots \pi_k(s') Q_i(s', x_1, \ldots, x_k, y_1, \ldots, y_l)$
        $Q_i(s, \vec{a}) \leftarrow (1 - \alpha) Q_i(s, \vec{a}) + \alpha [r_i + \gamma NashQ_i(s')]$
        $s \leftarrow s'$
    **end**
**until** *for each step of the game*
```

4.5 Summary

This chapter presented fundamental derivations of reinforcement learning algorithms. Understanding the theoretical underpinnings of the methods used is crucial, because convergence guarantees only apply in certain cases. Much of the literature on Markov Decision Processes (Section 4.1), value functions (Section 4.2), and elementary solution methods (Section 4.3) stem from the excellent book by Sutton and Barto [145]. Some of the fundamental concepts in reinforcement learning rely on maximising rewards based on current knowledge of the system (exploitation) and acting in an uncertain environment to gain additional knowledge about the system (exploration). The interplay of these decisions over time has a significant impact on the optimality of learning in the single-agent case. The convergence guarantees require that agents greedily exploit the current knowledge in the limit. As such, undirected exploration techniques, such as ϵ-greedy and Boltzmann policies, were presented that provide strong convergence guarantees.

Since this thesis models decentralised optimisation problems within the framework of multi-agent reinforcement learning, recent work in this area has been covered in Section 4.4. The goals of multi-agent systems is to achieve stability of the learning dynamics and adaptation to the dynamic behaviour of the opponents in the environment. As such, applying the familiar approaches from single-agent learning often fails, because the fundamental theorems are violated. Recently, research synthesised the game-theoretic framework with multi-agent systems to address some of the issues related to finding optimal joint actions. Interestingly, both cooperative and competitive scenarios can be designed to achieve optimal or near optimal behaviour in certain scenarios. Some of the intriguing approaches, such as biologically-inspired, swarm intelligence [48, 179], auctions [138, 167, 172], etc., were considered beyond the scope of this thesis.

— On the mountains of truth you can never climb in vain: either you will reach a point higher up today, or you will be training your powers so that you will be able to climb higher tomorrow.

Friedrich Nietzsche

5 Reinforcement Learning for Distributed Task Assignment Problems

Reinforcement Learning (RL) techniques have been at the forefront of intelligent agent research in a number of application domains, such as routing [119, 126, 132], robot control [26, 58, 147], and task allocation [148, 151, 152, 187]. Frequently, those domains are governed by decentralised processes which require learning agents to act independently in order to achieve a specific objective. As such, decentralised optimisation algorithms are frequently used to explicitly or implicitly decompose a large problem into manageable smaller parts which in orchestration contribute to the solution of the problem. Modelling these problems as decentralised Markov decision processes (Dec-MDPs) gives rise to complex continuous interaction dynamics. This is due to two forces which govern the nature of the learning dynamics. First, the interaction pattern represents a topology determining which agents can interact with each other. The non-linear effects can be attributed to the mutual influence on the learning behaviour itself, and these effects are further intensified, if the topology evolves over time. Second, the influx of events into the system (possibly from human sources or business interactions) render such a system thermodynamically open, constantly introducing changes to the operating conditions.

This decentralised setting and nonlinear behaviour often renders optimal planning solutions intractable. In fact Bernstein et al. showed that for the general Dec-MDP the complexity is NEXP-hard [20]. With some restrictive assumptions, such as only considering closed systems, static interaction sub-graphs, and independent learning algorithms, planning algorithms can be brought to bear to solve the optimisation task. However, real-world scenarios are both generally too large and cannot easily be forced into an idealised and controlled environment.

Additionally, Dec-MDPs imply a non-stationary environment which does not lend itself to asymptotically greedy policies. Therefore, large-scale simulation studies are necessary to study the learning behaviour of individual agents and the overall system dynamics online admitting constant adaptive learning cycles. This implies that the trade-off between exploration and exploitation needs to be carefully balanced.

The application domain in this thesis deals with distributed task assignment problems (DTAP), which arise in load-balancing applications, workflow systems or supply-chain management. More abstractly, this model resembles nested multi-armed bandits (network of bandits) with associative memory where the reward is distributed once all relevant arms have been pulled, i.e., rewards have a stochastic delay. Intelligent agents are employed at each server to gauge the performance of their neighbours by evaluating the response time of the sub-task completion. At each time unit, the agent makes a decision to minimise the response time, that is the time between issuance of a task request and its fulfillment, in the long run. In other words, the optimisation objective is placed on quality of service as perceived by the entity that issued the request, i.e., a client for external task requests or agents for internal sub-task requests.

The agent needs to take into account that one execution path might get saturated. Unless agents continuously adapt, delays will ensue and eventually cascade through the task network with the inevitable consequence of deteriorating system performance. The focus is placed on autonomous agents to solve the complex problem of sequentially assigning tasks in an environment.

As realistic scenarios are often modelled as complex adaptive systems, this thesis adopts an integrated approach to combine complex networks and queueing theory for the reinforcement learning environment with Markov decision processes for the reinforcement learning agent. Queueing theory deals with the mathematical study of queues, which occur whenever instantaneous demand exceeds the operational capacity to provide a service. Their general applicability and the growing interest in uncovering traffic patterns in complex networks suggest the combination of both research branches. The network evolution models introduced in Section 3.2 are used to establish a typical queueing network for use in simulations. This is a novel approach providing a framework which is exploited here in the simulation study of distributed task assignment. Once the network is established for a simulation run, it is considered fixed and does not change over time. This is unlike Abdallah and Lesser [1] who let their agents self-organise into an overlay network structure. However, they make an assumption about queueing stability (rate of service is larger than the rate of arriving tasks per agent) that is relaxed in the distributed task assignment problem discussed here, where queues may become temporarily unstable. The agent's learning objective of minimising the response times implies that instability will not go unnoticed and the agent will adapt to it accordingly, if a continuous learning policy is employed. This can be seen from the load-to-delay function of server queues in equation (3.8). An unstable queue exhibits a utilisation (or load) of larger than 100% which increases the delay and ultimately affects the response time of a task.

The reinforcement learning agent interface is formalised into a decentralised Markov decision process in Section 5.2. The MDP formalism takes into account the reinforcement learning environment as a directed acyclic graph representation of the hierarchical task structure. Additionally, the MDP formalism supports cooperative behaviour by means of a global reward structure that fairly assigns credit to the individual agents involved in completing a task. This avoids selfish behaviour among the agents. Instead the reward structure motivates agents to act in concert to accomplish the common goal of minimising the response times. However, given the asynchronous interactions among the agents with stochastic delayed feedback, it is no longer trivial to assume a decentralised control algorithm will accomplish its goal. In abstract terms an agent's actions are based on its knowledge about the state. In fact, in most realistic problem settings the knowledge is based on local state information, which implies only partial knowledge of the system. This can be problematic, because an agent relies on delayed feedback to reinforce existing knowledge. If one agent downstream fails to complete a task, it may go unnoticed when acknowledgements and/or timeouts are absent from the system design and consequently the feedback never arrives. Assuming this is an agent's preferred action selection, then missing reinforcement signals will not alter knowledge about a degradation of performance and the agent will continue to select this action. Essentially, the cause and effect are removed from action selection, which is the fundamental assumption that guarantees learning. This general problem is well-known in distributed systems as the "Byzantine generals problem" [85] and is very hard to solve accurately and efficiently. Fagin et al. [52, Chapter 10] discuss Byzantine failures with respect to multi-agent systems. This thesis does not deal with this problem, because the queueing network is not modelled to include faulty service stations.

Section 5.1 presents related works. The related works cover recent machine learning techniques that avoid reliance on these difficult-to-tune parameters, such as the learning rate or balancing exploration and exploitation. This section also comprises specific modelling approaches for distributed task assignment and distributed resource allocation problems. In this thesis, task assignment and resource allocation are considered separately although mathematical solution methods provide appropriate abstractions to deal with both type of problems. Distributed task assignment generally deals with allocating a number of tasks to given resources to maximise some measure of utility. In contrast, resource allocation deals with the reverse problem of allocating resources to a given (expected) number of tasks. One assumption is made on the nature of how tasks are handled in this thesis, in that tasks arrive sequentially in the task network, rather than in bulk.

Section 5.2 formalises the distributed task assignment problem using the DEC-MDP framework introduced in Definition 4.4 with extensions to account for the specifics of the queueing-theoretic abstractions for the task hierarchy. Further, two solution concepts are investigated in this chapter: one that has its roots in game theory and one that applies single-agent learning into a multi-agent setting. More specifically, the SARSA(0) reinforcement learning agents are endowed with a weighted policy learner (WPL) or an ϵ-greedy policy representing the two solution concepts respectively [2, 145]. Further, the state space is generalised with a standard back-propagation neural network function approximator to estimate the Q-function that maps state signals to action-values.

The results are presented in Chapter 6, which evaluates the DEC-MDP model with respect to the underlying network topology to investigate the impact of all learning parameters on the objective of the optimisation algorithm. An emphasis is placed on the empirical comparison of two policies: the weighted policy learner and the ϵ-greedy policies. The need for understanding and analysing the behaviour of these multi-agent systems is compelling. However, appropriate tools need to be adopted that provide statistically meaningful results. In particular, in order to analyse the influence of the reinforcement learning parameters (learning rate and momentum of the back-propagation neural network, and the discount factor of SARSA(0)) to the total average time of events in the system, a response surface methodology (RSM) is employed (see Chapter 2 for a thorough introduction). RSM facilitates a computationally efficient way of performing simulation studies in regions with highest uncertainty. The simulation output is assumed to vary continuously in the input space (which is mostly the case for computer experiments), so that a smooth interpolation function can be fitted to the data. This methodology treats the simulation code as a black box, whereby the uncertainty in the response surface is evaluated using a mean-squared error criterion of a set of validation locations. That way the behaviour of complex systems modelled by numerical simulation can be efficiently understood, which is similar to the approach of active learning for directed exploration [25].

5.1 Related Works

The related works of this chapter covers machine learning methods that reduce the reliance on difficult-to-tune learning parameters to avoid the dilemma of balancing exploitation and exploration in Section 5.1.1. Then some approaches covering resource-allocation are presented in Section 5.1.2. Though similar to task assignment problems, resource allocation deals with the allocation of resources to facilitate the computation of tasks. In that sense, it is a reverse problem to task assignment, which is covered in Section 5.1.3. Task assignment, on the other hand, deals with assigning tasks to available (static) resources.

Chevaleyre et al. [31] presented a survey on issues in multi-agent resource allocation with a focus on the interface of computer science and economics. They provide a tentative definition for multi-agent resource allocation as

Definition 5.1 (Multi-agent Resource Allocation). *Multi-agent Resource Allocation is the process of distributing a number of items amongst a number of agents [31].*

This definition is quite general and neither indicate what the items to be distributed are, nor why the items are distributed. For the purpose of this thesis, this definition is too broad, because it does not allow a distinction between assigning a task to available resources versus allocating resources to accomplish given tasks. However, both kinds of problems can be formalised into a common optimisation framework.

For example, grid or cloud computing provides ample scope for multi-agent resource allocation (for a short comparison between grid and cloud computing see Foster et al. [55]). Their use alludes to the deployment of services, or more generally computing tasks, with on-demand capabilities. As such, a large pool of resources can be allotted dynamically to the computing needs at any given time. Multi-agent systems provide a great promise to unify resource allocation, payment, and job processing. In particular, as grid/cloud computing systems grow in size, central management becomes prohibitively expensive.

Manufacturing systems on the other hand resemble more the general problem of task assignment (or task scheduling), where one task is part of a production plan with certain dependencies between the tasks. Because manufacturing systems exist with real-time constraints optimal solutions are often elusive. Rather, feasible solutions that are stable are sought to avoid interruptions.

5.1.1 Learning Theory

To find globally optimal learning parameters a Kriging response surface modelling technique was used [42]. Kriging is a prediction method of continuous-valued outputs, which is generally used to interpolate data. Traditionally being used in the geostatistics domain since the 1960s, it has increasingly gained in importance in the machine learning community designated as Gaussian Processes (GP) [120]. The iterative nature of learning a predictive model has been exploited to adaptively select samples to improve the prediction (also known as adaptive sampling or active learning) [25, 35, 68].

While active learning can be used off-line to establish and improve a metamodel about the system behaviour, recent advances in the reinforcement learning literature introduced methods to avoid difficult-to-tune parameters, such as the learning rate or exploitation-exploration schemes.

In a game-theoretic framework Tuyls et al. mathematically derived a connection between the exploitation-exploration scheme from Q-learning with Boltzmann exploration and the selection-mutation mechanisms from evolutionary game theory [158] based on 2-player games. Related to their work, Gomes and Kowalczyk analysed Q-learning in typical 2-player games with the ϵ-greedy exploration mechanism [65]. They derived a system of difference equations to calculate the expected evolution of the Q-values and the expected behaviour of the agents.

Zhang et al. present a direct policy Fair Action Learning (FAL) search technique [187]. Similarly to Generalised Infinitesimal Gradient Ascent WoLF (GIGA-WoLF) [23], FAL approximates the policy gradient of each state-action pair with the difference of the expected Q-value on that state and its Q-value. As such it learns a stochastic policy that increases the probability of actions receiving a higher reward then the current average. Consequently, FAL will converge to a fair policy reflecting the expected reward for all actions and states. However, if one action is always more favourable than the other ones, FAL will converge to a deterministic policy, which is not always desirable. The weighted policy learner (WPL) from [1, 2] algorithm addresses this issue to ensure that all actions have a minimum probability of being selected.

Algorithm 5.1 illustrates the direct stochastic WPL policy. This policy ensures that no action probabilities converge to a deterministic policy using: $\Pi_X(x) = \arg\min_{x':\text{valid}(x')}(x - x')$, which returns a policy that is closest to x and satisfying the constraints that it sums to 1 and action probabilities are greater than a given parameter ϵ. This operation performs a projection of the policy update, $\pi + \Delta\pi$, onto the probabilistic simplex X. ζ denotes the update rate and is a free parameter of WPL.

The weighted policy learning (WPL) algorithm by Abdallah and Lesser has been applied in distributed task allocation and showed interesting results [1], because it converges to a stable stochastic policy. The WPL algorithm was designed with the need of quickly converging to a stable stochastic policy. This is achieved by performing a gradient ascent towards a stable policy and slow down learning gradually for as long as the gradient does not change direction and learn fastest when the gradient changes direction. This is in contrast to "Win or Learn Fast" (WoLF) algorithms, such as GIGA-WoLF [23], which use approximations to determine when an agent is moving towards or away from a Nash Equilibrium.

Algorithm 5.1 illustrates the weighted policy learner (WPL) [1, 2]. The last step of the Algorithm 5.1 performs an Euclidean projection,

$$\Pi_X(\mathbf{x}) = \arg\min_{\mathbf{y}} \{\|\mathbf{x} - \mathbf{y}\|_2 \mid \mathbf{y} \in X\},$$

Algorithm 5.1: WPL: Weighted Policy Learner

```
Input: Let $Q(s, a)$ be the expected reward for executing action $a$ in state $s$
$\bar{Q} = \sum_{a \in A} \pi(a) Q(s, a)$
foreach action $a \in A$ do
    $\Delta(a) \leftarrow Q(s, a) - \bar{Q}$
    if $\Delta(a) > 0$ then $\Delta(a) \leftarrow \Delta(a)(1 - \pi(a))$
    else $\Delta(a) \leftarrow \Delta(a)(\pi(a))$
end
$\pi \leftarrow \Pi_X(\pi + \zeta\Delta)$
Output: A new policy $\pi$
```

of the policy update, $\pi + \Delta\pi$, onto the simplex X subject to $\sum_{a \in A}^{n} \pi(a) = 1$ and $\pi(a) \geqslant 0$, which can be implemented using a linear time algorithm from Duchi et al. [50]. Thereafter, the vector is scaled, such that no action probabilities converge to a deterministic policy. More specifically, all action probabilities are determined to be greater than ϵ, denoted by $\pi \succeq \epsilon$.

Learning algorithms in continuous state spaces are often faced with the challenge of how to partition the state space to achieve an accurate mapping from state to actions and compact representation in memory at the same time. A priori discretisation of a state space can be either coarse which requires less experience to learn well or fine which requires a higher level of experience. The effect of the scheme employed has a huge impact on the general learning performance. To overcome the difficulties associated with a priori discretisation of continuous state spaces Nouri and Littman [112] developed a regression-based approach to use a multi-resolution exploration technique in continuous spaces based on regression trees as function approximators with variable degrees of discretisation over time and the state space. This way, regions of high uncertainty are deemed candidates for more exploration, where the regions are identified within the framework of regression trees. Their approach is a variation of the kd-tree structure that organises points of the k-dimensional state space. A new state vector is entered in the appropriate leaf-node that represents the encompassing region of this vector. As such a leaf node represents a number of state vectors. If a certain threshold is reached then the node is split into two half-regions along one dimension chosen by round-robin. The purpose of the regression tree is to provide a measure of uncertainty for certain regions of a state space. That way, exploration can be guided towards those regions that require more experience. Also, as more states are visited the regression tree evolves a variable degree of generalisation.

More specifically, the uncertainty about a function approximation at a given state $\mathbf{s} \in \mathcal{S}$ is quantified using a real-valued measure of the form $0 \leqslant \text{knownness}(s) \leqslant 1$, where the two extreme values (0 and 1) represent the degenerate cases of no knowledge and complete knowledge. The splitting rule is determined by a parameter ν. The size, $l.size$, of the region at node l in the regression tree is determined by its ℓ_∞-Norm and the number of points residing inside this node is given as $l.count$. Given n points in total represented within the regression tree, μ denotes a normalising size of the tree that allocates ν/k points inside each cell k over a hypothetical uniform discretisation of the space. μ is then given as $\mu = \frac{1}{\lfloor nk/\nu \rfloor}$, where the state space is normalised as $\|s\|_\infty \leqslant 1$. Then, upon finding the leaf node that contains a state s, the knownness value can computed according to

$$\text{knownness}(s) = \min\left(1, \frac{l(s).\mathtt{count}}{\nu} * \frac{\mu}{l(s).\mathtt{size}}\right). \tag{5.1}$$

Equation (5.1) balances the coverage ratio in terms of the number of points within a region with the ratio of region of interest with respect to the hypothetical uniform discretisation. As more knowledge about a region is exploited more points are added to that region which encourages finer discretisation. Conversely, regions of high uncertainty can be determined that would benefit from further sampling.

Another way of addressing exploration in reinforcement learning algorithms in MDPs with discrete state and action spaces is to apply a Bayesian approach to quantifying the uncertainty over MDPs. In Asmuth et al. [10], Bayesian sampling is used which maintains the uncertainty in the form of a posterior probability over models. The K models are sampled from the posterior whenever the number of transitions from a state-action pair reaches a specified threshold B. This is similar to RMAX, where a state-action pair is considered known, if it has been observed B times. After sampling, a complete MDP is constructed with the same state space and an augmented action space of $K|\mathcal{A}|$ actions, where action $a_{i,j}$ from model i corresponds to the jth action of the MDP. The transition and reward functions are taken directly from the models. They showed that their algorithm takes near-optimal actions and that it explores better than ϵ-greedy and Boltzmann exploration.

The least-squares policy iteration (LSPI) [84] is a state of the art model-free and off-policy reinforcement algorithm, that has been extended by Li et al. to combine LSPI with the RMAX exploration technique [87]. LSPI does not rely on an explicit learning rate tuning parameter and is guaranteed to converge. Originally being an off-policy learning approach, LSPI is combined with the sample-efficient RMAX exploration technique to be used for on-policy learning agents. The efficiency of the algorithm is demonstrated on well-known standard benchmark problems.

5.1.2 Resource Allocation

Chow and Kwok [32] introduced a software-engineering approach to balancing load for distributed multi-agent computing, where agents represent brokers between a client request and services provided through the service network. The service network spans multiple agents distributed over multiple computing nodes. The aim of their research is to minimise the variance of the load among all computing nodes in the cluster taking into account the communication pattern among the agents and the computation demands of the individual agents, which in effect minimises the average response time of accomplishing the task requests.

The hierarchical structure is depicted in Figure 5.1. The "work agents" are responsible for the computation of a particular task. In order to accomplish a task, "work agents" communicate with other agents in the cluster. Similarly to the topological structure of the task network presented in this thesis, the interaction graph among the agents is fixed. It essentially presents an overlay network structure, which self-organises into agent-assignments to particular hosts, in order to minimise the variance of the load in the cluster. For that purpose, each "work agent" has a credit value assigned to

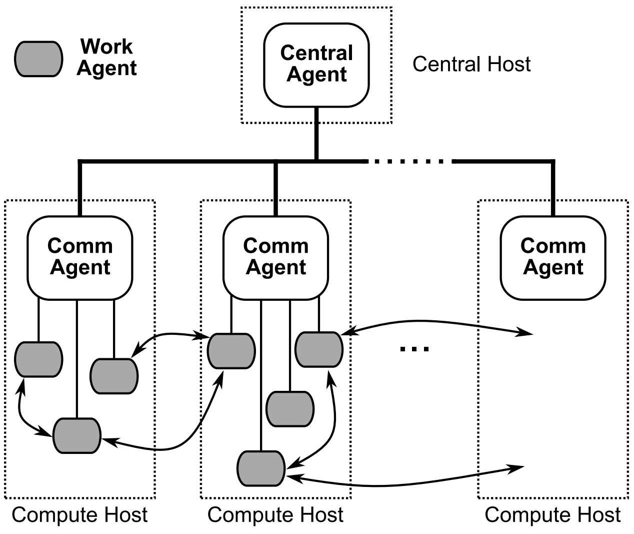


Figure 5.1: Load-balancing Agent Interaction

it, which is defined as

$$C_i = -x_1 w_i + x_2 h_i - x_3 g_i, \tag{5.2}$$

where x_1, x_2, x_3 are positive application-specific coefficients that weight the computational load, w_i, the intra-host communication load, h_i, and the inter-host communication load, g_i, by agent i. This credit value expresses the affinity of a "work agent" to stay at the current host. The higher this value the higher the affinity to stay at the same host. While the computational load and the inter-host communication contribute negatively to the credit value, inter-host communication has a positive effect. This way, the agent strives to organise itself around agents it communicates with most. However, if the load the agent contributes to a host becomes large or the communication pattern changes, then the agent increases the pressure to move to a different host. The load information is gathered and the clocks of the hosts are synchronised periodically. Each "work agent" aggregates the load, $u_i = h_i + g_i$, and communicates that to the communication agent which is co-located with each host. The communication agent reports the load values to a central authority that decides on a migration plan for the agents. A suitable candidate for migration is a "work agent" that exhibits an aggregate load above a specified load threshold and the one with the lowest credit value on a given host and the respective target host is determined based on the dominant inter-host communication of the candidate agent. Once selected for migration, a copy semantics is employed, meaning that a new agent is initialised at the target host and all state information is copied to the new agent before the old one is killed.

By migrating incrementally, the load within a cluster is balanced, taking into account the continuous dynamics of the interactions in specific time intervals. This migration strategy takes advantage of the community structure of the interaction topology and attempts to co-locate each community near each other to reduce the inter-communication overhead. Chow and Kwok [32] demonstrated that better load characteristics are attained compared to a workload-based load-balancing strategy [136].

Unlike agents in this thesis, the agents employed in Chow and Kwok [32] are not based on learning algorithms. Rather, their autonomy comes about in their ability to migrate based on load characteristics which is measured periodically in given time intervals. Additionally, the central agent that determines the migration plan is rule-based. The simplicity of this approach and its low runtime overhead are among the advantages, while a central authority presents a bottleneck for large systems and additionally a single point of failure.

In a series of articles, Tesauro et al. presented hybrid reinforcement learning using the $\mathsf{SARSA}(0)$ method with an ϵ-policy for resource allocation of servers in a prototype data centre [148, 151, 152] depicted in Figure 5.2.

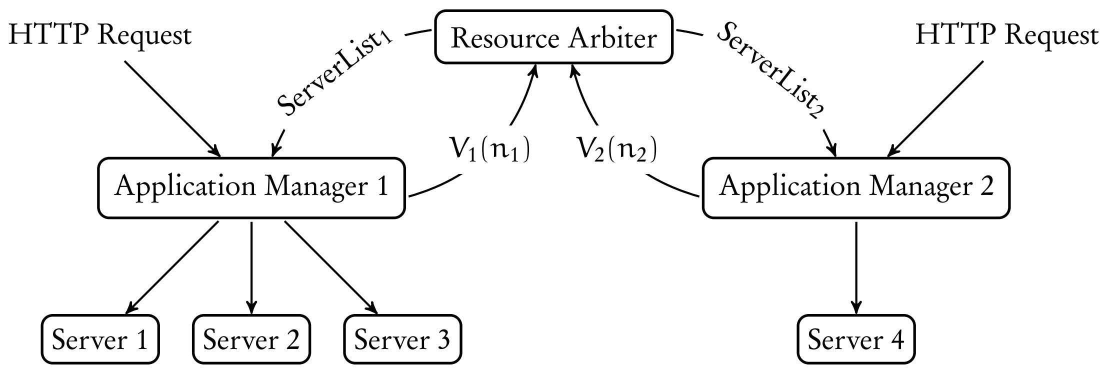


Figure 5.2: Prototype Data Centre

In the prototype data centre example the resource arbiter is responsible for allocating identical servers to the application managers. The application managers represent particular web applications being deployed in the data centre and each application manager learns a business utility curve governed by service level agreements and current queueing state variables. Tesauro et al. employed neural networks to generalise the state space including queueing-theoretic performance metrics to compute the business utility curve. This utility curve is reported to the resource arbiter which decides upon an optimal server allocation scheme for the available application managers. To avoid poor initial performance, the neural network is trained in batch mode for an initial time period alongside a good external policy derived from theoretical queueing models. In their scenario learning may continue online in batch mode by collecting data while the previously improved policy, π', is employed, and then train a further improved policy, π''.

Modelling decentralised control as discussed in this thesis and the approach presented by Tesauro et al. are very similar, in that the motivation for using reinforcement learning to deal with dynamically changing environments and the learning methods employed overlap. Additionally, the approach presented in this thesis and the approach for autonomous computing in data centres rely on the framework of queueing theory to provide the state variables for the reinforcement learning method.

The main difference lies in the constraints given by the scenario. While a data centre is controlled by a single cooperation, optimising the business utility may encompass centralised solutions. For scalability reasons Tesauro decomposed the reinforcement learning module and co-located those with the application manager [150]. A central authority correlates the business utility curves presented by each application manager and allocates servers for each appropriately. As a result, the topology of this scenario is star-shaped, which may still pose scalability constraints for a large number of application managers. This option is not given in naturally heterogeneous environments, where the participating entities are inherently autonomous such as in the application domain covered in this thesis. Consequently, heterogeneous environments require careful design of the decentralised control algorithms to avoid selfish behaviour by the agents. This implies that no central authority can be put in place to control the decentralised optimisation algorithm.

As such a simulation-based approach is employed to scale up the number of interacting agents using a social network evolution model presented in [41] to analyse globally optimal learning parameters [42]. Additionally, there is a need to investigate stable stochastic policies that avoid greedy behaviours, such as the weighted policy learner introduced by Abdallah and Lesser [2].

5.1.3 Distributed Task Assignment

From a practical perspective, distributed task assignment covers e-Commerce applications that are composed of autonomous and potentially heterogeneous services across the boundaries of independent provider organisations. With such cross-organisational integration of business processes, synergies can be fostered and competitiveness of the organisations involved can potentially improve. In an abstract way, the structure of the task network in this thesis identifies autonomous entities that may represent organisations and their public service interface. Internally, however, each public service can be composed of additional private services, such as accounting and customer relationship management. As such, an autonomous entity represents an intra-organisational business process. The public view of it is identified with basic queueing properties, like a service rate and an arrival rate of external requests. Given a technical or semantic description of a service and a suitable language to describe the interaction between services, problems such as optimising a workflow with respect to the capabilities of the services arise. Artificial intelligence planning techniques, such as the hierarchical task network planner SHOP2, can be used to produce a sequence of actions defining a workflow that accomplishes a complex task [183]. However, in the face of choice, i.e., several services fulfill the same functionality, an optimisation algorithm can take into account the qualitative difference between those alternative services. Kritikos and Plexousakis use mixed-integer programming to solve this optimisation problem given a set of constraints on Quality of Service (QoS) metrics [82]. These techniques require strong consensus between the interacting services and therefore do not take into account the dynamic evolution of services over time. For example the QoS characteristics of a service might improve or deteriorate over time. Unless, an online optimisation algorithm is employed, changes in operating conditions will be neglected.

Schaerf et al. [134] investigated the effects of multi-agent reinforcement learning in the context of load-balancing problems relying purely on local information. Their system model encompasses a number of agents that receive tasks from the environment and assign those tasks to available resources. Their setup exhibits a separation between agent and resources, such that the agents are autonomous in their decision-making and the resources are passive only executing the tasks assigned to them. The learning task of the agents is then to perform trial-and-error actions with the environment and observe the cause and effect of the task assignment.

Definition 5.2 (Multi-agent Multi-resource Stochastic System)**.** *A multi-agent multi-resource stochastic system is a 6-tuple* $\langle \mathcal{A}, \mathcal{R}, \mathcal{P}, \mathcal{D}, \mathcal{C}, SR \rangle$*, where* $\mathcal{A} = \{a_1, \dots, a_N\}$ *is a set of agents,* $\mathcal{R} = \{r_1, \dots, r_M\}$ *is a set of resources,* $\mathcal{P} : \mathcal{A} \times \mathcal{N} \to [0, 1]$ *is a task submission function,* $\mathcal{D} : \mathcal{A} \times \mathcal{N} \to \mathbb{R}$ *is a probabilistic job size function,* $\mathcal{C} : \mathcal{R} \times \mathcal{N} \to \mathbb{R}$ *is a probabilistic capacity function, and* SR *is a resource selection rule [134].*

Each resource is assigned a non-homogeneous capacity, which changes over time determined by the function $\mathcal{C}$. An agent can either be idle or busy, where tasks can only be submitted according to a probability $\mathcal{P}$, if the agent is idle. Moreover, tasks are assigned a size determined by function $\mathcal{D}$. As new tasks are generated an agent selects one of the available resources according to a selection rule SR. The learning objective is to distribute the load evenly among the available resources. When an agent assigns a task to a resource its state changes to busy and waits until it is notified that the tasks is finished. Resources can operate infinitely many tasks, i.e., there is no queue. However, as more tasks are assigned to one resource, its service deteriorates. Resources compute all tasks in parallel, where the capacity is equally distributed to the tasks and the size of the task determines how long it takes to be finished.

Their learning model essentially is stateless, which means it does not correlate some state information with the feedback received. The experience of past task assignments is maintained in an efficiency vector which represents all available resources denoted by $\boldsymbol{ee}_i$ of agent i. The only other information an agent maintains is the total number of tasks completed by each resource, $\mathbf{jd}_i$. As in typical reinforcement learning tasks, the efficiency estimation is updated after receiving a feedback tuple $(r, t_{\text{start}}, t_{\text{stop}}, S)$, where t_{start} and t_{stop} are the start time of the task assignment and the end time of task completion respectively, and S is the size of the job. The update rule follows as

$$\boldsymbol{ee}_i(r) \leftarrow WT + (1 - W)\boldsymbol{ee}_i(r) \tag{5.3}$$

$$T = (t_{\text{stop}} - t_{\text{end}})/S \tag{5.4}$$

$$W = w + (1 - w)\mathbf{jd}_i(r), \tag{5.5}$$

where w is a real-valued weight factor. The adaptive selection policy SR is given as probabilistic function

$$\mathbf{pd}'_i(r) \leftarrow \begin{cases} \boldsymbol{ee}_i(r)^{-n}, & \text{if } \mathbf{jd}_i(r) > 0 \\ \mathbb{E}[\boldsymbol{ee}_i(r)]^{-n}, & \text{if } \mathbf{jd}_i(r) = 0, \end{cases} \tag{5.6}$$

where n is a positive real-valued parameter and $\mathbb{E}[\boldsymbol{ee}_i]$ represents the average of the values in the efficiency vector $\boldsymbol{ee}_i$ over all resources satisfying $\mathbf{jd}_i \succ 0$. Normalising $\mathbf{pd}'_i(r)$ according to

$$\mathbf{pd}_i(r) = \mathbf{pd}'_i(r) / \sum_{r' \in \mathcal{R}} \mathbf{pd}'_i(r'),$$

the function $\mathbf{pd}_i$ introduces a bias whose strength is determined by the factor n. The larger the value of n, the stronger the bias. Consequently, very large values for n will render the policy deterministic, which is often not desirable in multi-agent systems, because multi-agent systems are not stationary.

They showed that it is feasible to achieve adaptive load-balancing solely based on local feedback information. Similar results, albeit with additional local states, are presented in Section 6.2.2 and Section 6.3.2. Schaerf et al. [134] show that adaptive SR is able to learn changing conditions, such as load and capacity changes. In particular they observed parallels to the exploration-exploitation dilemma with the interplay of the parameters n and w. High exploratory activities (low n) requires a greater weight to more recent experience (high w). Also, they showed that selfish behaviour of a single agent with all others using an adaptive SR gain from their exploitative actions, while selfish behaviour by all agents leads to the "Tragedy of the Commons" where nobody gains [69]. Interestingly, employing communication to share the efficiency vectors among neighbouring agents does not achieve good results, because communicating agents settle to best response actions. Consequently, the agents become selfish in exploiting good resources and avoiding bad resources. This way, the load is not evenly distributed and therefore this simple communication scheme has a detrimental affect on the group of agents.

These observations are investigated in this thesis as well. It is expected that the "Tragedy of the Commons" does not play such a vital role, because the resource spaces agents have access to do not necessarily overlap. This is due to the explicit interaction structure imposed in this thesis. Moreover, it is crucial that agents incessantly adapt to the changes in the environment and therefore best response actions are not considered in this thesis.

Verbeeck et al. [164] investigated multi-agent reinforcement learning in settings where multiple pure Nash equilibria exist. One agent's Nash equilibrium strategy may provide the agent with a maximum payoff, while the other agent receives a suboptimal payoff, and vice versa for the other pure Nash equilibria. In these scenarios, agents often employ a mixed strategy which may leave both agents with a smaller payoff than with the pure Nash equilibrium strategies. This is the case when there is a chance that both agents do not coordinate on a task with the consequence of reduced payoffs. One proposed solution is to cycle between the pure Nash equilibria requiring some communication among the agents. This can be achieved with periodic policies, where the agents not only care about their own payoff, but also pay attention to the other agents' payoffs. These kind of agents belong to a homo egualis society, where agents are willing to give up their maximum payoff to the benefit of the other agents in the system fostering a sense of equality. To achieve this, agents periodically exchange their performance characteristics. The agent with the maximum payoff evaluates giving up its best action to give the other agents an opportunity to retrieve higher payoffs. Verbeeck et al. [164] investigated a simple load-balancing scenario depicted in Figure 5.3. Both agents, A1 and A2, have access to a common public resource, R2, and each have access to a private one, R1 and R3, respectively. The public resource is free, while the private resources have a cost associated with them. The agents stochastically assign a task to the respective resources they have access to which is then serviced according to an exponential distribution. Consequently, the reward of each task assignment is delayed. Their selection strategy have to be based on the judgement of the other agents behaviour and the load and capacity characteristics of the resources. If both agents incessantly assign tasks to the public resources, both their performance will deteriorate.

The scenario shown in Figure 5.3 exhibits two pure Nash equilibria: agent A1 always chooses

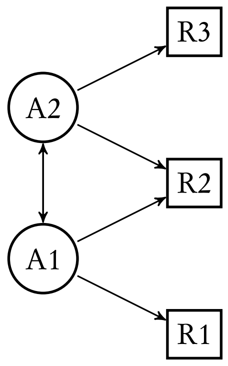


Figure 5.3: A simple scheduling game

the public resource R2, while agent A2 chooses its private resource, and vice versa. So, both agents have to gauge the queueing characteristics of the available resources and adapt their behaviour accordingly. Allowing the agents to communicate with each other a periodic policy can be agreed upon that gives a fair share of the public resource to all agents. Their results show that in two players scheduling games a mixed strategy without communication is in fact optimal. In this case the mixed Nash equilibrium is easily calculated, which is not the case anymore for games with more than two players. Considering more than two players, a good probability distribution over their action was found by agents playing periodic policies offline and following the learnt probabilities online.

Periodic policies are further investigated in [165] to reach Pareto optimal strategies in multi-agent common interest games without communication and in conflicting interest games with communication. Similar to Verbeeck et al. [164], learning is organised in two phases. The first phase is dominated by selfish exploration, where agents act as utility maximisers, and the second phase allows each agent to remove one action from their action spaces. The reduced joint action space of the multi-agent system is then explored selfishly by all agents before the next phase of synchronisation occurs. In the synchronisation phase, each agent normalises the Q-values for the respective actions into probability space. Then the average reward for the whole learning process is broadcast to the other agents as well as the average payoff of the private action that the agent converged to in the previous exploration phase. The agent with the highest cumulative payoff and the highest average payoff excludes the action it converged to in the exploration phase to allow the other agents to find more rewarding policies. This cycle repeats to allow the other agents to catch up with the best performing agent yielding periodic policies that alternate between the pure Nash equlibria. The objective of the load-balancing scenario presented in [165] is to evenly distribute tasks to slave nodes in parallel applications to maximise the efficiency. In particular the slave nodes request a task of a certain size from a master node. The master-slave architecture of parallel applications presents a communication bottleneck, because the master is the node that distributes the tasks according to the slave's request. If the requests for tasks arrive at a higher rate than the master can cope with, the slaves will become idle and the efficiency of the system decreases. With exploring selfish reinforcement learning, agents in the exploration phase converge to a particular task size. The immediate feedback signifies the inverse blocking time which is the waiting time of a request from a slave being serviced by the master node. At the end of each exploration phase the master node sends an average reward, calculated as the total computation time for the tasks that were processed in parallel, to all slaves. This average reward is used to update the Q-value of the converged action, which is subsequently removed from the action space. The cycle of exploration/synchronisation repeats. Verbeeck et al. [165] showed that the slave nodes specialise in certain task sizes with an even computation pattern that maximises the efficiency of the parallel application.

Montresor et al. [103] are inspired by ant colonies to address the limitations of master-slave architectures, such as those of volunteer computing [115] with the Seti@Home project being a prominent example. Volunteer computing are characterised by massive data sets which are divided into a large number of chunks that are distributed by a master node to interested peers (slave nodes). The slave nodes conduct a computationally expensive analysis of the data chunks and report the results. Messor provides a computing infrastructure that is inherently decentralised with the intention of balancing the load in P2P systems allowing arbitrary peers to initiate computational tasks. The biological metaphor that provides a basis for the P2P load-balancing application are Messor ants [22] that group objects in their environment into piles to clean up their nests. As such, agents are modelled after the behaviour of the ants, which are mostly governed by simple rules without a central authority dictating how agents should behave. Through their interaction seemingly intelligent global behaviour can be observed, for instance in the case of Messor evenly distributed tasks in a P2P environment. Swarm intelligence systems are additionally accompanied by characteristics such as robustness, which makes them particularly suitable for dynamic P2P systems where the load characteristics can suddenly change by peers joining or leaving. From an engineering perspective the bottom-up approach to designing a decentralised solution is very attractive, because a focus is placed on simple rules that govern the interaction of the agents in the system and exhibit an emergent character that solves the overall global objective. Messor is based on a middleware called Anthill [12] that abstracts the ant colony metaphor to provide interfaces for distributed computing applications. An Anthill model consists of nests that are associated with a peer to manage discovery of neighbouring nests and are associated with local applications interfacing with the user. An application issues a request to the nest, which in turn is confederated with a service carried out by one or many ants. The basic idea behind Messor ants is to follow simple rules: (1) when an ant is not carrying an object, it wanders about randomly until it encounters an object and picks it up; (2) when an ant is carrying an object, the ant drops it only after having wandered about randomly "for a while" without encountering any other objects [103]. The object in the context of Messor is a computational task to be distributed. Ants explore the environment to find nests that are overloaded. The nest identifier is recorded and when an underloaded nest is found in the environment, the nest identifier of the overloaded nest is transmitted, upon which the target nest downloads tasks from the overloaded nest. The mechanisms by which ants determine the load of a nest is entirely based on local information gathered by the ant in the environment. Under- or overloaded nests are determined based on an average load that the ant calculates based on its past encounters with the nests in the environment, where load is identified as the number of tasks in the queue of a nest. To introduce a more targeted search mechanism, ants store load information of past visited nests in a local storage facility of the nests it visits. Consequently, the search for over- and underloaded nests can be directed to those regions in the domain that are of most interest. To avoid suboptimal exploitative actions, a probability of exploration is introduced that avoids a bias towards a subset of nests. Montresor et al. [103] demonstrated that an initial ring overlay network of nests, self-organises into a different topology facilitating an even distribution of the load by the ants.

Another approach to modelling decentralised systems is collective intelligence (COIN) [156, 157, 179, 181]. This approach concentrates on the behavioural viewpoint of individual agents and their effect on the system, i.e., the collective. Agents in a multi-agent setting are regarded as utility maximises that are primarily concerned with their own performance, the private utility. However, carefully crafting utility functions brings about a win-win situation for the agent and the system as a whole. Each agent can be viewed as though it is striving to maximise its own private utility function while at the same time also maximising the world utility of the collective. The engineering discipline is based on division of labour, where the system is sub-divided into smaller parts that each can be solved efficiently. Moreover, no domain knowledge, i.e., no knowledge concerning the dynamics of the environment is assumed. The solution of the decentralised optimisation problem is brought about in a bottom-up fashion and therefore encompasses swarm intelligence solutions. Wolpert et al. [181] modelled Internet routing using COIN concentrating on the core concepts: subworlds, factored systems, constraint-alignment, and the wonderful-life utility function.

An agent, η, at a discrete time t is characterised as having a vector representing internal state and externally visible actions as $\underline{\zeta}_{\eta,t}$. The state of the collective capturing the time dynamics as well is denoted as $\underline{\zeta}$. The world utility is a function of this state as $G(\underline{\zeta})$, which is potentially not expressible as a discounted sum of the rewards. The division of the collective into subworlds, ω, constitutes a number of agents that together solve a partial problem and share a subworld utility $g_\omega(\underline{\zeta})$. A system of collectives is constrained-aligned, if the agents in separate subworlds do not affect each other directly. Further, a subworld-factored system is one whose subworld utilities $g_\omega(\underline{\zeta})$ increase only if it also increases the world utility $G(\underline{\zeta})$. However, negative side-effects of utility maximisation in one subworld may be experienced in another subworld. This allows for mathematically convenient division of labour, where their utilities are aligned to foster collaborative behaviour. It can be proven that optimal behaviour in a subworld with respect to the other agents within it, yield a global behaviour that corresponds to the agents reaching a Nash equilibrium [177]. Additionally, there can be no “Tragedy of the Commons” in a subworld-factored system [156].

Let $CL_\omega(\underline{\zeta})$ be a function that adjusts the states of all agents in subworld ω across all time to an arbitrary fixed value. In [181], the fixed value is 0. The wonderful life subworld utility (WLU) is then given as

$$g_\omega(\underline{\zeta}) \equiv G(\underline{\zeta}) - G(CL_\omega(\underline{\zeta})). \tag{5.7}$$

Essentially, the wonderful life subworld utility provides a measure of the change of the world utility with the subworld ω removed from the system. Additionally, it provides a mathematical trick that exhibits a dynamics-independent view of the system. Informally, this means that the WLU can be evaluated without inferring how the system would have evolved if an agent η's state were set to 0 at time t and the system evolved from there.

More specifically, the application of the COIN framework to network routing has been explored in [157, 180, 181]. Figure 5.4 presents the network architectures investigated in [181], where the nodes in black are dedicated destinations and all other nodes are routers that make routing decisions at each time step on how to forward packets to the destination and also generate traffic consisting of a pair of a real-valued traffic rate and a destination.

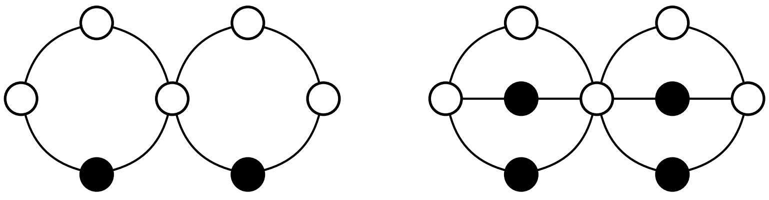

(a) Network A (b) Network B

Figure 5.4: Network Architectures

To account for an heterogeneous infrastructure, each router may have a different load-to-delay function, $W(x)$. This function takes an estimate of the load over a given time window at a given router and produces a measure of delay based on the load. The objective of the collective of routers and destination pairs is minimising the total delay encountered by all traffic in the network. Recall that the expected average delay is calculated as the running average of all delays encountered in a queue (3.8). The agent, η, is modelled as a unique pair of router and destination, where the vector $\underline{\zeta}_{\eta,t}$ holds the traffic sent along all edges emanating from η's router tagged for the respective destination at time t. Each subworld, ω, is constructed as the set of all agents that share a common destination node. While shortest path algorithms for routing packets in a network set $\underline{\zeta}_{\eta,t}$ to minimise the total delay emanating from η's router to the ultimate destination, the COIN modelled scenario tries to set $\underline{\zeta}_{\eta,t}$ in order to minimise the total delays of the subworld containing η. In other words the objective is to optimise the subworld utility g_ω. Wolpert et al. [181] modelled three scenarios, first a shortest path algorithm and second a COIN model both based on full knowledge of the window-averaged load at time $t-1$ of all routers in the network. These window-averaged load values are taken as estimations for the load at the given routers at time t. Third, a COIN model with limited knowledge only based on the reward received upon packets reaching their respective destination.

More formally, the load at router r at time t is determined by $\underline{\zeta}$, which gives a load-to-delay function $W_{r,t}(\underline{\zeta})$. The world utility is then the sum of all encountered delays at all routers over time, $G(\underline{\zeta}) = \sum_{r,t} W_{r,t}(\underline{\zeta})$. Subsequently, the wonderful life subworld utility is $g_\omega(\underline{\zeta}) = \sum_{r,t} \Delta_{\omega,r,t}(\underline{\zeta})$, where $\Delta_{\omega,r,t}(\underline{\zeta}) = [W_{r,t}(\underline{\zeta}) - W_{r,t}(CL_\omega(\underline{\zeta}))]$. Δ represents the delays of a packet accrued along the path towards the destination. Once a packet reaches the destination, all delays of all packets received are summed and acknowledged to all the router within the subworld. These acknowledgements are used by the COIN model with limited knowledge to evaluate the WLU-based reward of its subworld. Their results show that the COIN-based models outperform the shortest path algorithm. The COIN model based on limited knowledge shows a reduced performance to the one with full knowledge about the collective, but still surpasses the shortest path-based algorithm. These results confirm that despite the absence of centralised communication and control authorities, the individual utility maximising behaviour yields good global performance without disadvantaging any agents.

In an extended study, Wolpert and Tumer [180] show that COIN-based models for network routing almost always avoid the Braess' paradox. Braess' paradox states that selfish routing behaviour on a network can result in a lower throughput when additional capacity through a new edge in the network is introduced. In particular the ideal shortest path algorithm introduced side-effects that lead to the observation of the Braess' paradox. With a COIN-based approach, Braess' paradox can almost always be avoided while at the same time exhibiting significantly improved global throughput performance.

5.2 Decentralised MDP Framework

This section formalises the distributed task assignment problem using the the DEC-MDP framework introduced in Definition 4.4 with extensions to account for the specifics of the queueing-theoretic abstractions for the task hierarchy.

The formalism of Markov decision processes (MDPs) is the fundamental part of many stochastic planning problems. In particular, extensions to the basic MDPs for multi-agent systems received much attention. Often, the aim of these extensions is to formulate a particular problem, such that the complexity of planning is reduced and can be tackled with specific techniques. However, it has been shown that decentralised knowledge and control in a multi-agent setting is NEXP-hard [20]. So solving even simple problems is extremely hard and it is unclear how large-scale realistic problems can be solved efficiently. In this section, a decentralised MDP for queueing networks modelling distributed task assignment is defined, where the interaction network spans a finite number of agents defined as a directed acyclic graph evolved using the adapted BBV or ER models.

The distributed task assignment problem borrows its representational form from graph-theory. The hierarchical structure of such a queueing network is then defined as the set of nodes together with the set of pairwise relationships between them.

Definition 5.3. *A queueing network for distributed task assignment is defined by a* ***directed acyclic graph*** $\mathcal{DAG} = \langle \mathcal{V}, \mathcal{A} \rangle$, *where*

- $\mathcal{V}$ *is a finite set of vertices (or in the context of queueing network, servers). Each vertex* i *is associated with the poisson arrival rate* λ_{0i} *and the exponential service rate* μ_i.
- $\mathcal{A}$ *is a finite set of arcs, which are ordered pairs of vertices without self-loops and with no path that starts and ends at the same vertex,* $\mathcal{A} \subseteq \{(u, v) | u, v \in \mathcal{V} \wedge u \neq v\}$.

Definition 5.4. *Let* $\mathcal{DAG}$ *be a directed acyclic graph with vertices,* $\mathcal{V}$*, and arcs,* $\mathcal{A}$*. A* ***sub-task*** s_i *is defined as the work item to be completed at vertex* i *in a queueing network. If the vertex has* $\mathtt{deg}^+(i) = 0$*, then the sub-task is called* ***atomic****. The completion of a sub-task is stochastic given as the exponential service rate* μ_i.

Definition 5.5. *Let* $\mathcal{DAG}$ *be a directed acyclic graph with vertices,* $\mathcal{V}$*, and arcs,* $\mathcal{A}$*. A* ***local task*** t_i *is defined as a sequence of sub-tasks given as* $t_i = [s_1, \ldots, s_k]$ *for vertex* i *for* $k > 1$ *such that* (v_j, v_{j+1}) *is an arc in* $\mathcal{A}$ *for* $1 \leqslant j < k$*. A local task reduces to the atomic sub-task, if the vertex has* $\mathtt{deg}^+(i) = 0$*. The length of a local task is defined to be* k*. A* ***local task*** *is called a* ***task****, if it is requested through an external arrival event to the task network.*

Since the graph is directed and acyclic no sub-task can appear more than once in the local task sequence.

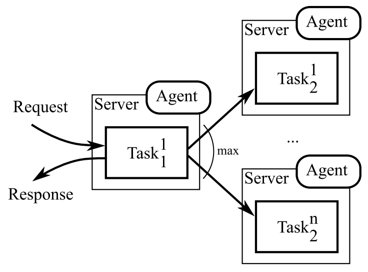


Figure 5.5: Multi-agent Task Structure

Figure 5.5 illustrates a typical task structure for distributed task assignment within the framework of queueing networks. Each server is managed by an agent to fulfill a sub-task. Once the sub-task is completed, the agent has a choice of forwarding the request to one of the servers to complete the next part of the overall task. The goal of the agent is to optimise this choice. One way to achieve this, is to minimise the response time which is calculated as the accumulated time it takes to complete all sub-tasks in the local task hierarchy or equivalently and more generally maximise the long-running reward. So once an atomic sub-task is completed (i.e., a leaf-node is reached), a response to all participating agents in reverse order of execution is sent to notify the successful completion of a task. Each agent observes the local state as the queueing metrics defined in equations (3.8) - (3.11).

Definition 5.6. *An n-agent continuous state **DEC-MDP** of a queueing network is defined by a tuple* $\mathcal{M} = \langle N, \mathcal{DAG}, \mathcal{A}, \mathcal{S}, \mathcal{P}, \mathcal{R}, \Omega, \mathcal{O} \rangle$, *where*

- N *is the number of agents in the environment.*
- $\mathcal{DAG}$ *is the directed acyclic graph describing the hierarchical structure of the task assignment problem. Each agent is represented as a vertex on the graph and the finite action set* $A = A_1 \times A_2 \times \cdots \times A_N$ *is implicitly provided by Definition 5.3, where the action set is equivalent to the set of arcs defined by the DAG,* $A \equiv \mathcal{A}$.
- $\mathcal{S}$ *is finite set of queueing network states.*
- $\mathcal{P}_{ss'}^{\vec{a}} = P\{s_{t+1} = s' \mid s_t = s, \vec{a}_t = \{a_1, \dots, a_i\}\}$ *is the transition probability of state* s' *when the actions* $\{a_1, \dots, a_i\}$ *have been taken in state* s.
- $\mathcal{R}_{ss'}^{\vec{a}} = \mathbb{E}\{r_{t+1} \mid s_t = s, \vec{a}_t = \{a_1, \dots, a_i\}, s_{t+1} = s'\}$ *is the expected value of the next reward taking actions* $\{a_1, \dots, a_i\}$ *in state* s *and transitioning to the current state* s'.
- Ω *is the set of all observations for each of the agents.*
- $\mathcal{O}$ *is the observation function.* $O\left(s, \{a_1, \dots, a_i\}, s', \{o_1, \dots, o_i\}\right)$ *is the probability of agents 1 to* i *seeing observations* $\{o_1, \dots, o_i\}$ *after selecting actions* $\{a_1, \dots, a_i\}$ *in state s.*

Satisfying the Markov property, the goal of reinforcement learning is to learn a policy π that maps a state vector to an action to maximise the total amount of reward received over the long run. Since the transition function P is not known the agent learns an action-value function known as $SARSA(0)$ (equation (4.26)). The optimal value function $Q^*(s, a)$ is guaranteed to converge, provided that the learning rate decays to zero asymptotically and the policy for action selection chooses the action with highest Q-value in a given state, i.e., is asymptotically greedy.

However, since each agent makes local decisions to minimise the response time of local task completion, agent interaction implies a non-stationary environment. Therefore, an asymptotically greedy action selection would likely result in saturated paths without being able to recover from it. As a result, all learning agents in the system would need to employ a policy that allows a certain level of exploration at all times. From the perspective of queueing theory, the routing probabilities in Definition 3.1 become deterministic under a greedy policy. If the stability criterion (3.3) holds, after solving the traffic equations in matrix notation (3.2), then the Jackson Theorem [76] holds as well and a global optimal solution can be reached with an asymptotic greedy action selection (see Definition 3.1). However, in practice the Jackson Theorem cannot be guaranteed. Deterministically selecting among the available actions will likely congest this path. So an agent needs to adapt to changing conditions and provide a means of continuous optimisation while also converging to a stable stochastic policy [2]. From a topological perspective, a greedy policy that always selects the best action reduces the graph to a tree. A tree is less robust, because any failure will have detrimental effects on the system performance, if no measures are employed to cope with broken or badly performing actions.

The fact that a certain level of exploration will always be required by the agent implies that strong guarantees of the Jackson Theorem cannot be made. The transition function will always select among all available actions that settle to best solutions temporarily, but fluctuate over the long-run. In effect, this is one of the main features of learning. Agents linger with their currently best action and exploit their knowledge accordingly only to drop it in favour of a better one once it stops performing well. As a consequence, multi-agent reinforcement learning systems exhibit hysteresis. A subjective belief model of the performance of the available actions is maintained. In the case of reinforcement learning, the belief model is represented by a mapping from state signals to action-values. The belief model is temporarily fulfilled and shifts towards a better model when they cease to be fulfilled. This, however, does not rule out that this shift is irreversible. Exploration ensures that the whole action space is considered. This informal insight has a resemblance to biology. Agents differ in their role within a task network and they have distinct interaction patterns based on their belief models. Thus the system is co-evolutionary. Chapters 6 and 7 analyse whether a task network in a particular context converges to an equilibrium solution or whether the system remains in an open-ended state of co-evolution. This line of argument follows that of Arthur and applies to situations where agents are not able to build a perfect model of the world and act optimally. Arthur [9] showed that a multi-agent system where perfect global knowledge is not attainable, inductive reasoning provides a model that facilitates self-organising behaviours with robust solutions.

To facilitate a compact representation of the state space standard backpropagation feedforward neural networks with one hidden layer are employed on each arc of the task network to estimate the Q-values for each action given a state vector [127]. Alternatively, each agent could employ a neural network with one output neuron for each action. The disadvantage of this approach is that the utility of one action affects the other ones as well and hence results in poorer performance compared to using neural networks for each action (i.e., arcs in the task network) separately [3]. Equation (4.26) can then be expressed as the general gradient-descent update rule for neural network training as

$$\Delta\omega_{t+1} = \alpha[\upsilon_{t+1} - Q(s_t, a_t)]\nabla_{\omega_t} Q(s_t, a_t) + \eta\Delta\omega_t, \tag{5.8}$$
$$\upsilon_t = r_t + \lambda Q(s_t, a_t), \tag{5.9}$$

where η is a constant representing the momentum, which determines the effect of past changes to the weight vector, ω, and $\nabla_{\omega_t} Q(s_t, a_t)$ is the vector of partial derivatives of the value function $Q(s_t, a_t)$ with respect to the weight vector ω_t. So, the action-value estimation is updated every time a task in the DTAP network is completed. That means, that all value functions of all arcs in the DTAP network that were involved in forwarding a request to the next sub-task will be updated according to $\omega_{t+1} = \omega_t + \Delta\omega_{t+1}$. So, the optimal action-value function Q^* is approximated with a parametric function approximator, Q_ω, where ω is the vector of weights as given above.

The neural network is continuously trained over a sliding window of the most recent N_w input data with the objective to minimise the mean squared error. As new input/output pairs are added, the oldest ones are dropped in a first-in-first-out fashion. If the mean-squared error of the data contained in the window are below a given error threshold, training the network does not proceed to avoid over-fitting. The hidden neurons employ a sigmoidal activation function and the output neuron is equipped with a hyperbolic tangent activation function.

The formulation in Definition 5.6 can be simplified by factoring the state space and the observation function (see Definition 4.5). Factoring the state space of a Dec-MDP queueing network is straight-forward in the case considered here. Equations (3.8) - (3.12) are calculated locally for each server (i.e., agent). The system state is calculated as the total expected average waiting time, equation (5.10), total expected average number of events in the queue, equation (5.11), the mean utilisation, equation (5.12), and the total average time a task request spends in the system, equation (5.13).

$$W = \sum_{i=1}^{n} \hat{w}_i \tag{5.10}$$
$$Q = \sum_{i=1}^{n} \hat{q}_i \tag{5.11}$$
$$\bar{U} = \frac{\sum_{i=1}^{n} \hat{u}_i}{n} \tag{5.12}$$
$$S = \sum_{i=1}^{n} \hat{s}_i. \tag{5.13}$$

5.2.1 Modelling Cooperation in DTAPs

Modelling cooperation in multi-agent systems is mostly a question of social welfare, because the agents participating in such a system are often referred to as a “society of agents”. If communication among the agents is absent from the design of the system, then the cooperative behaviour relies on the structure of the reward function that allocates rewards (or utilities in game-theoretic terminology) to each agent that participated in accomplishing a task. Utilitarian social welfare is defined as the sum of the individual rewards to measure the quality of accomplishing a task to the system as a whole. While this welfare function may be appropriate for many decentralised problem settings, it is problematic in scenarios where a fair share of the overall reward is required by each individual agent. Other social welfare functions exist [8, 105], but are not further elaborated upon, because the reward function presented in this thesis provides a natural interpretation (see Section 5.2.1.2). Consequently, this leaves out a range of topics that deal with negotiating to achieve Pareto optimal outcomes to everyone's mutual benefit. Designing incentives for a negotiation process is known as mechanism design.

5.2.1.1 Centralised Communication and Control

There are different ways to achieve cooperative behaviour in decentralised multi-agent learning domains. One is an architectural design that represents a team of agents with a single learning authority. In a most extreme case, a single agent learns behaviours for all its team members covering the entire task hierarchy. Otherwise, some logical partitioning of the task hierarchy can be devised with a single learning authority for each team. From a networks perspective this is equivalent of finding disjoint communities. If the task hierarchy is dynamic, i.e., services can join, leave, and rewire existing connections, then this requires agents to self-organise into team structures. As a consequence, a potential benefit of partitioning the task hierarchy is offset by the complexity of a higher level organisation structure that must be learnt itself. Additionally, the space complexity of team learning cannot be neglected either. Assuming an environment with $\mathcal{S}$ states, a team with N agents can take any of $|\mathcal{S}|^N$ states [71, 117]. So, team learning suffers from the curse of dimensionality.

Another constraining factor are the communication channels between agents. A single authority requires consensus of the states of its team members and therefore posits strong guarantees onto the communication medium. In practice, however, strong guarantees on the communication medium can rarely be assured. In particular spatially dispersed agents may not be reachable. If unrestricted communication is assumed, then the question is whether this setting is really multi-agent. Stone and Veloso [143] argue that unrestricted communication is isomorphic to single-agent systems, where complete state information can be exchanged to derive optimal behaviours.

Cooperation can also be achieved by modelling the other team members and either forming coalitions [29, 30], or estimating state-action values for the joint actions [33]. The latter requires complete knowledge of the actions taken. In a deep task hierarchy, this is problematic, because actions that cannot be observed need to be communicated. Additionally, it also poses scalability issues, because each agent requires a model of all relevant joint actions.

5.2.1.2 Decentralised Communication and Control

Primarily, cooperative behaviour for distributed task assignment problems in this thesis is achieved in a decentralised fashion to accommodate the natural setting without imposing central authorities.

The rewards received by each agent are not independent, that is the global reward is not equal to the sum of the local rewards. Noting that the reward is the response time of the completion of a local task one can informally see that a reward can only be given, when the local task is completed. When an external request for a task enters the system an execution path is given as $[s_1, s_2, \dots, s_i]$. So the local tasks can be similarly expressed as $[t_1, t_2, \dots, t_i]$, where each t_i is a subsequence of the execution path, i.e., $t_1 = [s_1, s_2, \dots, s_i]$, $t_2 = [s_2, \dots, s_i]$ until finally $t_i = [s_i]$ is an atomic sub-task. Consequently, since the rewards are computed based on completion times of local tasks t_i a reward relationship of $r_1 < r_2 < \cdots < r_i$ is established, so each r_i is the negative value of the response time of the local task t_i. While the rewards account for all completion times of the local tasks, the time of events in the system is only taken for completed tasks, i.e., the server receiving external arrival events measure the response time of the completion of the task this external event triggers (eq. (5.13)).

This reward structure is said to be global, because it incorporates the rewards of all agents involved in completing a task. Additionally, it is implicit, because the mathematical framework of queueing networks provides the definition in terms of completion times. A global reward structure is a feature of cooperative multi-agent systems, as distinct from local reward functions that encourages competitive behaviour among selfish agents. As a consequence no communication is required to correlate rewards and apportion the reward fairly to the agents involved. This makes the credit assignment problem straight-forward to solve.

In addition to implicit coordination given by the global reward structure, agents may share their states with interested parties. In distributed task assignment an agent may be interested in the current state of the agents downstream, e.g., to evaluate the current utilisation or delay in queues. This way, faster learning times can be achieved, because the agent does not rely on its experience to find out about the performance, but instead can communicate directly with the relevant agents. This postulates cooperative agents that are willing to share information and be truthful about their state. It is not assumed that agents cheat to receive more task transactions.

In this thesis, it is assumed that the communication is immediate without a communication delay, so that an agent can include a neighbour's state directly into their state representation. This may lead to more complicated overlapping state spaces, so it may not be possible to factor the state space anymore.

5.2.1.3 Collaborative Function Approximators

Function approximators in reinforcement learning are used to reproduce the Q-value function that maps state signals to action values. Their use is very generic and powerful, because each state variable that the agent senses in its environment can be accounted for as an input neuron to the neural network. The structure of the distributed task assignment problem modelled as queueing networks provides rich information about the state, such as delay in queues (3.8) and utilisation (3.11).

Intuitively, optimal decisions in a given scenario can be made, if an agent knows the state of the queues of its neighbours. Allowing communication between an agent and its neighbours, this state information can be queried and incorporated as well. this means that the neighbouring states are more expressive and allow better predictions on how an action is going to perform. However, optimal decisions in queueing networks in general are difficult. Forwarding task requests to the shortest queue (shortest-line rule) minimises the expected delay of each task request and the long-run average delay per task request when the service-time distribution is exponential [175]. For general service-time distributions this result does not hold anymore [174]. Assuming that the agent knows the time currently serviced tasks have been in service additional to the queue length, then the expected delay of each task request can be calculated conditioned on the remaining service times. Given this information, a rule to forward task requests to queues with the shortest expected delay seems a natural choice. However, Winston [175] give counterexamples of when this rule does not minimise the long-run average delay per task request. Whitt [174], Winston [175] considered queueing systems that resemble a toll plaza scenario, which are more simple than the queueing networks considered in this thesis. Therefore, the use of function approximators provides richer information to guide an agent's decision process, because a mapping of the long-run average response time with respect to the queueing state variable(s) is established.

Each agent maintains its own function approximators. In order to avoid interferences in supervised learning, one neural network function approximator is employed for each action [3]. This implies that the number of function approximators scale with the number of arcs in the task hierarchy.

This also means that there is more than one function approximator for certain actions, e.g., choosing a particular service in the task hierarchy. This is depicted in Figure 5.6. The agent for service s2 and the agent for service s3 both maintain a function approximator for the their target service s4.

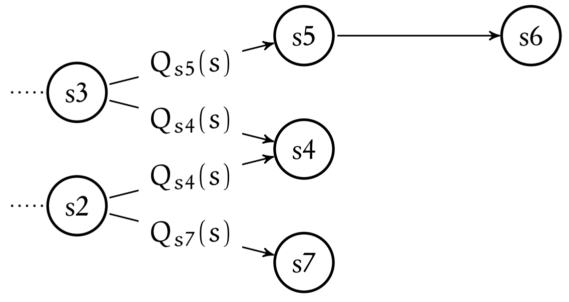


Figure 5.6: Function Approximation in Distributed Task Assignment

Given a cooperative multi-agent reinforcement learning setting function approximators can be collaboratively trained agents sharing the same destination node downstream under certain conditions. Thus, the state-action value mapping can collaboratively learnt. First, communication channels to all direct target services need to be established with benevolent agents at the other end. Second, the state representation can only include states from the neighbours, because agents cannot observe the state of the services using them.

Not only does this design scale with the number of agents in the environment, instead of the number of actions, it also accounts for concurrent learners. A local function approximator implies that a mapping from states to action values based on an agent's experience alone misses out on the adaptation underway in parallel. This is particularly true for queueing networks in general, because the most expressive local state is that of an agent's neighbours.

As a result, the learning rates of the agents will be faster, because each agent trains the neural

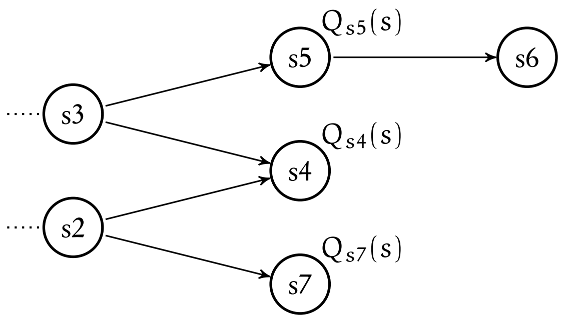


Figure 5.7: Collaborative Function Approximation in Distributed Task Assignment

networks at the target node instead of locally. This way the information about a target service is shared among all agents connecting to it.

5.3 Summary

Multi-agent reinforcement learning in the context of distributed task assignment problems was investigated. An integrated approach is adopted to combine a complex network evolution model, queueing theory and Markov decision processes into an agent-based simulation environment formalised by a decentralised Markov decision process. By evolving social networks that adhere to the semantics of queueing systems, large-scale computer experiments are possible, which are empirically investigated in the next chapter. An emphasis was placed on the global, implicit reward structure to facilitate a cooperative multi-agent learning environment. This included the specification of how agents learn, in particular using $\mathsf{SARSA}(0)$ reinforcement learning coupled with a neural network function approximator to estimate the state-action value function. An extension to the learning architecture was introduced to train the Q-value function collaboratively involving agents with the same target agent. That way, the learning rate can potentially be reduced, while also incorporating more informative state signals from the target agents.

This chapter also covered some related works including more advanced machine learning methods that avoid difficult to tune parameters. More importantly, however, are the related works in task allocation and distributed task assignment to provide an overview of how other researchers addressed problems similar to the one laid out in this thesis.

— Ever tried. Ever failed. No matter. Try Again. Fail again. Fail better.

Samuel Beckett

6 Calibrating Multi-agent Reinforcement Learning Methods

Modelling distributed task assignment as presented in the previous chapter is challenging, because the learning algorithms employed need to be adjusted such that optimal long-term responses and optimal short-term behaviours are attained simultaneously. For example the uptake of available information or experience has to be quick enough to efficiently incorporate knowledge for future decisions and slow enough to avoid vastly fluctuating between those decisions. The scope for calibrating the learning methods for multi-agent task assignment problems in this chapter therefore includes finding appropriate parameters for all free parameters that yield minimum total event processing times in all scenarios. Moreover, it is beneficial to understand to what degree the free parameters influence the total event processing time in the vicinity of the optimal settings. Complementary to this chapter, the next one investigates the dynamic time-evolving learning behaviour in more detail.

In pursuit of these two goals different scenarios are investigated with a focus on two different learning policies (ϵ-greedy and weighted policy learner), two different underlying task topologies (BBV and ER models introduced in Section 3.2), and two different queueing stress levels. Note that the queueing stress level is induced by the network evolution model based on two constants, δ_V and δ_E. Higher stress levels imply an increased utilisation rate, more events are processed within the task network, and there is less latitude for decision making processes. Higher stress is exercised for lower values of these constants. As values for these constants approach 1, the utilisation tends to 100%. Importantly, the queueing performance measures become unstable for utilisation rates of $u > 80\%$. The two different stress levels correspond to $\delta_V = \delta_E = 1.2$ and $\delta_V = \delta_E = 1.8$ respectively.

The size of the task network in all evaluation scenarios is fixed to 1000 nodes, with a maximum out-degree $d_{max} = 3$ for the BBV network evolution model which will create approximately 2000 arcs (or actions). To facilitate a similarly sized random network based on the ER model the generator

$G(N, M) = G(1000, 2000)$ is used.

The calibration of the multi-agent reinforcement learning methods is conducted within the intervals $\alpha \in [0.001, 0.1]$ for the learning rate, $\lambda \in [0.5, 0.9]$ for the discount factor of the $SARSA(0)$ reinforcement learning method, $\eta \in [0.01, 0.5]$ for the momentum of the function approximator, and $\zeta \in [0.001, 0.1]$ for the update factor of the gradient vector in the weighted policy learner algorithm. The generic architecture of neural networks as function approximators for the state-action mappings provides some flexibility of representing the state of each agent. To represent the state for the state-action mapping any of the online queueing metrics presented in Section 3.1 can be taken into account, but for simplicity the purely local average waiting time (equation (3.8)) is used. In all multi-agent reinforcement learning scenarios, the neural network is instantiated with 8 hidden neurons. This neural network architecture was determined using small-scale pilot experiments. If the state representation included more variables then a larger hidden layer may be necessary.

Section 6.1 gives a brief overview of the methodology used to control the experiment and conduct the analysis. For more details on response surface methodology see Chapter 2.

Section 6.2 provides the results for distributed task assignment over network structures according to the BBV model (Section 3.2.2). For the experiments in this section, the network structure itself exhibits a degree of freedom. The calibration of the learning method relies on the network structure instead of the network instance. The multi-agent reinforcement learning scenarios are compared against baseline tests that do not incorporate any intelligent decision-theoretic solution. Section 6.2.1 presents the solution with a simple uniformly random action selection. This solution assumes that the actions are all taken with equal probabilities, i.e., $1/\mathtt{deg}^{+}(i)$ at server i. Section 6.2.2 evaluates reinforcement learning techniques and compares the results with the baseline test of uniformly random decision policies. Section 6.2.2.3 introduces two new scenarios where the function approximators are collaboratively trained by nodes that share the same destination node in the task network. This has the advantage that agents can benefit from each other's experience with a particular service node downstream.

Similarly, Section 6.3 presents the results for distributed task assignment over random network structures. This time, the structure of the graph is fixed, because the random graph generation rules based on the Erdős and Rényi model (Section 3.2.3) introduce too much variation in the queueing performances with different random networks. Similarly to above, the multi-agent reinforcement learning scenarios are compared against uniformly random decision policy.

6.1 Response Surface Methodology

In the approach presented in this thesis the simulation experiments are controlled systematically within a response surface modelling framework called Kriging [94, 107] presented in Chapter 2. The input domain, $\Omega \subseteq \mathbb{R}^d$, covers the free parameters of the respective method employed. An optimal Latin Hypercube Sampling design of experiments is used to sample the input domain, Ω [185]. At each sampled location, x_i, a number of simulations are performed with different random number seeds and the system state is calculated as the total event processing time, equation (5.10), total expected average number of events in the queue, equation (5.11), the mean utilisation, equation (5.12), and the total event processing time, equation (5.13). The replications are controlled online to achieve the confidence level of 90% with an error of 10% (see equation (2.29)).

A metamodel is fitted for the total event processing time S (equation (5.13)) using Monte Carlo Markov Chain sampling (Algorithm 2.1). The model quality is assessed with the $R^2_{\text{prediction}}$ and the $RMSE$ statistics on a set of validation locations. $R^2_{\text{prediction}}$ is a normalised quantity that ranges usually from 0 to 1 with 1 indicating a perfect fit. If $R^2_{\text{prediction}} < 0.95$ then a validation set of scattered locations over the domain is sampled and their mean-squared error, MSE, evaluated. The location with the highest mean-squared error is chosen as the new location where the simulation is performed and added to the data set. Fitting another response surface, this procedure repeats until $R^2_{\text{prediction}}$ shows the desired level of accuracy. Once the model achieves a good fit, global optima on the surface are determined using simulated annealing [64]. Simulated annealing uses a randomised neighbourhood search strategy to escape local minima, which makes it less likely to fail to converge on difficult functions [80]. Finding these global optima is embedded into the sequential improvement of the Kriging metamodel as well, until the difference between the sampled optima and the interpolated ones are satisfactory.

Then, in a smaller sub-domain that includes the minimum response, a ridge analysis is conducted. Ridge analysis graphically portrays the behaviour of quadratic response surfaces (a polynomial with linear, interaction, and quadratic effects) and is able to elucidate factor dependence between the independent variables in an elementary fashion (see Section 2.5.1 for the mathematical derivation of the formulas). This way the nature of performance contours can be examined, which is beneficial in higher-dimensional spaces. For lower dimensional surfaces, contour plots can be visually examined to find optimal responses. From a practical standpoint, this is not feasible for dimensions higher than 5, because the number of pair-wise contour required scales as $d!/2$ in the number of dimensions, d. Since, the dimensionality of the scenarios is 3 for the ϵ-greedy learning policy and 4 for the weighted policy learner, both, the contours and the ridge analysis is presented.

In order to illustrate the sensitivities independent of the scale of the variables, all three independent variables are scaled to the unit interval, $\alpha, \lambda, \eta, \zeta \in [0, 1]$.

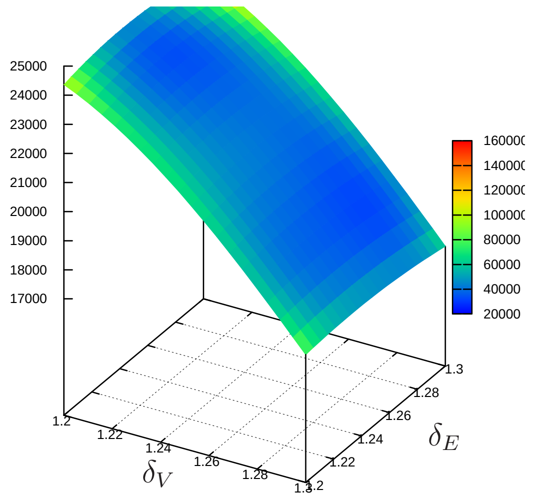
All the results with respect to MCMC Kriging are included in Appendix A.

6.2 DTAPs with the BBV Model

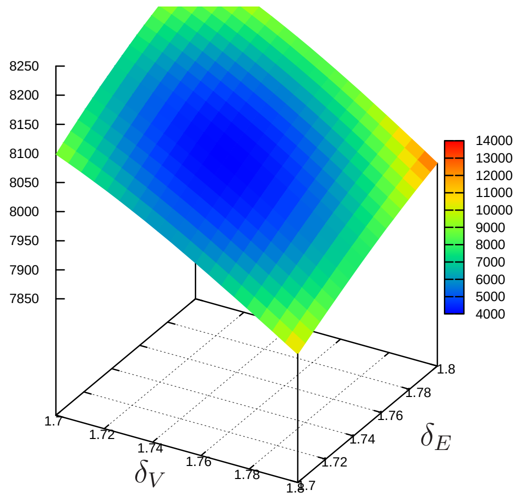
6.2.1 Uniformly Random Transition Probabilities

Section 3.2 introduced a model to generate queueing networks with n servers. Given the two designs, $\delta_V \in [1.2, 1.3]$ with $\delta_E \in [1.2, 1.3]$ and $\delta_V \in [1.7, 1.8]$ with $\delta_E \in [1.7, 1.8]$, the resulting networks are stable with respect to equation (3.3). The resulting average response time surfaces based on 22 and 21 samples for the first and second designs respectively are presented in Figure 6.1.

As expected from the analytical results obtained in Figure 6.1(b), these surfaces show that the slope in the direction of δ_V is more pronounced in Figure 6.1(a) than in Figure 6.1(b), showing the different queueing stress levels on the performance of the total event processing time. Although, because of the stochastic nature of the simulations, this can only be considered an approximation to the analytical behaviour, but it is enough to illustrate the performance of the queues without any intelligent decision policies employed. The stochastic irregularities are most pronounced in



(a) With $\delta_V \in [1.2, 1.3]$ and $\delta_E \in [1.2, 1.3]$



(b) With $\delta_V \in [1.7, 1.8]$ and $\delta_E \in [1.7, 1.8]$

Figure 6.1: Event Processing Time for the BBV Model

Figure 6.1(b) with a slightly tilted surface, which results in a minimum response with $(\delta_V, \delta_E) = (1.8, 1.7)$ and a maximum response with $(\delta_V, \delta_E) = (1.7, 1.8)$ respectively. The minima and maxima are presented in Table 6.1.

The mean-squared errors are encoded in a colour gradient on the surfaces. The boundaries of Figure 6.1(b) show higher errors compared to the centre of the design domain. Additional samples along the boundaries may have corrected the tilted surface to provide a better fit to the analytical results.

Table 6.1: Results of BBV DTAPs with a uniformly random decision Policy

	$\delta_V, \delta_E \in [1.2, 1.3]$		$\delta_V, \delta_E \in [1.7, 1.8]$	
Variable	Minimum	Maximum	Minimum	Maximum
δ_V	1.3	1.2	1.8	1.7
δ_E	1.3	1.24	1.7	1.8

The discrete-event simulation of the uniformly random decision policies are considered baseline test scenarios for the reinforcement learning-based scenarios. All scenarios will be examined with high and low queueing stress levels with respect to the total average event processing time. The mean total event processing time which is simulated based on the Kriging surfaces for the uniformly random decision policies are 24380.94 for the high load character (at $(\delta_V, \delta_E) = (1.2, 1.2)$) of the queue and 8002.26 for the low load character (at $(\delta_V, \delta_E) = (1.8, 1.8)$) respectively.

6.2.2 Adaptive Transition Probabilities

This section evaluates intelligent agents that are endowed with the $SARSA(0)$ temporal-difference reinforcement learning method. Two different policies are considered that map action values to action selection probabilities. ϵ-greedy is an undirected policy that exploits the acquired knowledge about the environment most of the time and occasionally explores randomly. This policy, however, does not reflect the uncertainties about actions. In contrast, the weighted policy learner is a directed policy, because the policy space is directly updated based on a policy gradient. Consequently, it assigns probabilities to actions that reflect the perceived performance.

6.2.2.1 $\mathsf{SARSA}(0)$ with ϵ-greedy Policy

First the $\mathsf{SARSA}(0)$ reinforcement learning method with the ϵ-greedy policy is evaluated. The Kriging metamodel is fitted for the total average event processing time covering the domain $\Omega = (\alpha, \lambda, \eta)$, where $\alpha \in [0.001, 0.1], \lambda \in [0.5, 0.9], \eta \in [0.01, 0.5]$ with a total of 36 simulations. To admit continuous adaptive cycles as discussed in Section 5.2 the ϵ-greedy policy is fixed to $\epsilon = 0.05$. The global minimum and maximum of the Kriging function was found to be $23641.03(257306.01)$ and $24366.33(257307.76)$ time units respectively (estimated MSE (2.17) in brackets)[1]. Figure 6.2 presents the probability densities of both, the minimum and maximum responses on the Kriging surface and the density for a uniformly random decision policy at $(\delta_V, \delta_E) = (1.2, 1.2)$. The densities for the minimum and maximum responses given all samples in the MCMC chain were generated from $N(\mathbb{E}(y(\mathbf{x})), Var(\hat{y}(\mathbf{x})))$, where the expected value is defined in equation (2.6) and the variance is given in equation (2.17). This approach integrates the stochastic uncertainty (from the computer simulation) given in dispersion matrix $\mathbf{\Psi}$ and the structural (from predicting unknown locations) and parametric (from the MCMC inference) uncertainties. The densities with the solid lines represent the simulated best and worst possible calibration of (α, λ, η) for the multi-agent reinforcement learning method in this scenario. Looking at the mean of these distributions indicate that both best and worst cases perform better than uniformly random decision policies. The probability densities presented in Figure 6.2 provide more detailed information integrating all sources of uncertainties. The density that represents the optimal response (solid curve in light blue) has very little overlap with the density of the baseline test (dashed curve in light blue) and consequently can be considered a significant improvement. However, the worst possible calibration (solid curve in dark blue) within the design domain does overlap significantly with the simulation that does not employ intelligent agents. This illustrates the importance of investigating the learning behaviour with different parameters.

The contour plots of the pair-wise independent variables with respect to the total event processing time is presented in Figure 6.3. The learning rate, α, versus the discount factor, λ, shows two regimes where the response is minimised. For $\lambda \sim 0.5$, the learning rate has little impact on the queueing performance metric considered here. Also, for $\lambda \sim 0.9$ the response is minimised for $\alpha \to 0.001$. The second contour plot relates the learning rate with the momentum parameter, η, of the neural network function approximator. It shows that $\eta \sim 0.27$ and $\alpha \to 0.001$ minimise the response. Similarly, in the last contour plot the minimum response is attained for $\eta \sim 0.27$ and two regimes for the discount factor $\lambda \to 0.5$ and $\lambda \to 0.9$.

A summary of the minimum, maximum, and the normalised eigenvalues of the response surface are presented in Table 6.2. Once the minimum response is determined using simulated annealing over the Kriging function a subset of the design domain is used to analyse the sensitivities of the learning parameters. Given the optimal calibration of the learning method, the sub-domain for the canonical analysis is defined as $\Omega_{CA} = (\alpha, \lambda, \eta)$, where $\alpha \in [0.001, 0.01], \lambda \in [0.7, 0.9], \eta \in [0.2, 0.4]$.

[1]The concept of time is purposefully left abstract as discrete time steps

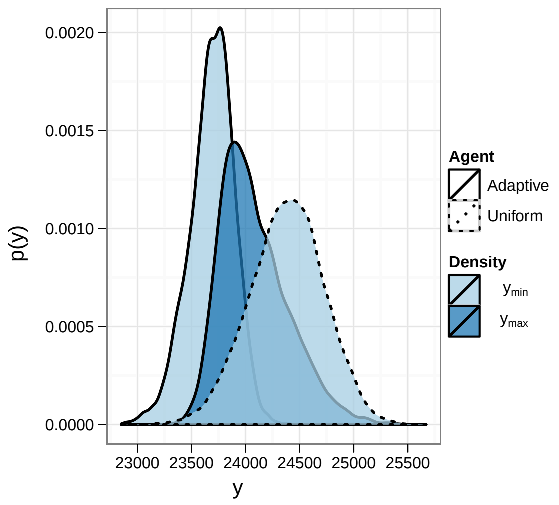


Figure 6.2: Probability Distributions of the minimum and maximum responses of the BBV model with $(\delta_V, \delta_E) = (1.2, 1.2)$

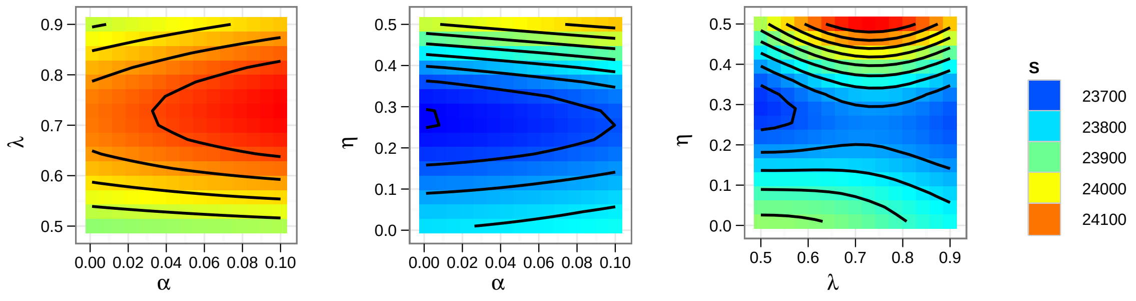


Figure 6.3: Contour Plots of the BBV model with $(\delta_V, \delta_E) = (1.2, 1.2)$

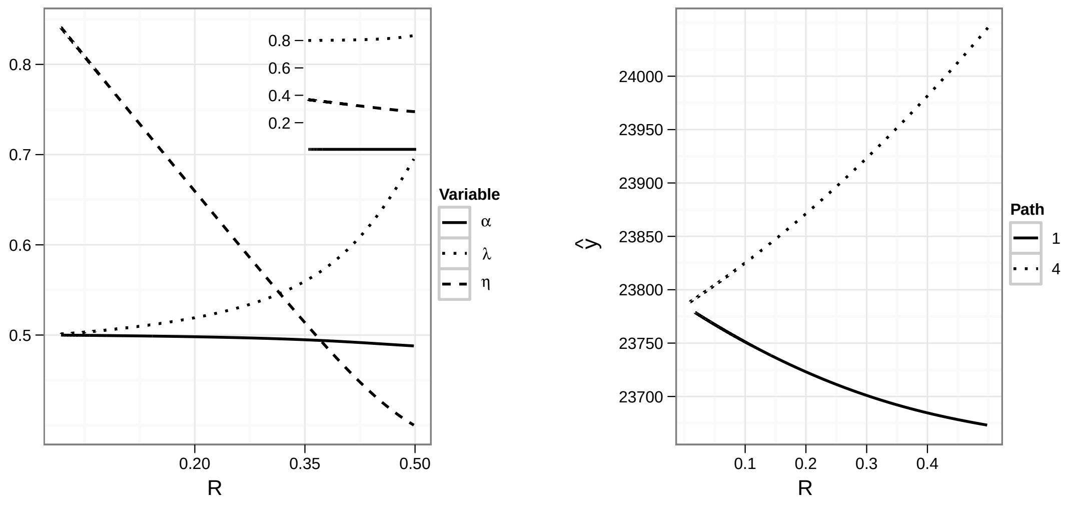
The ridge analysis plotted in Figure 6.4 of the response surface reveals the sensitivities of the respective independent variables on a path from a focal point in the design sub-domain, Ω_{CA}, to the minimum response. Recall that ridge analysis proceeds by constructing a mental sphere of radius R and records the coordinates (here (α, λ, η)) of the minimum attainable response. Essentially, the minimum ridge is the path of steepest descent for any distance from the focal point. Tracing the “minimum” path from the focal point, $(0.0055, 0.8, 0.37)$, towards the global minimum shows that the momentum of the neural network function approximator, η, has the highest impact on the total event processing time, followed by the discount factor, λ. The learning rate, α, does not exhibit a significant impact on the optimisation (see Figure 6.4(a)). The path of the discount factor along the minimum ridge indicates that the stationary ridge is exactly along the λ axis. This can also be verified by inspecting the first and last contour plot in Figure 6.3.

While the reinforcement learning algorithm succeeds in finding more optimal policies for the agents in the distributed task assignment network, the total event processing time is only marginally reduced.

One cause of this minor improvement can be attributed to the settings of the evolution model. The values for δ_V and δ_E are fixed to 1.2, which implies that each node induces the same factor of traffic into the network. This results in all nodes having similar performance characteristics. Further,

Table 6.2: Results of BBV DTAPs with SARSA(0) and ϵ-greedy Policy with $\delta_V = 1.2, \delta_E = 1.2$

Variable	Minimum	Maximum	Eigenvalues
α	0.001	0.1	50.73
λ	0.9	0.737151	1.52
η	0.272	0.5	-1005.23



(a) "Minimum $\hat{y}$" path for the independent variables (b) "$\hat{y}$" Ridges

Figure 6.4: Canonical Analysis for BBV DTAPs with SARSA(0) and ϵ-greedy Policy with $\delta_V = 1.2, \delta_E = 1.2$

the queueing metrics would all be quite sensitive to degrading performances at 80% utilisation across almost all nodes. The reinforcement learning agents operating under higher queueing stress levels have less latitude to adjust the action selection probabilities. Time-varying arrival and service rates for each node in the queueing network would present agents with a more heterogeneous queueing performance profile. These more dynamic situations would increasingly favour intelligent agents compared to uniformly random decision policies, because intelligent agents can recall learnt behaviour of situations and map those to optimal (or near optimal) action selections.

Another attribute of this minor improvement could be the employed policy. The ϵ-greedy policy is an undirected policy that favours the currently best action with $(1-\epsilon)$% probability. A stable stochastic policy, such as the weighted policy learner offers a more directed action decision policy with potentially improved queueing performances. Finally, the state-action mapping was purely based on local information, i.e., the local delay in the queue. A more meaningful state-action mapping would be to take into account the queueing metrics of the respective task nodes downstream. This setup requires a cooperative environment and is explored further in Section 6.2.2.3.

In order to illustrate the effect of reduced stress on queueing behaviour the same reinforcement learning method was used in a second scenario with network evolution parameters set to $(\delta_V, \delta_E) = (1.8, 1.8)$. The domain of the learning parameters is the same as above. The global minimum and maximum of the Kriging function was found to be 7003.39(2900.63) and 7119.898(4405.0) time units, which is an improvement of 12.5% and 11% respectively compared to the uniform agent.

Figure 6.5 presents the probability densities of both the minimum and maximum responses on the Kriging surface and the density for a uniformly random decision policy at $(\delta_V, \delta_E) = (1.8, 1.8)$. The density plot reveals that calibration within the design domain has very little impact on the learning performance. Both best and worst case parameterisations result in a significant better performance than with a uniformly random decision policy. So, within the given design domain of the reinforcement learning parameters adaptive agents have a significant advantage over agents with uniformly random decision policies. There is no overlap between the responses of the adaptive and the uniform agents (see Figure 6.5).

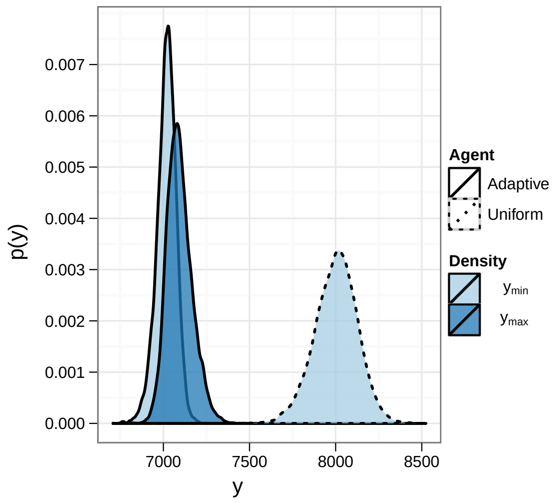


Figure 6.5: Probability Distributions of the minimum and maximum responses of the BBV model with $(\delta_V, \delta_E) = (1.8, 1.8)$

The contour plots of the pair-wise independent variables with respect to the total event processing time is presented in Figure 6.6. Notice that the range in the elevation levels on the contour plot is relatively low, which is reflected in the closeness of the best and worst-case parameterisations in Figure 6.5. The minimum response is best reflected in the last two contour plots with $\alpha \to 0.001, \lambda \sim 0.75, \eta \sim 0.2$. The first contour plot relates the learning rate with the discount factor and shows that λ has a higher influence on the response than the learning rate. This can also be seen in the results of the ridge analysis in Figure 6.7(a).

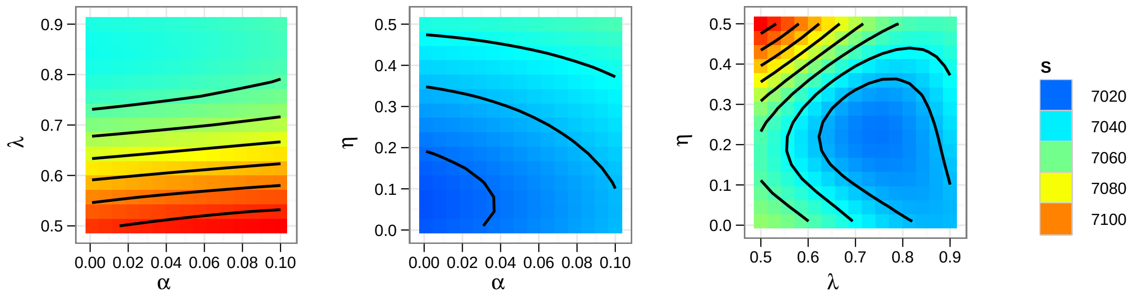


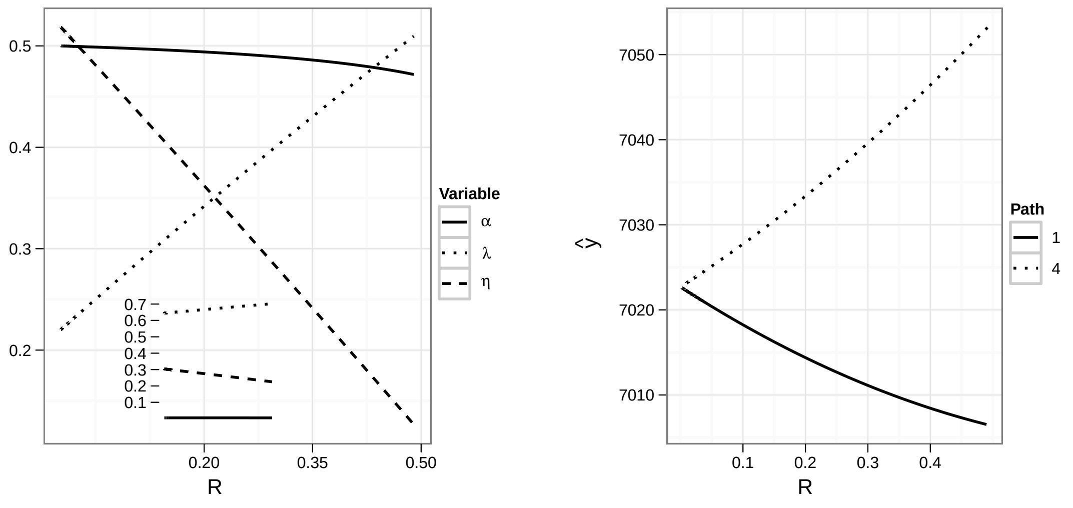
Figure 6.6: Contour Plots of the BBV model with $(\delta_V, \delta_E) = (1.8, 1.8)$

A summary of the minimum, maximum, and the eigenvalues of the response surface are presented in Table 6.3. The sub-domain for the canonical analysis is defined as $\Omega_{CA} = (\alpha, \lambda, \eta)$, where $\alpha \in [0.001, 0.01], \lambda \in [0.6, 0.8], \eta \in [0.2, 0.4]$.

Table 6.3: Results of BBV DTAPs with SARSA(0) and ϵ-greedy Policy with $\delta_V = 1.8, \delta_E = 1.8$

Variable	Minimum	Maximum	Eigenvalues
α	0.001	0.1	−7.69
λ	0.74	0.5	−57.46
η	0.20	0.5	−27.36

The ridge analysis is conducted from the focal point $(0.0055, 0.64, 0.3)$ and the results are plotted in Figure 6.7. Compared to the previous scenario with higher queueing stress levels, the optimal learning rate is the same. The optimal discount and momentum are both reduced from $\lambda = 0.9, \eta = 0.27$ to $\lambda = 0.74, \eta = 0.2$. On the minimum ridge defined from the focal point to the minimum response, the learning rate is stable, i.e., it has very little influence on the total event processing time. In contrast, both, the discount factor and the momentum are most sensitive to changes.



(a) "Minimum $\hat{y}$" path for the independent variables (b) "$\hat{y}$" Ridges

Figure 6.7: Canonical Analysis for BBV DTAPs with SARSA(0) and ϵ-greedy Policy with $\delta_V = 1.8, \delta_E = 1.8$

6.2.2.2 SARSA(0) with WPL

This section presents the results for the SARSA(0) reinforcement learning method coupled with the weighted policy learner. The results of the calibration with respect to the event processing time are given as probability densities of both, the minimum and maximum responses on the Kriging surface and the density for a uniformly random decision policy at high stress levels, i.e., $(\delta_V, \delta_E) = (1.2, 1.2)$, in Figure 6.8. The best calibration given the parameters for the learning methods presented in Table 6.4 improves the uniformly random decision policy by 13.78%. The worst case, however, significantly deteriorates the system performance by 21% compared to the uniform agent.

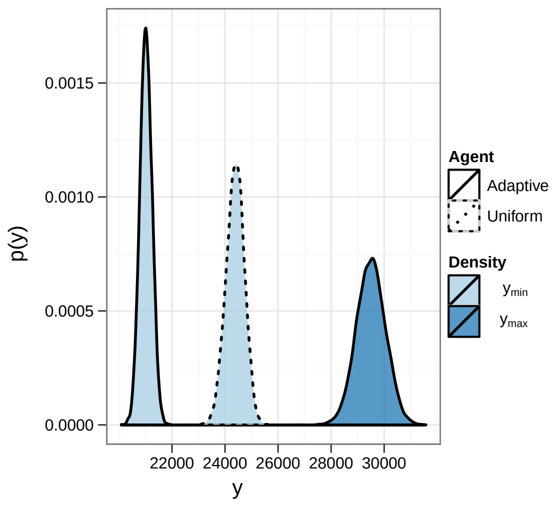


Figure 6.8: Probability Distributions of the minimum and maximum responses of the BBV model with $(\delta_V, \delta_E) = (1.2, 1.2)$

The contour plots shown in Figure 6.9 confirm this result. WPL introduces another parameter, ζ, to the learning parameters, which according to the contour plot has very little impact on the response with respect to all other variables. Additionally, there are two regimes that illustrate the region of optimum response. These are identified as the learning rate $\alpha \to 0.1$, the momentum $\eta \sim 0.1$, and the discount factor $\lambda \to 0.5$ or $\lambda \to 0.9$.

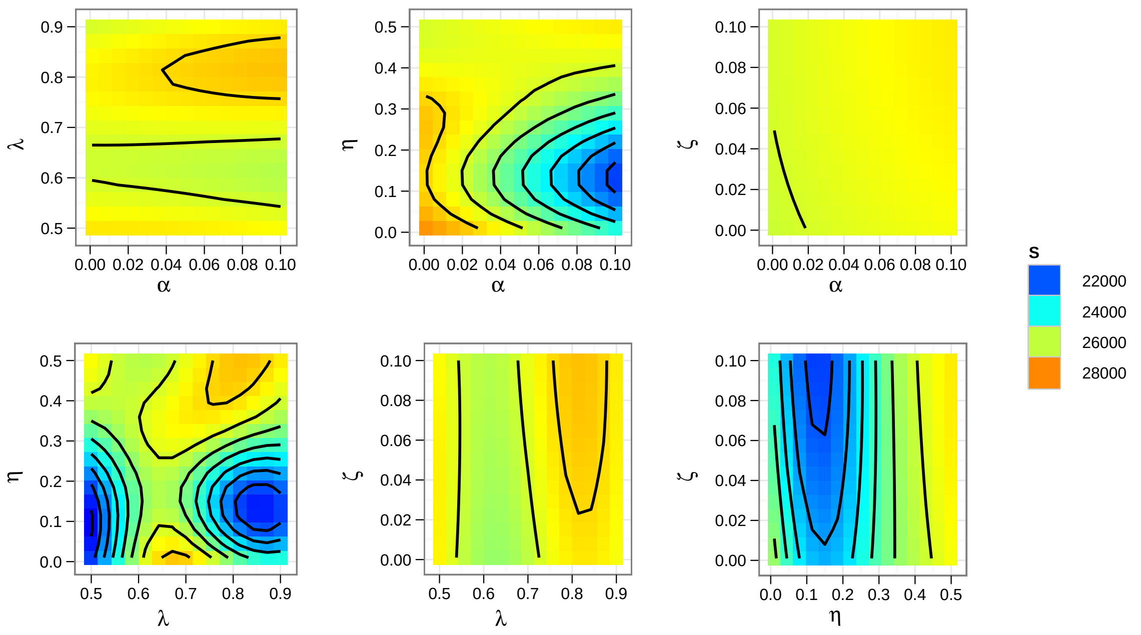


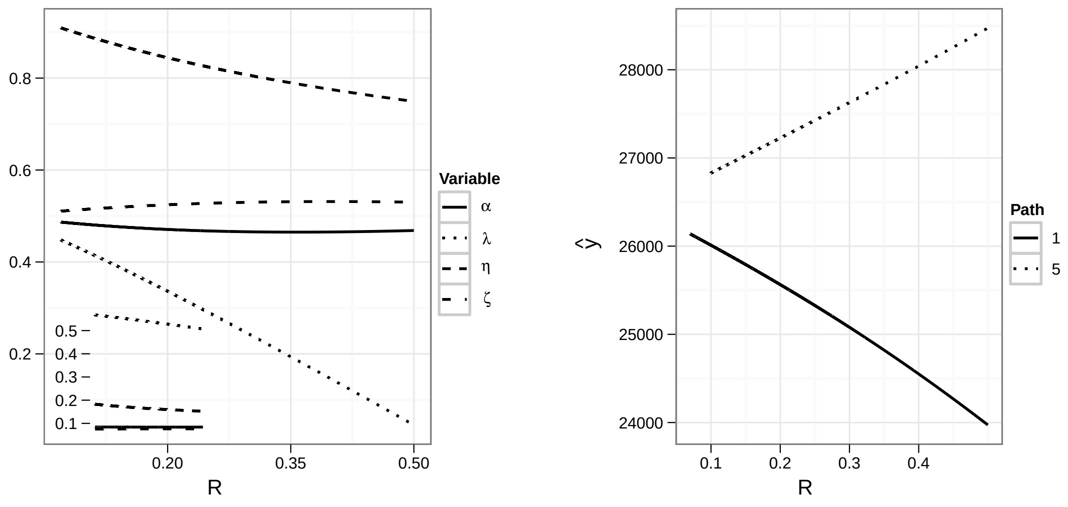
Figure 6.9: Contour Plots of the BBV model with $(\delta_V, \delta_E) = (1.2, 1.2)$

The summary of the minimum and maximum responses, and the normalised eigenvalues of the coefficient matrix of the second-order response surface are shown in Table 6.4. The ridge analysis was conducted within the domain $\Omega_{CA} = (\alpha, \lambda, \eta, \zeta)$, where $\alpha \in [0.07, 0.1], \lambda \in [0.5, 0.65], \eta \in [0.01, 0.2], \zeta \in [0.05, 0.1]$.

Table 6.4: Results of BBV DTAPs with SARSA(0) and WPL Policy with $\delta_V = 1.2, \delta_E = 1.2$

Variable	Minimum	Maximum	Eigenvalues
α	0.1	0.001	−272.05
λ	0.5	0.9	11.32
η	0.096	0.01	−49.96
ζ	0.1	0.001	−2353.08

The ridge analysis is conducted with the focal point defined to be $(0.0055, 0.67, 0.2, 0.075)$ and the results are presented in Figure 6.10. Two ridges are identified originating from the focal point (see Figure 6.10(b)). The solid line determines the path towards the global minimum with the respective change in the learning parameters presented in Figure 6.10(a). The most sensitive learning parameter is the discount factor, λ. The monotone increase of the discount factor indicates that the path aligns with the λ-axis towards the global minimum. The other learning parameters are stable over the extent of the design domain, Ω_{CA}.



(a) "Minimum $\hat{y}$" path for the independent variables (b) "$\hat{y}$" Ridges

Figure 6.10: Canonical Analysis for BBV DTAPs with SARSA(0) and WPL Policy with $\delta_V = 1.2, \delta_E = 1.2$

The second scenario with the weighted policy learner illustrates the learning performance under low queueing stress levels. Figure 6.11 shows the probability densities of both, the minimum and maximum responses on the Kriging surface and the density for a uniformly random decision policy at $(\delta_V, \delta_E) = (1.8, 1.8)$. As with the ϵ-greedy method and the same queueing stress levels presented in Section 6.2.2.1, both best and worst case parameterisations over the specified design domain exhibit a significant improvement over the uniformly random decision policy. Comparing the best calibration with the uniformly random decision policy shows an improvement of 22.5%. Additionally, the weighted policy learner improves over the ϵ-greedy method by 11.5%.

The contour plots of the pair-wise independent variables with respect to the total event processing time is presented in Figure 6.12. As before the policy gradient update factor, ζ, has very little impact on the response with respect to all other variables according to the contour plot.

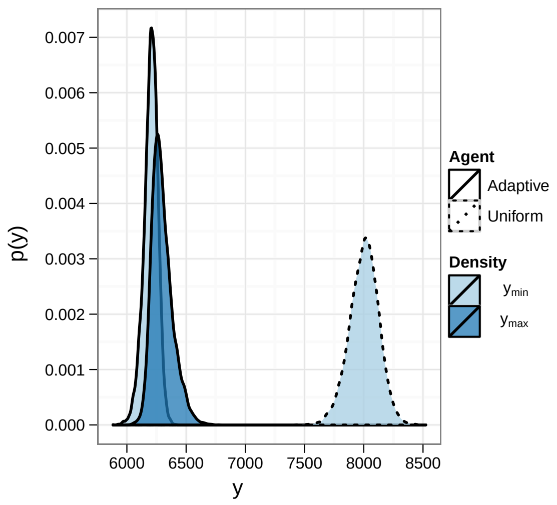


Figure 6.11: Probability Distributions of the minimum and maximum responses of the BBV model with $(\delta_V, \delta_E) = (1.8, 1.8)$

The summary of the minimum and maximum responses, and the normalised eigenvalues of the coefficient matrix of the second-order response surface are shown in Table 6.5. Given the values for the learning parameters that minimise the event processing time, the sub-domain for the canonical analysis is defined as $\Omega_{CA} = (\alpha, \lambda, \eta, \zeta)$, where $\alpha \in [0.001, 0.01], \lambda \in [0.5, 0.8], \eta \in [0.2, 0.5], \zeta \in [0.01, 0.1]$.

Table 6.5: Results of BBV DTAPs with SARSA(0) and WPL Policy with $\delta_V = 1.8, \delta_E = 1.8$

Variable	Minimum	Maximum	Eigenvalues
α	0.001	0.1	3.12
λ	0.572	0.9	−12.87
η	0.35	0.5	66.38
ζ	0.1	0.055	68.96

The ridge analysis is conducted with the focal point defined to be $(0.0055, 0.72, 0.2, 0.055)$ and the results are presented in Figure 6.13. Two ridges are identified originating from the focal point (see Figure 6.13(b)). The solid line determines the path towards the global minimum with the respective change in the learning parameters presented in Figure 6.13(a). In the original scale (subplot embedded in Figure 6.13(a)) the most sensitive learning parameter is the momentum, η, for the neural network function approximator. The increasing sensitivity of ζ with increasing radii of the sphere around the focal point can be attributed to the minimum ridge path being aligned with the ζ-axis. The other variables show a stable regime for $R > 0.4$.

6.2.2.3 Collaborative Function Approximators

This section investigates the effects of collaborative function approximators to neighbouring agents for the SARSA(0) reinforcement learning method with the ϵ-greedy policy and the weighted policy learner over task network structures modelled using the BBV model. As argued in Section 5.2.1.3, this allows the function approximator (here a feedforward backpropagation neural network) to be

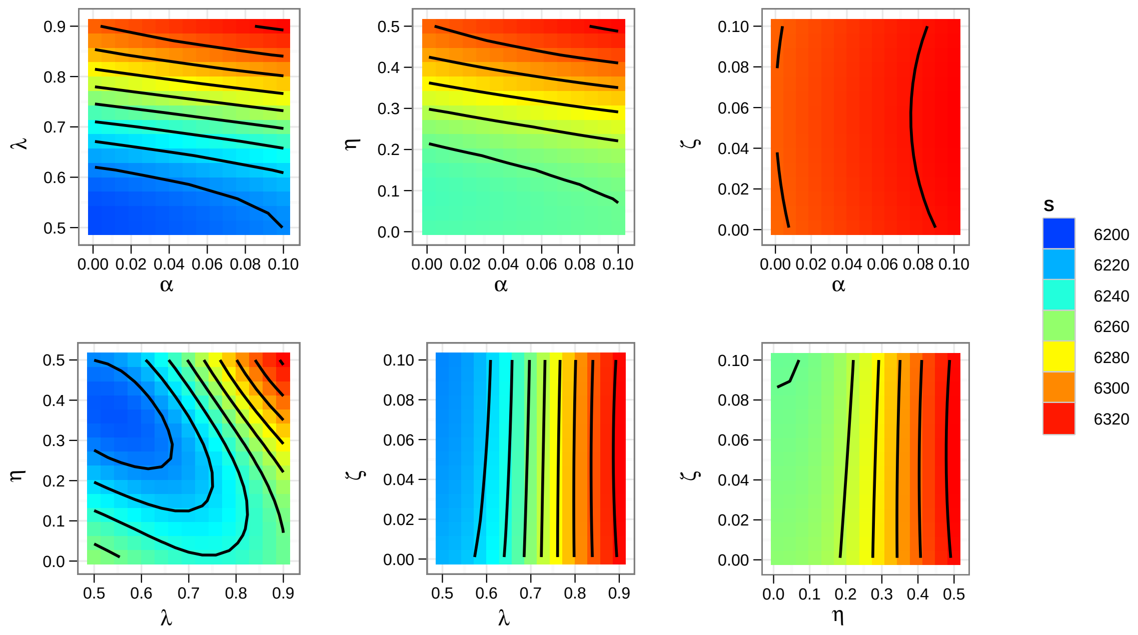


Figure 6.12: Contour Plots of the BBV model with $(\delta_V, \delta_E) = (1.8, 1.8)$

trained continuously for a given target node, rather than only when transactions are recorded for a given source and target node. This learning architecture is expected to adapt faster to changes in the environment, because all agents using a given target node can immediately take advantage of the most recent information. Additionally, the decisions by agents are based on the current delay in the queue of the nodes downstream.

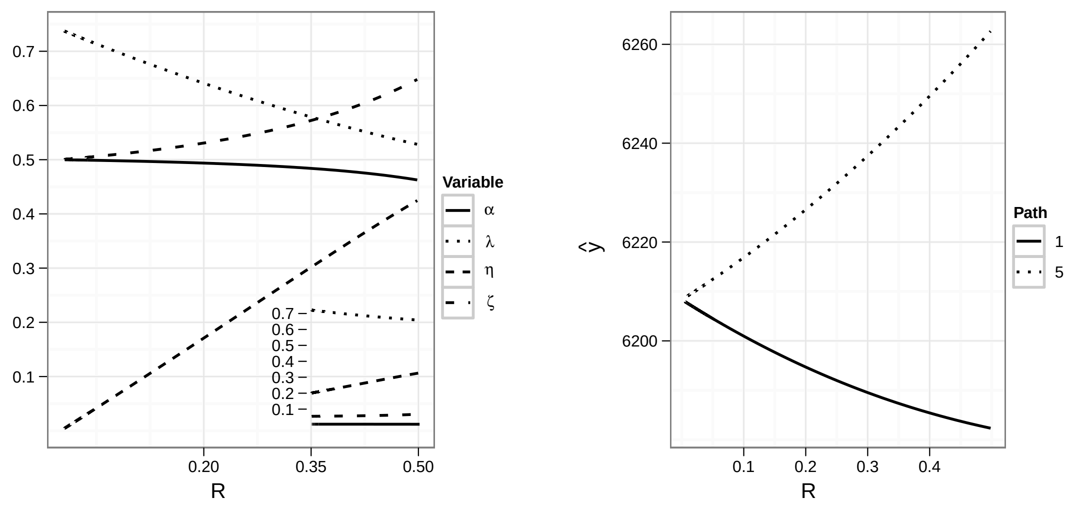
The state representation takes into account the delay in the queue, because the delay is a measurable state variable that directly contributes to the response time of a given task. Other queueing performance measures can be taken into account as well, but to be able to compare to previous scenarios, the state representation is the same.

First the SARSA(0) reinforcement learning method with the ϵ-greedy policy is investigated. The task network is generated using the BBV model with $(\delta_V, \delta_E) = (1.2, 1.2)$. The global minimum and maximum of the Kriging function was found to be 23902.1682(26419.46) and 24182.99(56825.53) time units respectively (estimated MSE (2.17) in brackets). A summary of the minimum, maximum, and the eigenvalues of the response surface is presented in Table 6.6.

The characteristics of the density curves presented in Figure 6.14 are similar to that presented in Figure 6.2 with almost exactly the same mean responses. Thus, collaborative function approximators do not exhibit a discernible impact near the vicinity of the minimum response in this case.

The contour plots of the pair-wise independent variables with respect to the total event processing time is presented in Figure 6.15. The contours resemble the characteristics of the ones presented in Figure 6.3. So, overall the equilibrium performances of the scenarios with and without collaborative function approximation does not differ significantly.

While the event processing time of this scenario is almost the same as the one with individual function approximators, the values of the learning parameters are significantly different. The learning rate is by an order of a magnitude higher, the discount factor is reduced from 0.9 to 0.69,



(a) "Minimum $\hat{y}$" path for the independent variables

(b) "$\hat{y}$" Ridges

Figure 6.13: Canonical Analysis for BBV DTAPs with SARSA(0) and WPL Policy with $\delta_V = 1.8, \delta_E = 1.8$

Table 6.6: Results of BBV DTAPs with SARSA(0) and ϵ-greedy Policy with $\delta_V = 1.2, \delta_E = 1.2$ and Collaborative Function Approximators

Variable	Minimum	Maximum	Eigenvalues
α	0.03	0.1	−16.81
λ	0.69	0.58	−32.39
η	0.13	0.5	−178.52

and the momentum for the function approximator is reduced from 0.27 to 0.13. This means, that for collaborative function approximators to operate optimally, the uptake of information is increased, the rewards are more heavily discounted, and the trajectory of weight changes of the neural network training algorithm is lower.

The extent of the sensitivities, however, mirrors the results presented in Figure 6.4 for the same scenario with independent function approximators for each action. The graphical presentation of the sensitivities is plotted in Figure 6.16.

The sub-domain for the canonical analysis is defined as $\Omega_{CA} = (\alpha, \lambda, \eta)$, where $\alpha \in [0.01, 0.1], \lambda \in [0.6, 0.8], \eta \in [0.01, 0.3]$ and includes the parameter settings that yield the minimum response. The focal point of the ridge analysis was set to be $(0.075, 0.79, 0.28)$. In the scenario with independent function approximators the optimal discount factor is obtained at 0.9, which resulted in a stationary ridge along the λ axis. This is not the case with collaborative function approximators. Visual inspection of the first and second contour plots in Figure 6.15 verifies this result.

The second scenario investigating the collaborative function approximation uses the WPL policy instead of the ϵ-greedy one. The global minimum and maximum of the Kriging function was found to be 24014.75(101780.80) and 28018.6(161636.72) time units respectively (estimated MSE (2.17) in brackets). A summary of the minimum, maximum, and the eigenvalues of the response surface is presented in Table 6.7.

The density curves presented in Figure 6.17 show that collaborative function approximation

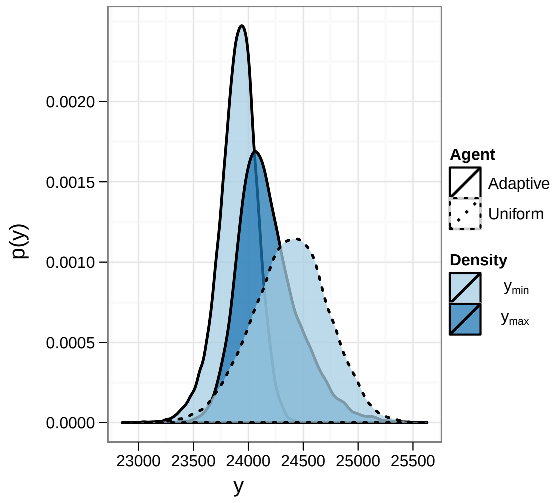


Figure 6.14: Probability Distributions of the minimum and maximum responses of the BBV model with $(\delta_V, \delta_E) = (1.2, 1.2)$ and Collaborative Function Approximators

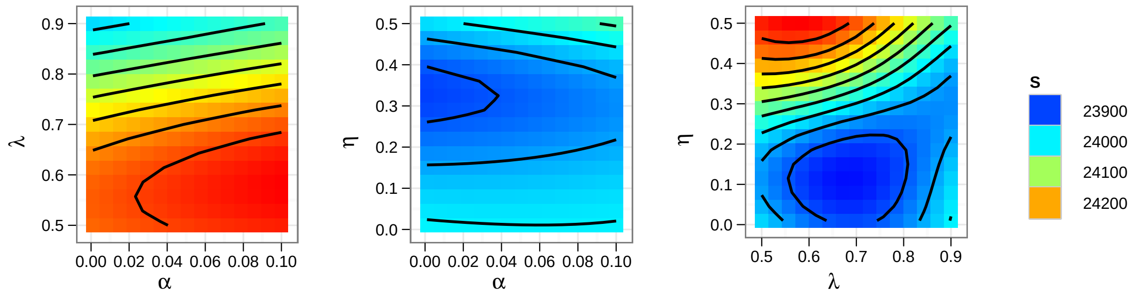


Figure 6.15: Contour Plots of the BBV model with $(\delta_V, \delta_E) = (1.2, 1.2)$ and Collaborative Function Approximators

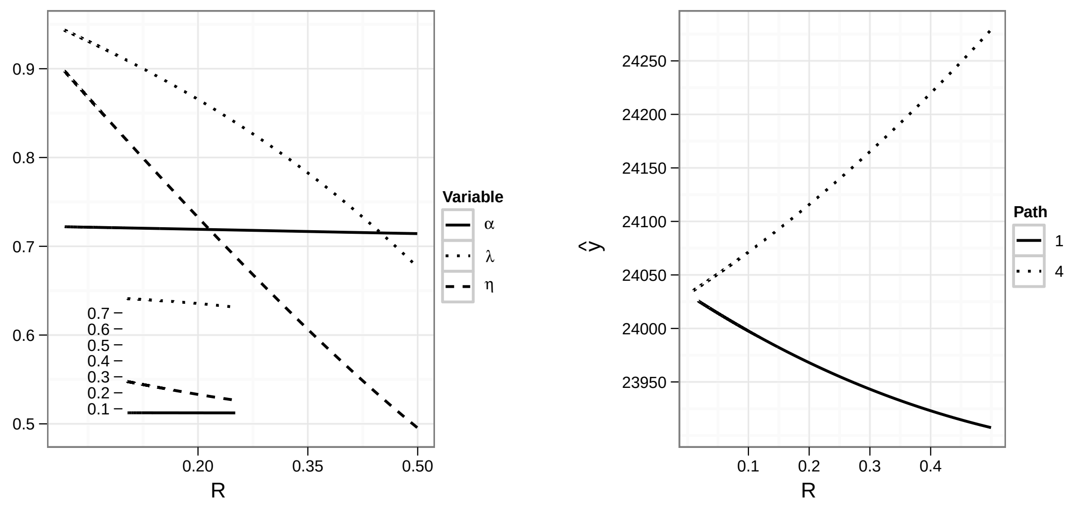
does not improve the system performance over independently learning the action-value function. This is quite surprising, since the weighted policy learner in covered in Section 6.2.2.2 outperforms the uniformly random decision policy.

The contour plots of the pair-wise independent variables with respect to the total event processing time is presented in Figure 6.18. The dependence of the variables to each other shows that the optimal total average event processing time is attained with $\alpha \to 0.001$, $\lambda \to 0.9$, $\eta \sim 0.3$, and $\zeta \sim 0.1$. The overall results are summarised in Table 6.7.

Table 6.7: Results of BBV DTAPs with SARSA(0) and WPL Policy with $\delta_V = 1.2, \delta_E = 1.2$ and Collaborative Function Approximators

Variable	Minimum	Maximum	Eigenvalues
α	0.001	0.1	−664.92
λ	0.9	0.9	−526.96
η	0.33	0.5	354.96
ζ	0.1	0.001	−1919.37

While the event processing time of this scenario is almost the same as the one with individual function approximators, the values of the learning parameters are significantly different, which



(a) "Minimum ŷ" path for the independent variables (b) "ŷ" Ridges

Figure 6.16: Canonical Analysis for BBV DTAPs with SARSA(0) and ϵ-greedy Policy with $\delta_V = 1.2, \delta_E = 1.2$ and Collaborative Function Approximators

illustrates that collaboratively learning the action-value function results in an entirely different interpretation of the learning parameters. The learning rate is by two orders of a magnitude higher, the discount factor is reduced from 0.9 to 0.5, and the momentum for the function approximator is increased from 0.096 to 0.33. This means, that for collaborative function approximators to operate optimally, the uptake of information is increased, the rewards are more heavily discounted, and the trajectory of weight changes of the neural network training algorithm is higher.

The ridge analysis in Figure 6.19 shows a stable regime for the learning rate, the momentum, and the gradient update factor of WPL. The discount factor monotonously increases as the radius of the sphere around the focal point of $(0.0055, 0.825, 0.23, 0.08)$, which indicates that the minimum ridge aligns with the λ axis.

The sub-domain for the canonical analysis is defined as $\Omega_{CA} = (\alpha, \lambda, \eta, \zeta)$, where $\alpha \in [0.001, 0.01], \lambda \in [0.75, 0.9], \eta \in [0.2, 0.4], \zeta \in [0.06, 0.1]$. Overall the results obtained from this section are not favourable of collaborative function approximation, which intuitively should have improved the learning behaviour. One explanation of this result is that the task network is static and the queueing characteristics are homogeneous. More dynamic settings may show an entirely different result and therefore would be interesting to explore in future works. It also indicates that the scenario with independent function approximation learns optimal policies quite fast and settles to good solutions. Exterior pressure onto a system by introducing heterogeneous Poisson arrival rates that challenge the agents individually may favour a setup where function approximators are trained collaboratively. Also, the design domain may be the limiting factor of finding better solutions for the weighted policy learner with collaborative function approximators.

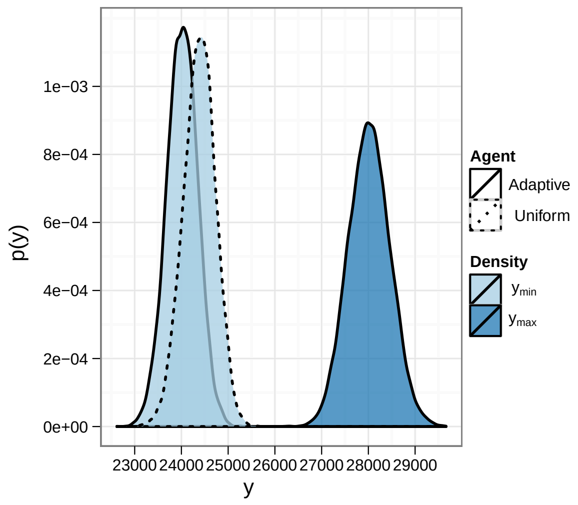


Figure 6.17: Probability Distributions of the minimum and maximum responses of the BBV model with $(\delta_V, \delta_E) = (1.2, 1.2)$ and Collaborative Function Approximators

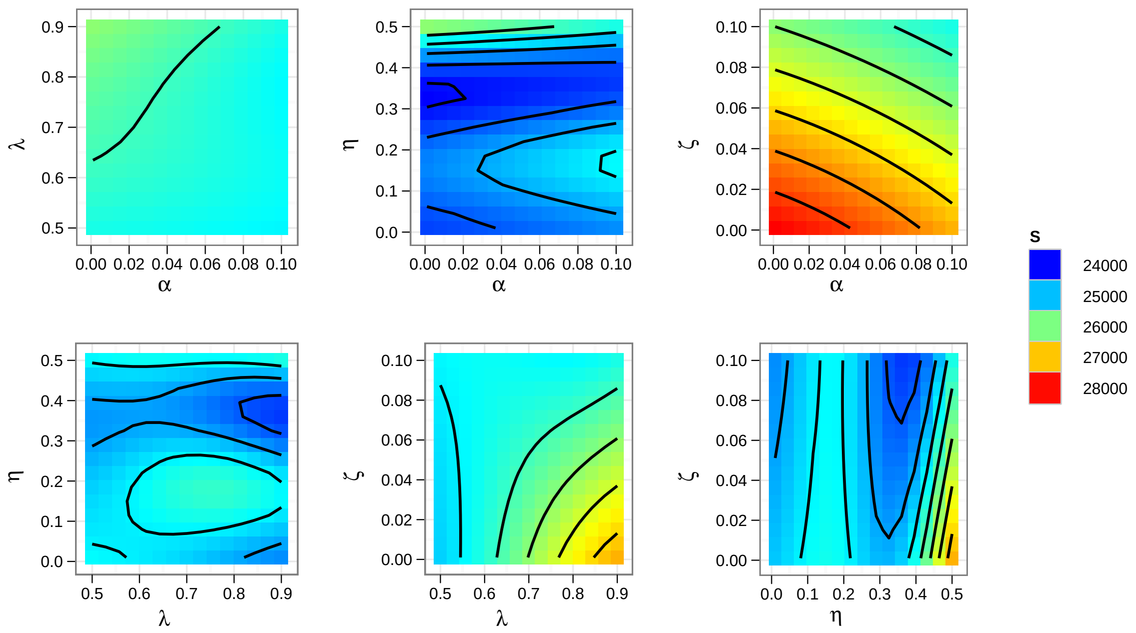
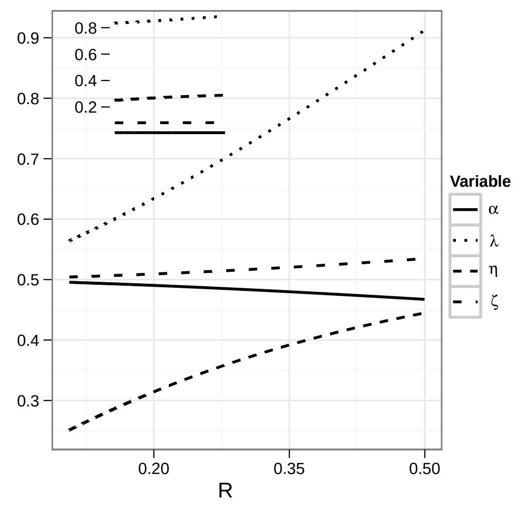
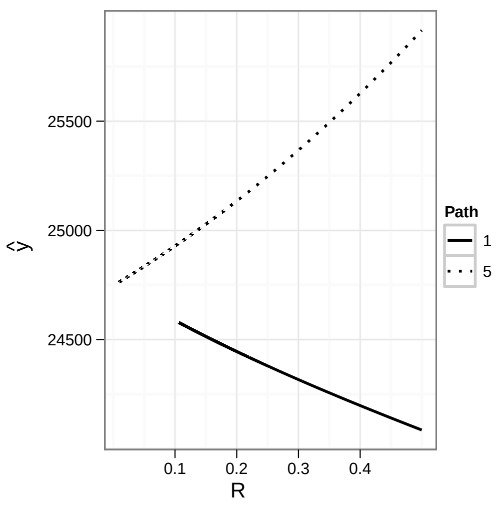


Figure 6.18: Contour Plots of the BBV model with $(\delta_V, \delta_E) = (1.2, 1.2)$ and Collaborative Function Approximators



(a) "Minimum $\hat{y}$" path for the independent variables



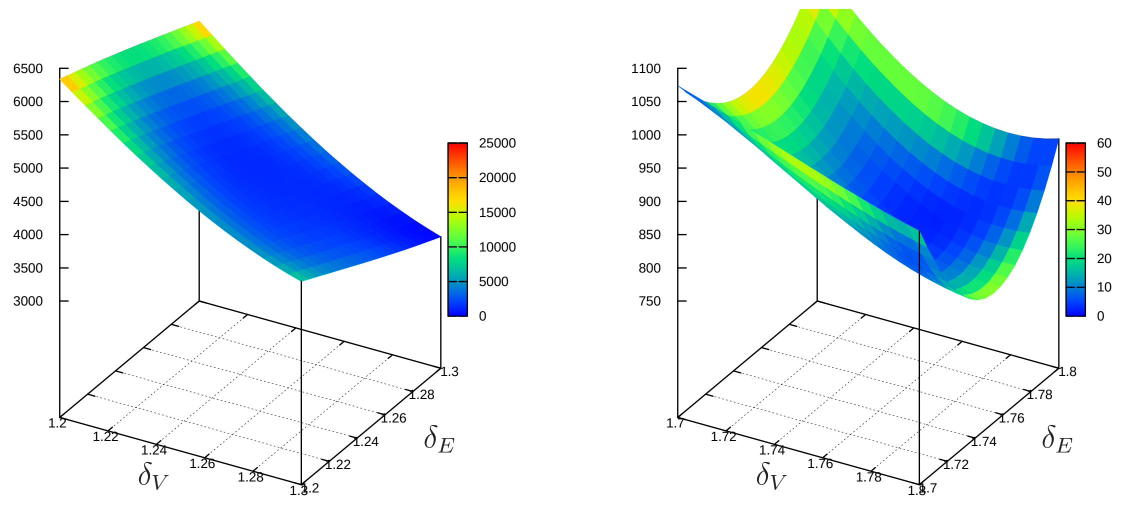
(b) "$\hat{y}$" Ridges

Figure 6.19: Canonical Analysis for BBV DTAPs with SARSA(0) and WPL Policy with $\delta_V = 1.2, \delta_E = 1.2$ and Collaborative Function Approximators

6.3 DTAPs with the ER Model

6.3.1 Uniformly Random Transition Probabilities

Analogously to the analysis in Section 6.2.1 this section details the results of distributed task assignments with uniform agents over random ER networks (details of the evolution model can be found in Section 3.2.3). Again, two designs are evaluated with $(\delta_V, \delta_E) \in [1.2, 1.3]$ and $(\delta_V, \delta_E) \in [1.7, 1.8]$. The response surfaces of the event processing time with 19 and 28 samples for the first and second designs respectively are presented in Figure 6.20. The stochastic nature of the simulations yield slightly tilted surfaces, that do not reflect the analytical solution perfectly, but similarly to the BBV model illustrate the uniform agent behaviour well enough to act as a baseline test against intelligent decision making processes.



(a) With $\delta_V \in [1.2, 1.3]$ and $\delta_E \in [1.2, 1.3]$ (b) With $\delta_V \in [1.7, 1.8]$ and $\delta_E \in [1.7, 1.8]$

Figure 6.20: Event Processing Time for the BBV Model

As with the analytical model of the respective queueing behaviour over the random network topology presented in Section 3.3.3 the event processing time decreases with both increasing values for the constants δ_V and δ_E. The respective minima and maxima are presented in Table 6.8.

Table 6.8: Results of ER DTAPs with a uniformly random decision Policy

	$\delta_V, \delta_E \in [1.2, 1.3]$		$\delta_V, \delta_E \in [1.7, 1.8]$	
Variable	Minimum	Maximum	Minimum	Maximum
δ_V	1.3	1.2	1.8	1.7
δ_E	1.3	1.2	1.75	1.8

6.3.2 Adaptive Transition Probabilities

Similarly to Section 6.2.2, this section examines the effect of the choice of learning policies employed for SARSA(0) reinforcement learning agents. These encompass the ϵ-greedy policy (evaluated in Section 6.3.2.1) and the weighted policy learner (evaluated in Section 6.3.2.2).

6.3.2.1 SARSA(0) with ϵ-greedy Policy

Analogously to the simulation setup in Section 6.2.2 the SARSA(0) reinforcement learning method with the ϵ-greedy policy is evaluated in this section using a homogeneous network topology. The Kriging metamodel is fitted for the total event processing time covering the domain $\Omega = (\alpha, \lambda, \eta)$, where $\alpha \in [0.001, 0.1], \lambda \in [0.5, 0.9], \eta \in [0.01, 0.5]$ with a total of 37 simulations. As in the previous scenarios, the ϵ-greedy policy is set to $\epsilon = 0.05$. The global minimum and maximum of the Kriging function was found to be 5469.92(5315.96) and 6807.4288(13967.17) time units respectively (estimated MSE (2.17) in brackets). A summary of the minimum, maximum, and the eigenvalues of the response surface are presented in Table 6.9. Given the best parameterisation of the learning method, the sub-domain for the canonical analysis is defined as $\Omega_{CA} = (\alpha, \lambda, \eta)$, where $\alpha \in [0.001, 0.01], \lambda \in [0.6, 0.8], \eta \in [0.1, 0.4]$.

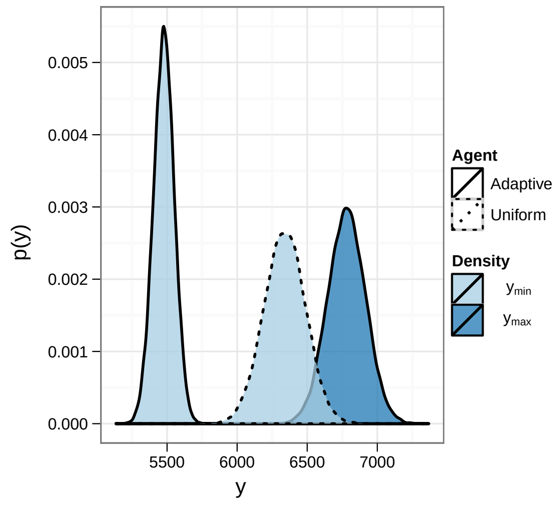


Figure 6.21: Probability Distributions of the minimum and maximum responses of the ER model with $(\delta_V, \delta_E) = (1.2, 1.2)$

Figure 6.21 shows the densities of the simulated best and worst case parameterisations of the multi-agent reinforcement learning methods with an underlying random network structure. The best setting for the SARSA(0) reinforcement learning method with an ϵ-greedy policy shows a 24% improvement over the worst setting and 16% improvement over the uniformly random decision policy. Interestingly, the worst possible parameterisation within the boundaries of the learning parameters yields a result that does no longer outperform uniformly random decision policies.

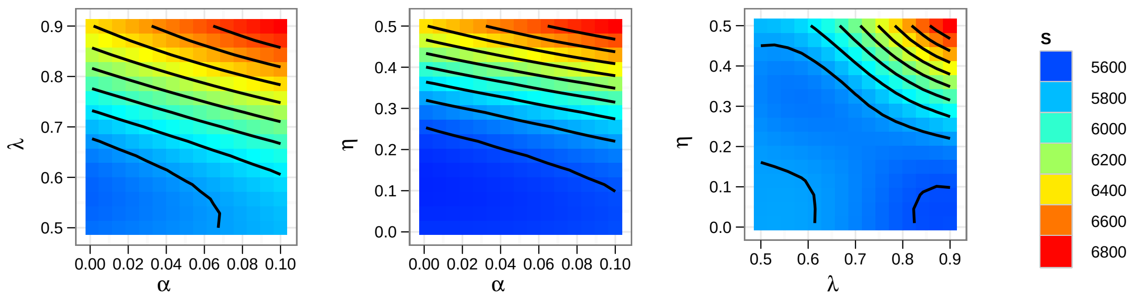


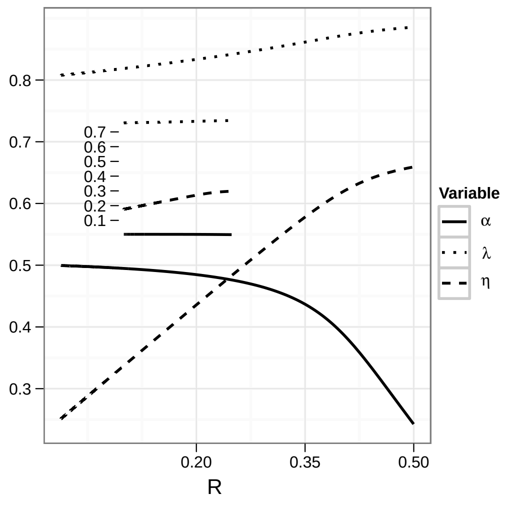
Figure 6.22: Contour Plots of the ER model with $(\delta_V, \delta_E) = (1.2, 1.2)$

The contour plots of the pair-wise independent variables with respect to the total event processing time is presented in Figure 6.22. The first two contour plots indicate that the minimum response is achieved in the lower left domain of each plot, i.e., for $\alpha \to 0.001, \lambda < 0.7, \eta < 0.3$. The surface behaviour of λ versus η in the last plot suggests a region with minimum response as a linearly decreasing relationship between both independent variables.

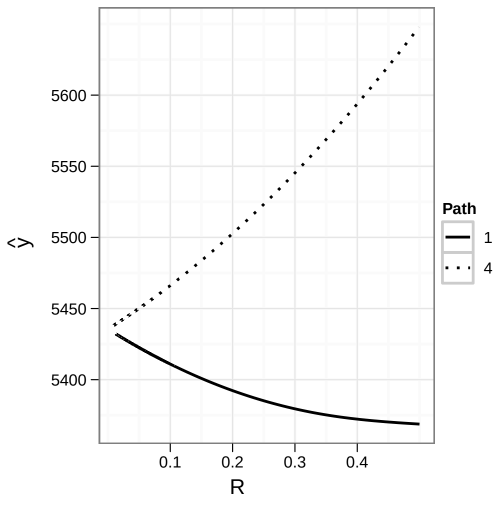
Table 6.9: Results of ER DTAPs with SARSA(0) and ϵ-greedy Policy with $\delta_V = 1.2, \delta_E = 1.2$

Variable	Minimum	Maximum	Eigenvalues
α	0.001	0.1	7.8
λ	0.6612	0.9	−549.18
η	0.3215	0.5	−1161.86

The ridge analysis plotted in Figure 6.23. Tracing the minimum ridge starting from the focal point $(0.0055, 0.76, 0.17)$ of the design domain towards the global minimum shows two different regimes, where the first 4/5 of the trace shows a steeper slope for the momentum, low sensitivities for the discount factor, and almost no measurable impact for the learning rate. This behaviour is reversed in the last fifth of the trace, with the discount factor and the momentum levelling off and the learning rate being responsible for most of the change in the response measurement. The behaviour of the learning rate is particularly interesting. It remains virtually constant for $R < 0.4$, and then decreases sharply. When one compares the minimum ridge in Figure 6.23(b) with this sharp decrease, one notices that the response value levels off simultaneously. So, the decrease in the learning rate for $R > 0.4$ has no noticeable effect on the response. Therefore, the learning rate is not critical to the optimisation. This curious behaviour, however, can be explained by the fact that the stationary ridge must be almost exactly along the α axis, which can be verified by looking at the first two contour plots of Figure 6.22.



(a) "Minimum $\hat{y}$" path for the independent variables



(b) "$\hat{y}$" Ridges

Figure 6.23: Canonical Analysis for ER DTAPs with SARSA(0) and ϵ-greedy Policy with $\delta_V = 1.2, \delta_E = 1.2$

Comparing the ER and BBV models and their effect on the SARSA(0) temporal-difference reinforcement learning method with an ϵ-greedy policy shows that the value for the learning rate is at the lower end of the design domain with a value of 0.001. Additionally, the learning rate exhibits a lower impact on the response measurement within the domain of the ridge analysis than anticipated. The character of the discount factor, λ, is significantly different between both network structures. In the BBV model, the discount factor is the most sensitive to changes and attains an optimal value of 0.9 with an optimal value of 0.66 in the ER model.

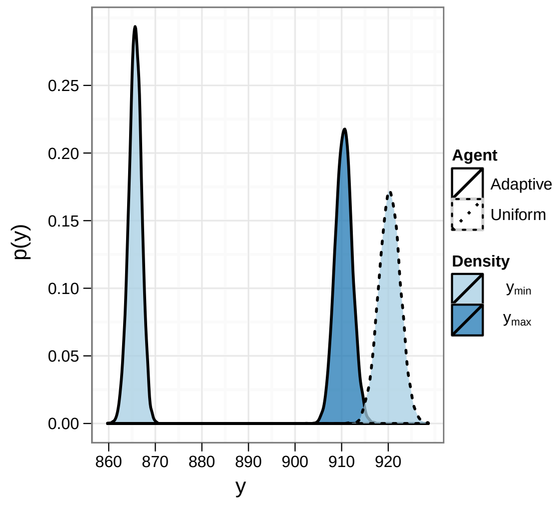


Figure 6.24: Probability Distributions of the minimum and maximum responses of the ER model with $(\delta_V, \delta_E) = (1.8, 1.8)$

As in previous scenarios the effect of stress on queueing behaviour is investigated with the same learning domain as above. The global minimum and maximum of the Kriging function was found to be 865.536(1.573) and 910.662(2.959) time units. Figure 6.24 presents the probability densities of both, the minimum and maximum responses on the Kriging surface and the density for a uniformly random decision policy at $(\delta_V, \delta_E) = (1.8, 1.8)$. The density plot reveals that calibration within the design domain has a significant impact on the learning performance. Both, best and worst case parameterisations result in significant better performance compared to a uniformly random decision policy. A summary of the minimum, maximum, and the eigenvalues of the response surface is presented in Table 6.10.

Table 6.10: Results of ER DTAPs with SARSA(0) and ϵ-greedy Policy with $\delta_V = 1.8, \delta_E = 1.8$

Variable	Minimum	Maximum	Eigenvalues
α	0.001	0.1	0.358
λ	0.745	0.9	10.49
η	0.373	0.5	-18.9

The contour plots are presented in Figure 6.25 indicate that an optimal response is attained for $\alpha \to 0.001, \lambda \sim 0.7$ and $\eta \sim 0.35$. In contrast the worst response can easily be identified at $(0.1, 0.9, 0.5)$.

The ridge analysis plotted in Figure 6.26 given the sub-domain $\Omega_{CA} = (\alpha, \lambda, \eta)$, where $\alpha \in [0.001, 0.01], \lambda \in [0.6, 0.8], \eta \in [0.2, 0.4]$ and the focal point $(0.0055, 0.65, 0.27)$. Tracing along the

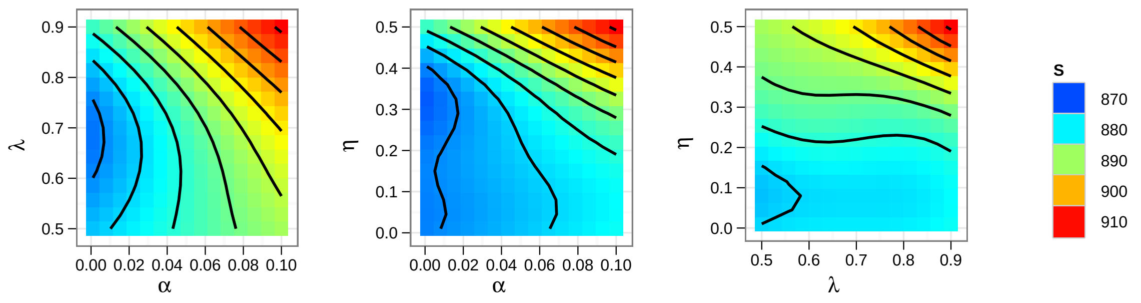
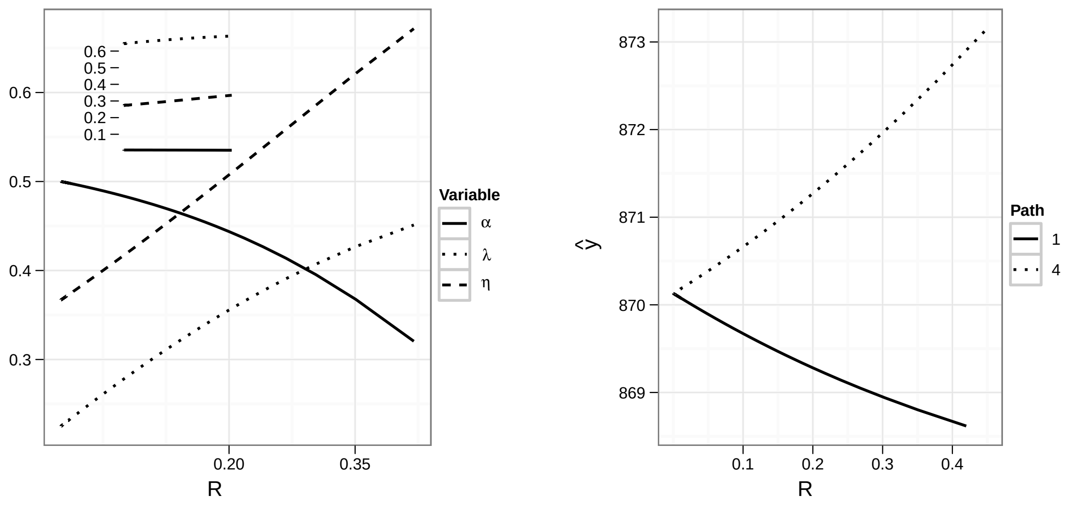


Figure 6.25: Contour Plots of the ER model with $(\delta_V, \delta_E) = (1.8, 1.8)$

minimum ridge illustrates that the momentum has the most impact on the response followed by the discount factor and the learning rate. Within the domain of the canonical analysis, the learning rate has the least impact on the response.



(a) "Minimum $\hat{y}$" path for the independent variables (b) "$\hat{y}$" Ridges

Figure 6.26: Canonical Analysis for ER DTAPs with SARSA(0) and ϵ-greedy Policy with $\delta_V = 1.8, \delta_E = 1.8$

6.3.2.2 SARSA(0) with WPL

This section only covers the weighted policy learner under low queueing stress levels.

As in all scenarios before, Figure 6.27 presents the probability densities of both, the minimum and maximum responses on the Kriging surface and the density for a uniformly random decision policy at $(\delta_V, \delta_E) = (1.8, 1.8)$. This scenario is the first in this chapter where the intelligent agents are not able to learn a better policy than the uniformly random decision policy. Both the best- and worst-case calibrations over the design domain show a decrease in the total event processing time by 0.8% and 4.7% respectively.

The summary of the minimum, maximum, and the eigenvalues of the response surface is given in Table 6.11.

The contour plots in Figure 6.28 show that the update factor of the gradient vector of the

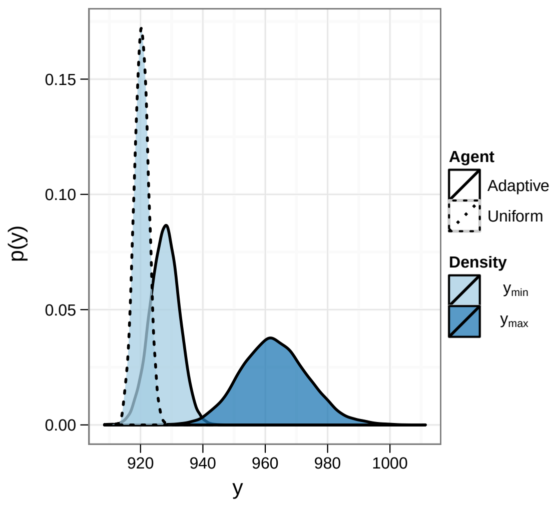


Figure 6.27: Probability Distributions of the minimum and maximum responses of the ER model with $(\delta_V, \delta_E) = (1.8, 1.8)$

Table 6.11: Results of ER DTAPs with SARSA(0) and WPL Policy with $\delta_V = 1.8, \delta_E = 1.8$

Variable	Minimum	Maximum	Eigenvalues
α	0.1	0.001	−28.88
λ	0.52	0.88	−14.73
η	0.01	0.01	8.36
ζ	0.1	0.074	16.77

weighted policy learner, ζ, has little impact on the response, while η and λ have the highest impact. This result is also reflected in the ridge analysis below.

The ridge analysis plotted in Figure 6.29 given the sub-domain $\Omega_{CA} = (\alpha, \lambda, \eta, \zeta)$, where $\alpha \in [0.01, 0.1], \lambda \in [0.5, 0.9], \eta \in [0.01, 0.5], \zeta \in [0.01, 0.1]$. Tracing along the minimum ridge (path 1 in Figure 6.29(b)) illustrates that the discount factor and the momentum of the function approximator are most sensitive to changes.

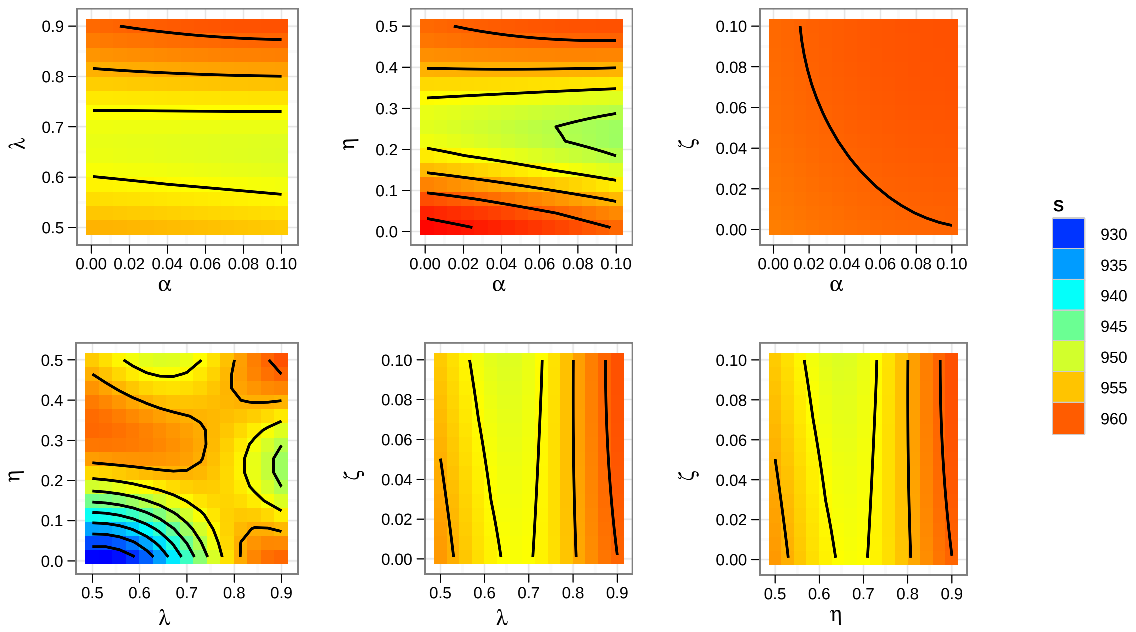
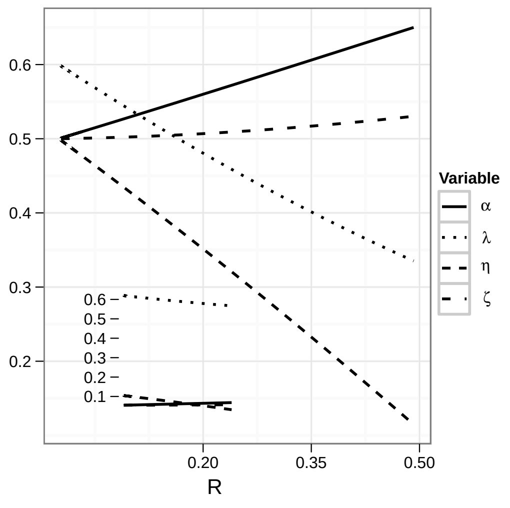
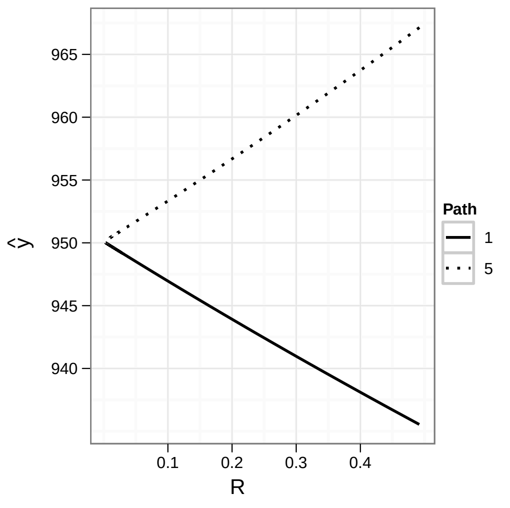


Figure 6.28: Contour Plots of the ER model with $(\delta_V, \delta_E) = (1.8, 1.8)$





(a) "Minimum ŷ" path for the independent variables (b) "ŷ" Ridges

Figure 6.29: Canonical Analysis for ER DTAPs with SARSA(0) and WPL Policy with $\delta_V = 1.8, \delta_E = 1.8$

6.4 Summary

This chapter presented a thorough study on how to find optimal parameters for reinforcement learning methods. Although the apparatus of response surface methodology can be applied to any computer experiment, it has not been fully exploited in the realm of machine learning and its application to distributed computing. In this sense, the results obtained in this chapter provide novel insights in the behaviour of multi-agent reinforcement learning as applied to distributed task assignment problems. An extensive comparative investigation was conducted to analyse the effect of different network topologies, queueing stress levels, and learning policies on the learning behaviour. To aid the analysis visual representations of the impact of the independent learning parameters was provided in form of contour plots. Supporting this, a ridge analysis was conducted throughout to examine the sensitivities of the independent variables in the vicinity of the optimal response.

Based on this extensive empirical study the following observations can be derived:

Observation 1. SARSA(0) *reinforcement learning with neural network function approximators to estimate the state-action value function outperforms uniformly random decision policies in most cases.*

This observation holds under the following assumptions.

Assumption 1.1. *The neural network function approximator employs a standard backpropagation training method that minimises the mean-squared error of the estimated output and the observed output. To facilitate continuous and potentially infinite learning horizons a moving window of the last* 100 *errors is used as a mean-squared error estimation.*

Assumption 1.1 provides a trade-off between keeping the full history of all errors, which is impractical for infinite learning horizons, and 1-step errors, which would introduce fluctuations in the error surface in weight space. Also, throughout the experiments the architecture of the neural network was fixed.

Assumption 1.2. *The neural network (more specifically, the multi-layer perceptron) has one input neuron for the state signal, one hidden layer with* 8 *neurons, and one output unit for the* Q*-value estimation. It follows, that neural networks are co-located with each action (here equivalent to an arc in the digraph of the task network).*

Maintaining individual function approximators for each action in the task network avoids interferences in training the neural network from exercising unrelated actions and consequently reduces potential instability [3].

Observation 2. *The structural difference of the task topology gives rise to different best parameterisations of the learning method.*

Observation 2 implies that any deployment of learning algorithms into real-world scenarios requires a careful study of the task topology and tuning the learning method respectively.

The induced queueing stress level also has a major impact on the learning performance and exhibits a different nature of the global behaviour of a prescribed learning method.

Observation 3. *A task network with relatively low stress levels provides more latitude for reinforcement learning agents to adapt, with the consequence that worst-case calibrations outperform uniformly random decision policies.*

Observation 3 is a stronger statement than the more general Observation 1. This observation is encouraging, because it alludes to a general indication that adaptive agents outperform fixed policies given careful calibration under high load. In light of the results obtained by Schaerf et al. [134], where fixed load scenarios favour fixed decision strategies, the fixed decision strategy chosen for the experiments in this thesis is based on uniformly random action selections. Such a strategy does not embed any domain knowledge of arrival rates and capabilities of each node to service the incoming requests. However, as queueing networks become large centralised optimisation to find fixed strategies becomes prohibitively expensive.

Assumption 3.1. *The load characteristics for the task networks are defined by prior probabilities upon establishing the task network and are fixed throughout the simulation.*

Assumption 3.2. *The topology of the task network is considered fixed once established.*

Assumption 3.3. *The stress levels are induced using two factors defined by the network evolution models for queueing networks.*

While assumptions Assumption 3.1 - Assumption 3.3 do not deter intelligent agents from finding better policies than a static non-adaptive one, adaptive systems are more suitable to dynamic environments. As such, future work will investigate non-homogeneous load configurations.

Observation 4. *The weighted policy learner by Abdallah and Lesser [2] is superior to the ϵ-greedy policy in most scenarios.*

Observation 4 is not surprising, because the weighted policy learner takes the relative performance of actions into account and adapts the selection probabilities accordingly. Additionally, it is in a sense risk-averse. The learning behaviour of WPL has two distinct qualities. As updates in the policy space are following the same trajectory, learning gradually slows until a stable stochastic policy has been reached. Otherwise, the learning behaviour is accelerated. The nature of this behaviour needs to be further examined to find out whether WPL agents are constantly in the fast learning phase or not. Empirically, this can be achieved with the related fair action policy by Zhang et al. [187].

Observation 5. *Collaborative Function Approximators do not show an improvement over individually training the action-value function.*

Assumption 5.1. *Collaborative action-value function approximators reside with destination nodes, rather than with each action.*

In distributed task networks a directed arc from node u to node v implies that a task being processed by u requires v to contribute to the completion of the task (see Definition 5.6). Independent action-value function approximators are instantiated with each arc, (u, v), in the task network and are maintained by the head node of this arc. By co-locating action-value function approximators with the tail node instead, all nodes that have an arc to this tail node collaboratively train the function approximator. As a consequence all nodes $\forall\ u\ \exists\ (u, v)$ share their action-value mapping.

Assumption 5.2. *Since collaborative function approximators are co-located with the tail nodes of each arc, it implies that the state signals are local to the approximator.*

Assumption 5.2 implies that state signals local to the head node cannot be included in the state representation of the function approximator anymore. In queueing networks it is often beneficial to incorporate the state signals of the tail node's queueing performance metrics, because those metrics are more informative for action selection than local queueing metrics.

Observation 6. *The learning rate of the reinforcement learning agents, α, and the gradient update factor of WPL, ζ, are the least sensitive to changes near the vicinity of the optimum response.*

Assumption 6.1. *Both, the learning rate and the gradient update factor of WPL cover two orders of magnitude in their respective range ($\alpha, \zeta \in [0.001, 0.1]$). The vicinity of the stationary point only covered a subset of the initial range for the experimental design.*

In most of the scenarios covered in this chapter, the learning rate and the gradient update factor attained values near or at the edge of the specified design domain. However, since both values showed a relative low sensitivity near the optimal value, an increased range of the design domain was not considered.

The next chapter extends to some of the observations by conducting an analysis of the non-linear dynamics of the learning methods employed.

— *It makes no sense to read a newspaper with a microscope.*

Valentino Braitenberg

7 Non-linear Dynamics of Multi-agent Learning

Moving from single-agent to multi-agent environments entails a host of issues that are very challenging to deal with. First, the fundamental convergence properties of single-agent reinforcement learning algorithms are violated, because agents co-adapt. Therefore, a distinction between learning and teaching cannot be made. Any action decision by an agent is a result of the other agents' past behaviour and has implications for their future behaviours. That means that the environment is non-stationary. To tackle the non-stationary environment, agents are ill-advised to settle on deterministic policies, because it cancels out adaptive forces still underway in the environment. Additionally, in the context of queueing systems asymptotically greedy policies are constrained by the stability criterion (3.3) which requires global knowledge to compute. Assuming an agent had access to the stability characteristics of the queueing network and could foresee that greedy actions have no detrimental effects on the network, it is in practice not desirable to have, for example, deterministic load-balancing or routing policies. Consequently, stochastic policies are often required to maintain a certain level of exploration to be able to discover congestion along one path and being able to unlearn previously considered good policies.

Analysing complex networks mostly comprises three levels. The lowest level is concerned with the topological analysis of the network only taking into account the dichotomous nature in which edges are either present or absent from the network [53, 110, 171]. The second level adds a distinction of the edges in terms of weight assigned to them and also the direction of the edges [5, 6, 16, 17, 91, 98, 109, 114, 133, 154, 168]. The third level accounts for a fitness or strength measure for each vertex in the network. To investigate how adaptation of individual vertices evolve and impact the network as a whole, the time-dependent edge and vertex dynamics are studied using snapshots of the networks.

This chapter presents three approaches on analysing the non-linear behaviour giving snapshots of the global state of the distributed task hierarchy. Section 7.1 analyses the queueing dynamics within the framework of queueing theory. Section 7.2 uses expert metrics derived from the reward received over a given time period to investigate immediate experiences of the agents. Section 7.3 takes a network view on the non-linear dynamics, in that the time-dependent dynamics of the arcs in the task network are investigated using techniques from social network analysis. In doing so, the rate of processing requests is integrated into the arcs instead of maintaining a probabilistic view on the arcs as in queueing theory. In particular, the influences of the agents to each other is studied.

7.1 Queueing Dynamics

The dynamics of queueing performance measures can be captured in several ways. The first metric defines a quantity that captures learning as distinct from random behaviour. This expresses itself in the matrix of routing probabilities $\mathbf{Q}$ (see Definition 3.1).

The distance measure from random behaviour is denoted as

$$d_n = \|\mathbf{Q}_{(n)} - \mathbf{Q}^r_{(n)}\|_1, \ \forall\, n \in \mathcal{V} \,\&\, \mathtt{deg}^+(n) > 0, \tag{7.1}$$

where $\mathbf{Q}^r_{nj} = \frac{1}{\mathtt{deg}^+(n)}$ is the probability of taking a uniformly random action for all actions j available to agent (or node in queueing terminology) n. The probability of taking actions $\mathbf{Q}_{(n)}$ is derived from the employed policy. This measure is bounded by $d_n = 0$, if the action selection probabilities are uniformly random, and $\sup\{d_n\} = 2$ for deterministic action selection as $\mathtt{deg}^+(n) \to \infty$. Also, $d_n = 0,\ n \in \mathcal{V}\ \&\ \mathtt{deg}^+(n) = 1$.

This metric does not make any qualitative statement about learning behaviour, because it cannot be ruled out that uniformly random behaviour is actually the best policy. Instead it gives an indication of how much the action selection probabilities are fluctuating due to adaptations to congestion for example. Moreover, it provides an indication of how distinctive the learnt policies are.

Besides this simple metric, the general queueing metrics introduced in Section 3.1 can be investigated as well. Given snapshots of the queueing network in a running discrete event simulation, steady-state queueing metrics can be calculated assuming a fixed routing probabilities matrix, $\mathbf{Q}$.

The previous chapter utilises response surface methodology to calibrate the reinforcement learning methods such that the total average event processing time is minimised. It was shown that significant improvements over a given design domain can be attained and that the respective learning method may not be universally better at minimising this objective than uniformly random decision policies. In the following paragraphs and sections the dynamic behaviour of the scenarios introduced in the previous chapter are studied in more detail. To do this, the optimal settings for the learning methods are used. Note, that the simulations based on the best calibration exhibit a stochastic character that is not influenced by sampling neighbouring locations. In other words, if the dispersion matrix, $\mathbf{\Psi}$), is defined then stochastic Kriging relaxes the assumption that the simulation output exactly equals the response surface as in traditional Kriging. As a consequence the stochastic computer simulation with the optimal parameters may deliver results that do not exactly resemble the equivalent prediction on the surface.

Figure 7.1 presents the convergence diagnostics of the queueing metrics utilisation, response time, and delay for the BBV model under high queueing stress (i.e., with $\delta_V = 1.2, \delta_E = 1.2$) including the 95% confidence intervals. One can see that the system utilisation rate for the task network is roughly the same for both learning policies with and without collaborative function approximations. The mean delay and mean response, however, are significantly reduced for the ϵ-greedy policy with collaborative function approximation and the WPL learner.

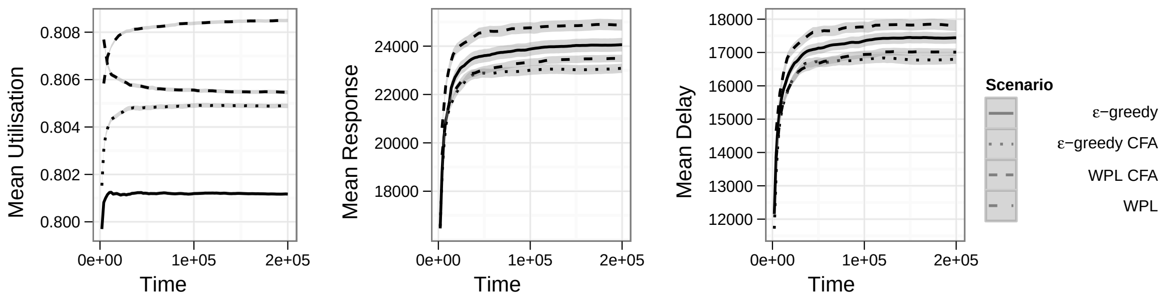


Figure 7.1: Convergence Diagnostics for the BBV model with $\delta_V = 1.2, \delta_E = 1.2$

Similarly, Figure 7.2 shows the diagnostic plots for the BBV model with a reduced queueing stress level (i.e., $\delta_V = 1.8, \delta_E = 1.8$). As before, the ϵ-greedy policy coupled with the collaborative function approximation exhibits slightly improved queueing performance.

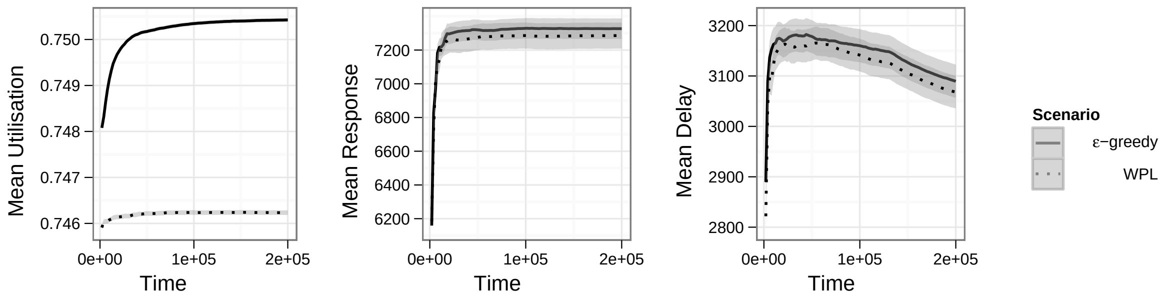


Figure 7.2: Convergence Diagnostics for the BBV model with $\delta_V = 1.8, \delta_E = 1.8$

The following diagnostic plots, Figure 7.3 and Figure 7.4, show the diagnostic plots for the ER model with both low and high queueing stress levels.

Figure 7.5 illustrates how much in the metric space of the probability simplex the learned behaviour deviates from a uniformly random policy. While ϵ-greedy learning methods exhibit a fixed distance from uniformly random policies (i.e., $\mathbf{Q}_{(i)} = [1/\mathrm{deg}(i)^+]_{\forall j \exists (i,j)}$), the weighted policy learner naturally is lower, because WPL updates the policy space directly and ensures that an action is selected with at least ϵ.

For the BBV model with $\delta_V = 1.2, \delta_E = 1.2$ the weighted policy learner with collaborative function approximation displays a more erratic behaviour compared to WPL alone. While WPL CFA covers a greater distance from uniformly random decision policies, it does not yield an improvement of the queueing performance compared to WPL (see Figure 7.1).

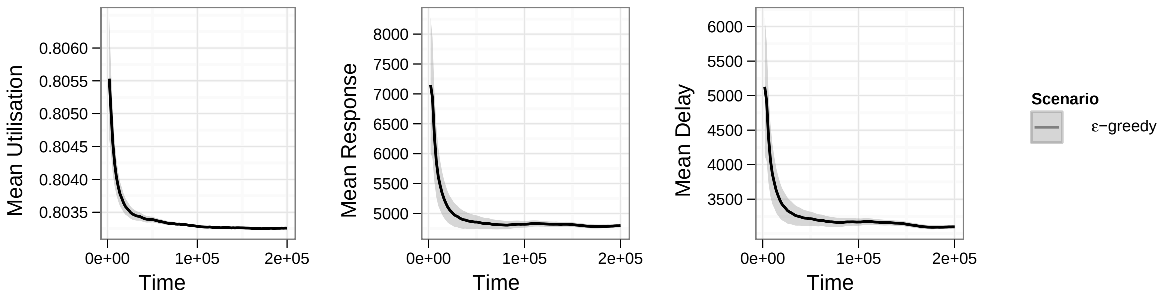


Figure 7.3: Convergence Diagnostics for the ER model with $\delta_V = 1.2, \delta_E = 1.2$

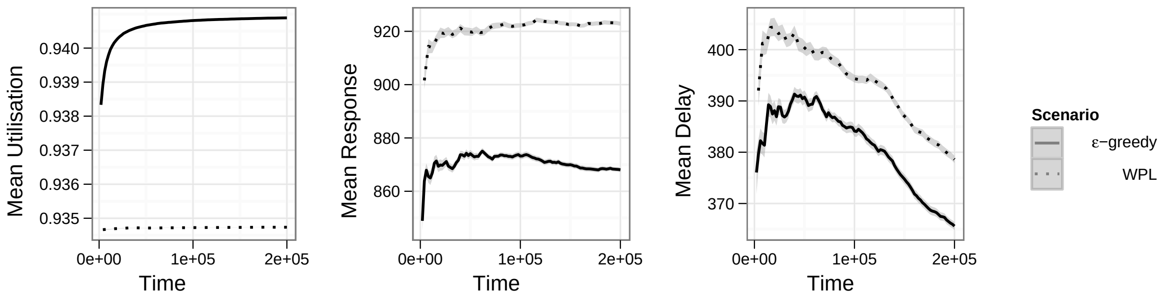


Figure 7.4: Convergence Diagnostics for the ER model with $\delta_V = 1.8, \delta_E = 1.8$

Figure 7.5 also shows that high queueing load results in learning policies that are more distinctive. A reason for this behaviour originates from non-linear queueing performance for utilisation rates above $\approx 80\%$ (see Figure 3.1). In this regime, a difference in performance between available servers downstream results in a quick adaptation towards the better performing server, which can be measured using the metric given in equation (7.1).

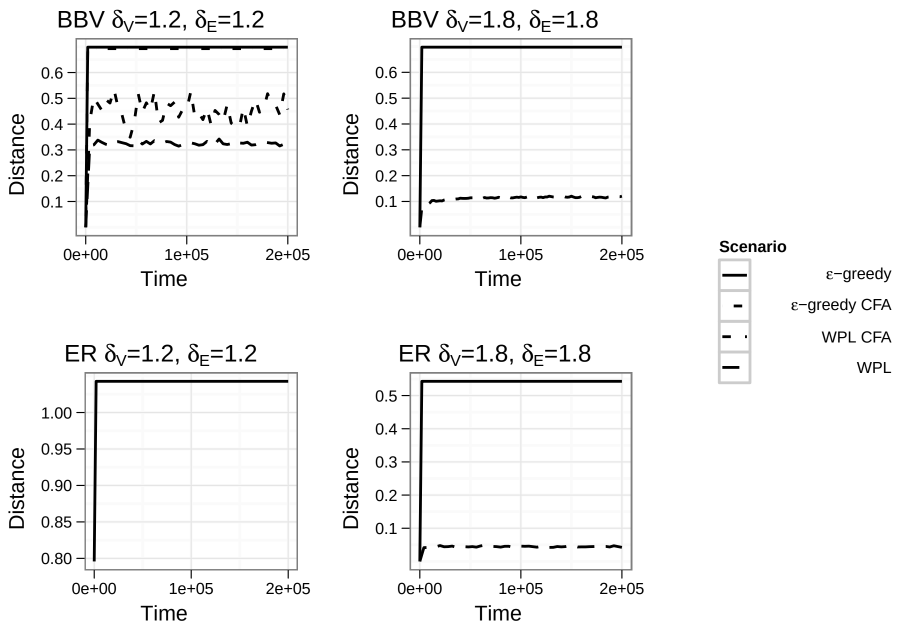


Figure 7.5: Distance from Uniformly Random Decision Policy

7.2 Reward-based Dynamics

A second way of evaluating the non-linear dynamics of multi-agent reinforcement learning is to study the evolution of the Q-values. In the limit, the Q-values give the expected discounted reward for a given action and policy. Because Q-values give expectations, their values are more stable than for example their integral part, the rewards.

Ahmadabadi and Asadpour [4] proposed expert metrics based on the rewards received which they incorporated directly into their Q-value function estimation. That way, an agent can take advantage of the expertness of its neighbouring agents. They use a weighted strategy sharing algorithm that distinguishes two modes of operation. First the cooperative agents learn individually to build up some expertness. After a given time step, the mode switches into a cooperative mode upon which the calculation of the expert metrics stops. In this mode the Q-table of an agent is calculated as the weighted average of its neighbours Q-table based on the respective expertness levels.

The expertness metrics used in this thesis are defined with respect to the expected discounted reward.

1. An algebraic sum of the reinforcement signals

$$e_i^N = \sum_{t=0}^{T} r_t(i). \tag{7.2}$$

2. A sum of the absolute value of the reinforcement signals

$$e_i^A = \sum_{t=0}^{T} |r_t(i)|. \tag{7.3}$$

3. A sum of the positive reinforcement signals

$$e_i^+ = \sum_{t=0}^{T} r_t^+(i). \tag{7.4}$$

$$r_t^+(i) = \begin{cases} 0 & \text{if } r_t(i) \leqslant Q_i(s') \\ r_t(i) - Q_i(s') & \text{otherwise.} \end{cases}$$

4. A sum of the negative reinforcement signals

$$e_i^- = \sum_{t=0}^{T} |r_t^-(i)|. \tag{7.5}$$

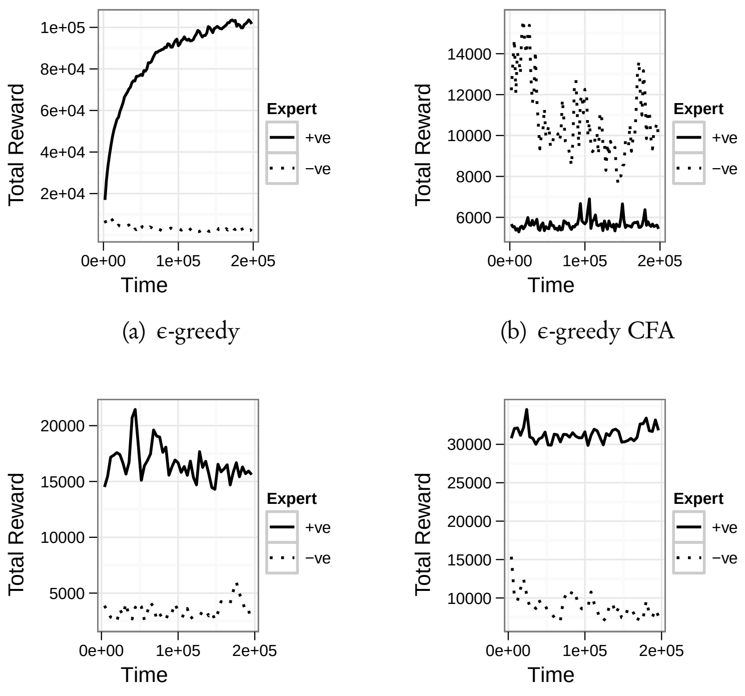
$$r_t^-(i) = \begin{cases} 0 & \text{if } r_t(i) \geqslant Q_i(s') \\ Q_i(s') - r_t(i) & \text{otherwise.} \end{cases}$$

Not only can these expert metrics be incorporated into the state signal perceived by each agent (see explicit coordination in Section 5.2.1.2), the evolution of these calculated independently within given time intervals can give indications about the learning behaviour of the agents.

Figure 7.6 and Figure 7.7 show the positive (7.4) and negative (7.5) expert metrics over the course of a single simulation run of the BBV models. In all except one cases, the positive value is significantly higher than the negative one. This implies that as learning proceeds through time negative rewards are minimised while positive rewards are maximised. Interestingly, the spread between positive and negative expert measures of the ϵ-greedy policy is the largest and the same scenario with collaborative function approximation employed yields a reverse relationship. Consequently, the ϵ-greedy policy appears to be the best policy to exploit acquired information. A similar result can be obtained for reduced queueing stress levels in Figure 7.7.

For WPL the spread is widest with collaborative function approximation. One reason why WPL does not achieve similar results to the ϵ-greedy policy can be attributed to the two distinct phases of the weighted policy learner. For as long as the gradient of the policy does not change direction learning slows down. If a change in direction takes place the weighted policy learner accelerates learning. If the rewards received over time cause constant changes in the policy gradient, then WPL will not be able to escape the accelerated learning phase and consequently it will be difficult to gear the policy into a more exploitative role.

While these reward measures give an indication on the learning performance, they do not provide a statement about optimality [88]. Instead, the concept of regret considers how well an agent would have done had it selected the optimal action, assuming the other agents in the environment choose the same actions again. Bowling [23] considered total regret as an evaluation criterion for the GIGA-WoLF learning algorithm for multi-agent systems. Equation (7.6) gives the formula for calculating total regret, which is defined as the difference between the maximum total reward of a pure strategy given the past history of interactions with the environment and the agent's total



(a) ϵ-greedy (b) ϵ-greedy CFA

(c) WPL (d) WPL CFA

Figure 7.6: Expert Metric of BBV with $(\delta_v, \delta_E) = (1.2, 1.2)$

actual reward.

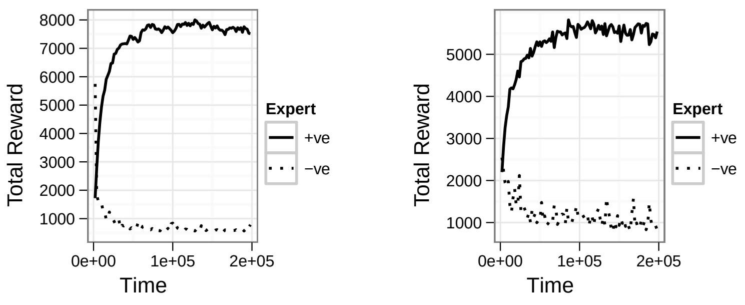
$$\mathcal{R} = \max_{a_i \in \mathcal{A}_i} \sum_{t=1}^{T} (r_i^{(t)}(\pi_{-i}, a_i) - r_i^{(t)}(\pi_{-i}, \pi_i)). \tag{7.6}$$

a_i represents the action for agent i that maximises total reward assuming the opponents strategy π_{-i} would not have been affected and $r_i^{(t)}(\pi_{-i}, a_i)$ is the received reward from following the pure strategy. The average regret is taken as the asymptotic average of the total regret, $\lim_{T\to\infty} \mathcal{R}/T$. Following this definition, an algorithm has no-regret if and only if the average regret is $\lim_{T\to\infty} \mathcal{R}/T \leqslant 0$. Conversely, a positive regret means the agent would have been better off following the pure strategy. This notion of regret is relatively weak, because it only takes into account a pure strategy. In contrast, incentive to deviate is defined as the difference between the actual reward and the best response at any given time step

$$br(\pi_i) = \arg\max_{\pi_i} \mathbb{E}[r_i^{(t)}(\vec{\pi}_{-i}, \pi_i)] \tag{7.7}$$

$$\begin{aligned} ID_{\pi_i^{(t)}} =& \max\{(0, r_i^{(t)}(\vec{a}_{-i}^{(t)}, br_i(\pi_i)) \\ & - r_i^{(t)}(\vec{a}_{-i}^{(t)}, a_i^{(t)}))\}, \end{aligned} \tag{7.8}$$

where π_i and $a_i^{(t)}$ represents the policy of and the action played by agent i respectively that led to the reward $r_i^{(t)}(\cdot,\cdot)$. Conversely, $\vec{\pi}_{-i}$ and $\vec{a}_{-i}^{(t)}$ are the policies of and actions played by the other agents involved in this task. The higher $ID_{\pi_i^{(t)}}$ for an agent i at time t, the higher the incentive to



(a) ϵ-greedy (b) WPL

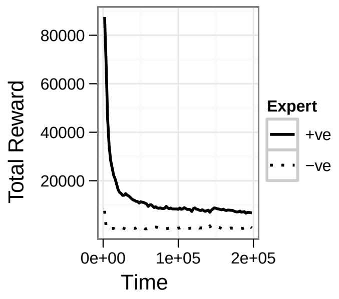
Figure 7.7: Expert Metric of BBV with $(\delta_\nu, \delta_E) = (1.8, 1.8)$

deviate from the current strategy.

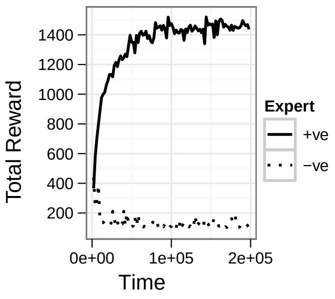
Figure 7.9 provides an overview of the regret measures of all scenarios. These reflect the results obtained from the reward metrics in that the ϵ-greedy policy achieves the lowest regret value. The two scenarios with collaborative function approximation are the worst cases in terms of regret.

The plots of the incentive to deviate metric are presented in Figure 7.10. For the high load BBV scenario, the ϵ-greedy and the WPL method have the lowest incentive to deviate. Analogously to the regret measure, the two scenarios with collaborative function approximation show the largest incentives to deviate from their current action selection policies. The WPL method over a random task topology (ER) shows significant fluctuations, while the ϵ-greedy strategy is more stable.

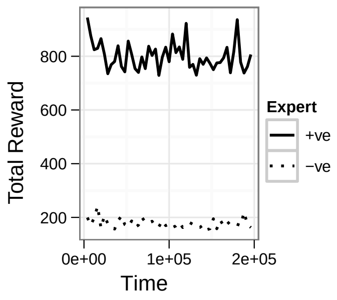
The next section presents a network view on the queueing performance and shows how only few nodes absorb the delay emanating from the whole network.



(a) ϵ-greedy with $(\delta_v, \delta_E) = (1.2, 1.2)$



(b) ϵ-greedy with $(\delta_v, \delta_E) = (1.8, 1.8)$



(c) WPL with $(\delta_v, \delta_E) = (1.8, 1.8)$

Figure 7.8: Expert Metric of ER

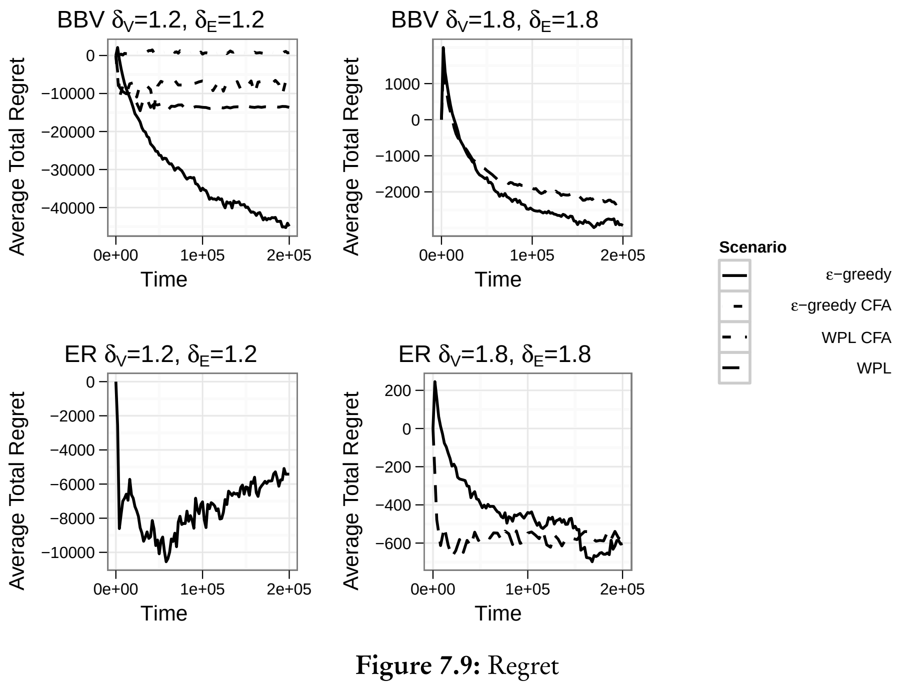


Figure 7.9: Regret

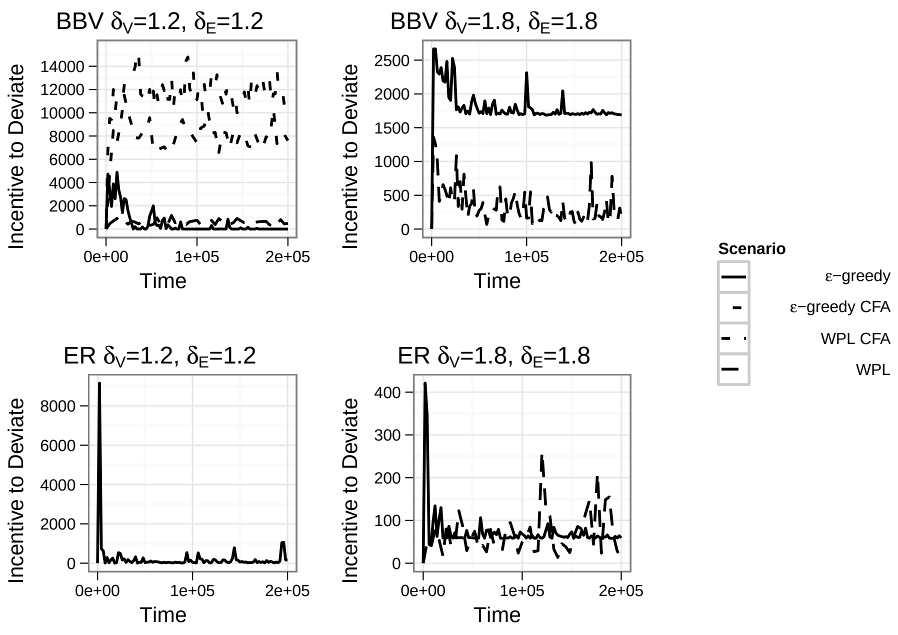


Figure 7.10: Incentive to Deviate

7.3 Influence Dynamics

Glattfelder and Battiston [63] introduced a framework to expose control structures within complex networks of corporations that are identified over their shareholding relationship. The interesting aspect of their framework is their generality, which is exploited in this section to adapt their model to the analysis of queueing networks. The strength of the model is that non-topological features can readily be integrated into graph-theoretical structures to provide a richer analytical framework that goes beyond purely topological analyses with or without weights. The non-topological state variables are sometimes referred to as fitness and are usually assigned to nodes in the network. Further, the fitness level of nodes contribute to shaping the topology of the network as a whole. For example, in the context of multi-agent queueing networks, a regret measure can be calculated for each node in the network. The objective of the individual agents in the network is to minimise regret (or equivalently maximise the reward in the long run), which is achieved by learning a policy that assigns a probability to all available actions. The dynamics of learning and their impact on the probability of selecting an action have implications to all agents downstream. The framework by Glattfelder and Battiston [63] provides a compelling way of quantifying the influence or control that agents exercise on each other.

7.3.1 Algebraic Graph Structures for Control Analysis

Section 3.2.1 presented standard network analysis techniques that focus on weighted or unweighted and directed or undirected measures of assortativity (degree-degree correlation) and clustering coefficient. These measures only account for the immediate neighbourhood of a particular vertex in the graph and thus does not take all paths of all lengths into consideration. However, control structures have far reaching impacts that go beyond the immediate neighbourhood. In fact, all nodes reachable from any particular node in a queueing network exhibit indirect control relationships. In this section the algebraic graph structures are derived to suit the analytical framework of the analysis of control in queueing networks. Since establishing the control analytical framework is generic for all queueing networks the terminology follows that of queueing theory accordingly. Without loss of generality, this framework readily applies to the analysis of decentralised control of the distributed task assignment problem.

According to the definition of queueing networks (Definition 3.1) the graph-theoretical context is embedded in the matrix specifying the routing probabilities, $\mathbf{Q}$. This matrix recovers the directionality of the links between nodes (i.e., the matrix is asymmetric) and a probabilistic interpretation of the weights (i.e., $\sum_i \mathbf{Q}_{(i)} = 1$). The first step is to define a weight matrix, $\boldsymbol{W}$, that integrate the number of events routed over the arcs in the network. Equations (3.16) and (3.17) define the actual and potential weight matrices respectively. The actual weight matrix is based on actual traffic passing through nodes in the steady state, because it relies on the aggregate arrival rates for each node, which is calculated by solving the traffic equation (3.2). The potential weight matrix, on the other hand, is based on the maximum number of events that can pass through a node. This is given by the service rate of the respective node. A node cannot process more events than is specified by this service rate. The matrix is then normalised according to

$$\bar{W}_{ij} = W_{ij} / \sum_k W_{ik}. \tag{7.9}$$

Normalising the weight matrix such that the columns sum up to 1 corresponds to the notion of first-order ownership in [63], where all nodes linking to a target node essentially own the target node.

A quantity called the concentration index that captures the relative importance of incoming arcs can be defined as follows:

$$z_j = \frac{(s_j^-)^2}{\sum_{i=1}^N W_{ij}^2}, \tag{7.10}$$

where the numerator is the square of the in-strength of vertex j (equation (3.18)). When all weights of the arcs with a target vertex j carry equal values, the concentration index is $z_j = \mathrm{deg}^-(j)$. The other extreme is when one weight is significantly larger than all others. In this case, the concentration index approaches the value one. With respect to queueing networks, z_j measures the effective number of transactions routing to node j. Conversely, a quantity measuring the prominence of the outgoing arcs is the control index. First, let H_{ij} measure the importance of i to j with respect to all other vertices connecting to j be

$$H_{ij} = \frac{\bar{W}_{ij}^2}{\sum_{l=1}^N \bar{W}_{lj}^2}, \tag{7.11}$$

where $H_{ij} \in [0, 1)$. A value of $H_{ij} \approx 1$ indicates that the vertex i is the most important vertex for target vertex j. Then the control index is given as

$$h_i = \sum_{j=1}^N H_{ij}. \tag{7.12}$$

Figure 7.11 depicts a small network, where vertices $\{i_1, \dots, i_3\}$ exercise control over vertices $\{j_1, \dots, j_3\}$ according to the arcs emanating from $\{i_1, \dots, i_3\}$. Since i_1 is the only vertex connecting to vertices j_1 and j_2 the values for $H_{i_1 j_1}$ and $H_{i_1 j_2}$ are both one and consequently i_1 is the most important source vertex. Intuitively, this means that the only way j_1 and j_2 can participate in processing events is through vertex i_1. If i_1 were to send all events to j_3 this could have economic implications.

Equations (7.10) and (7.12) solely account for topological features of the queueing network. Non-topological state variables can be integrated in this model yielding portfolio value

$$p_i = \sum_{j=1}^N \bar{W}_{ij} v_j \tag{7.13}$$

and a control value

$$c_i = \sum_{j=1}^N H_{ij} v_j. \tag{7.14}$$

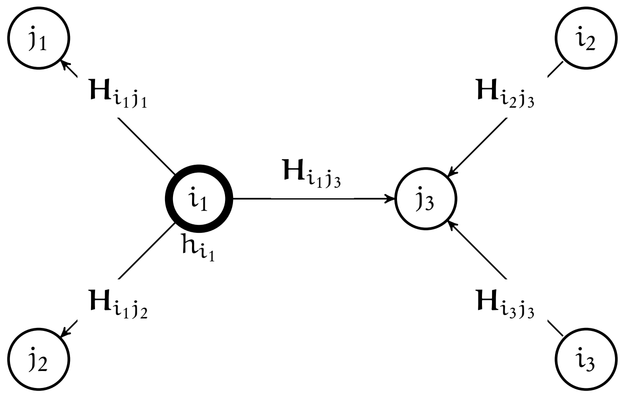


Figure 7.11: Control Index

Equation (7.14) only accounts for the immediate neighbourhood of a given node i and does not integrate the control of all paths of all lengths. Therefore, an integrated model can be defined for both ownership $\bar{W}$ and control H. Let M be the ownership or control matrix. It holds that

$$\sum_{i=1}^{N} M_{ij} \leqslant 1, \; j = 1, \ldots, N. \tag{7.15}$$

Then the integrated ownership or control model can be calculated through a recursive computation written in matrix notation as

$$\tilde{M} = M + M\tilde{M}, \tag{7.16}$$

and solving for $\tilde{M}$, the solution is given by

$$\tilde{M} = (I - M)^{-1}M. \tag{7.17}$$

Then the integrated control value for each node is calculated in two steps. First, the integrated fraction of control $\tilde{H}$ matrix is calculated using equation (7.17) and then analogue to equation (7.14) the integrated control value that integrates all paths of all lengths for node i is calculated as

$$\tilde{c}_i = \sum_{i=1}^{N} \tilde{H}_{ij} \upsilon_j. \tag{7.18}$$

7.3.2 Identifying the Backbone of Control

Following the method of generalising the backbone extraction for any weighted and directed network described in Glattfelder and Battiston [63], two conditions have to be met: (1) the non-topological value $\upsilon \succeq 0$ for all nodes in general and $\upsilon \succ 0$ for the leaf nodes in particular; (2) an arc from node i to j in a queueing network is defined as scalar real value in the weight matrix $\bar{W}_{ij} > 0$, which implies in the context of control that some value is transferred in the opposite direction of the arc. In a physical sense, this value can be interpreted and formalised as the flow Φ_i entering node i from each successor node j at time t.

$$\mathbf{\Phi}_i(t+1) = \sum_j \bar{\mathbf{W}}_{ij}\boldsymbol{\upsilon}_j + \sum_j \bar{\mathbf{W}}_{ij}\mathbf{\Phi}_i(t), \tag{7.19}$$

where $\sum_i \bar{\mathbf{W}}_{ij} = 1$ for nodes j that have a predecessor and $\sum_i \bar{\mathbf{W}}_{ij} = 0$ for the root nodes, which is guaranteed by equation (7.9). This flow value is a recursive function that adds the fraction $\bar{\mathbf{W}}_{ij}$ of the non-topological value, $\boldsymbol{\upsilon}_j$, produced at node j plus the same fraction of the inflow of j. In matrix notation, this yields

$$\mathbf{\Phi} = \bar{\mathbf{W}}(\boldsymbol{\upsilon} + \mathbf{\Phi}), \tag{7.20}$$

with the corresponding solution

$$\mathbf{\Phi} = (\mathbf{I} - \bar{\mathbf{W}})^{-1}\bar{\mathbf{W}}\boldsymbol{\upsilon}. \tag{7.21}$$

Similar to Glattfelder and Battiston [63], where the flow of control, $\mathbf{\Phi}$, in ownership networks (i.e., shareholder relationships) is transferred against the direction of equity stakes, queueing networks exhibit a flow of control in terms of delay accrued at each node which is transferred against the direction of the events passed from a node i to downstream nodes j. Intuitively this makes sense as the delay that is accrued downstream has a additive effect upstream.

The integrated control given in equation (7.18) corresponds to the definition of flow in equation (7.21) in the steady state.

Plotting (η, ϑ), where $\eta(k) = k/|\mathcal{V}|$ and $\vartheta(k) = (\sum_{i=1}^{k} \mathbf{\Phi}_i)/\sum_{j=1}^{N} \mathbf{\Phi}_j$ represent the first k ranked nodes by their $\mathbf{\Phi}_i$ value in descending order, yields a Lorentz-like curve. This curve is well-known in economic settings to provide a graphical representation of for example income distribution, where the x-axis ranks the poorest x% households relative to a percentage value of income on the y-axis. Here, this curve represents the nodes in the queueing network with respect to their importance as measured by the integrated control value $\tilde{\mathbf{c}}_i$.

The backbone of the network can be extracted by identifying the top ranked nodes, $\hat{\eta}$, that are responsible for a specified fraction, $\hat{\vartheta}$ (e.g. $\hat{\vartheta} = 80\%$), of the total flow in the network. Given the nodes, $\mathcal{V}_\mathcal{B}$, that cumulatively control $\hat{\vartheta}$ of the network, a sub-network can be defined that comprises these nodes and their original respective arcs between them.

Definition 7.1 (Backbone graph). *A* ***backbone*** *of control is a tuple $\mathcal{G}_\mathcal{B} = \langle \mathcal{V}_\mathcal{B}, \mathcal{A}_\mathcal{B} \rangle$, where $\mathcal{V}_\mathcal{B}$ is a finite set of vertices $\mathcal{V}_\mathcal{B} \subset \mathcal{V}$ and $\mathcal{A}_\mathcal{B}$ is a finite set of arcs $\mathcal{A}_\mathcal{B} \subset \mathcal{A}$. $\mathcal{V}_\mathcal{B}$ cumulatively control $\hat{\vartheta}$ of the network.*

Since there is no bipartite structure given by queueing networks as opposed to the shareholder network of Glattfelder and Battiston [63], the proxy to characterise the local patterns of control is given as

$$\bar{s} = \frac{\sum_{j=1}^{|\mathcal{V}_\mathcal{B}|} z_j}{|\mathcal{V}_\mathcal{B}|}. \tag{7.22}$$

Recall, the value $\hat{\eta}$ reflects the percentage of nodes that control the network corresponding to the threshold defined in $\hat{\vartheta}$. In order to facilitate comparisons of different networks, $\hat{\eta}$ is normalised as

$$\eta' = \frac{\hat{\eta}}{n_{100}}, \tag{7.23}$$

where n_{100} is the minimum number of nodes controlling 100% of the queueing network. η' captures the concentration of control with small values being indicative of few nodes controlling the global queuing network.

For queueing networks this has some profound impact. The top ranked nodes identified by $\hat{\eta}$ accrue most of the delay (if delay is the non-topological scalar to be considered). Therefore, any decisions by agents downstream with detrimental affects will be noticed by the agents upstream. If there are payments associated with routing transactions downstream, deteriorating queueing performances can be incorporated are reflected in negative rewards and consequently will lead to avoiding this path. The control manifests itself in decisions that affect potentially a number of nodes downstream.

7.3.3 Instantaneous Control Structures for Adaptive Queueing Systems

Putting this framework to work on adaptive queueing system is novel in that a network perspective can reveal structural properties that can explain the queueing performance in new ways. The local pattern of control, $\bar{s}$, is the average effective number of transactions in the backbone network. The higher this value, the more dispersed control is in the backbone. In the context of queueing network this implies that lower values for $\bar{s}$ indicate fewer nodes accrue the non-topological scalar, e.g., the delay in queues. Recall, that this metric integrates delay from all direct and indirect paths of all lengths downstream.

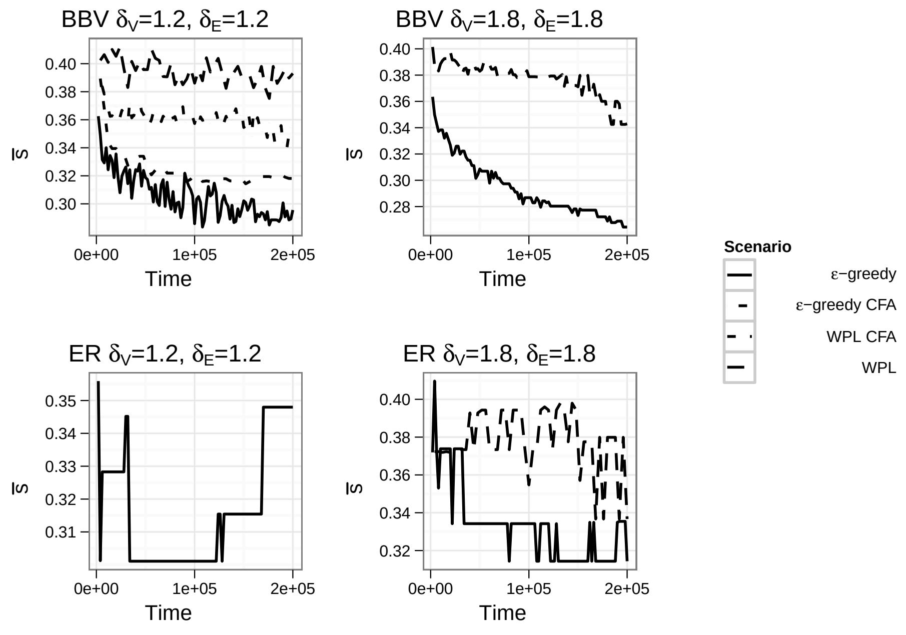


Figure 7.12: Local Control

The local pattern of control presented in Figure 7.12 shows that the WPL policy is able to disperse control better than the ϵ-greedy method. Intuitively, this makes sense, because the ϵ-greedy policy mostly selects greedy actions, which results in a concentration index (7.10) approaching 1 per node and consequently the local control index (7.22) attains a low value. In the high queueing stress scenario the ϵ-greedy outperformed WPL with respect to mean delay. Considering that the ϵ-greedy policy concentrates the overall control of the network in fewer nodes, the network structure is less robust. Therefore, the performance of the learning system needs to be put into context of the network structure to obtain a more detailed picture of the learning behaviour and its impact in networked applications.

In the context of queueing networks, $\hat{\eta}$ identifies the top ranked nodes that are responsible for accruing a defined fraction $\hat{\vartheta}$ (here 80%) of the total delays in the network. Consequently, a small value of η' signifies that the delay concentrates in a few nodes, while a large value for η' indicates that the delay is more spread between the nodes in the network. Figure 7.13 displays the evolution of η' over time covering the different scenarios examined in this thesis. It shows that with the WPL policy employed the delays in the queues are concentrated in fewer nodes compared to the ϵ-greedy method under high load in the BBV network. In contrast, low queueing load scenarios do not show any difference in the global concentration levels. Correlating the local control ($\bar{s}$) and the global concentration (η') indices, the WPL policy is able to disperse the control within the backbone more than the ϵ-greedy method. However, the backbone of the network with the WPL policy is concentrated in fewer nodes. Interestingly, a lower load profile for the BBV network yields a narrower backbone, while the level of dispersion within this backbone is similar. This shows that as the load increases the delay affects an increasing number of nodes in the network, while the level of dispersion between those affected nodes is nearly the same.

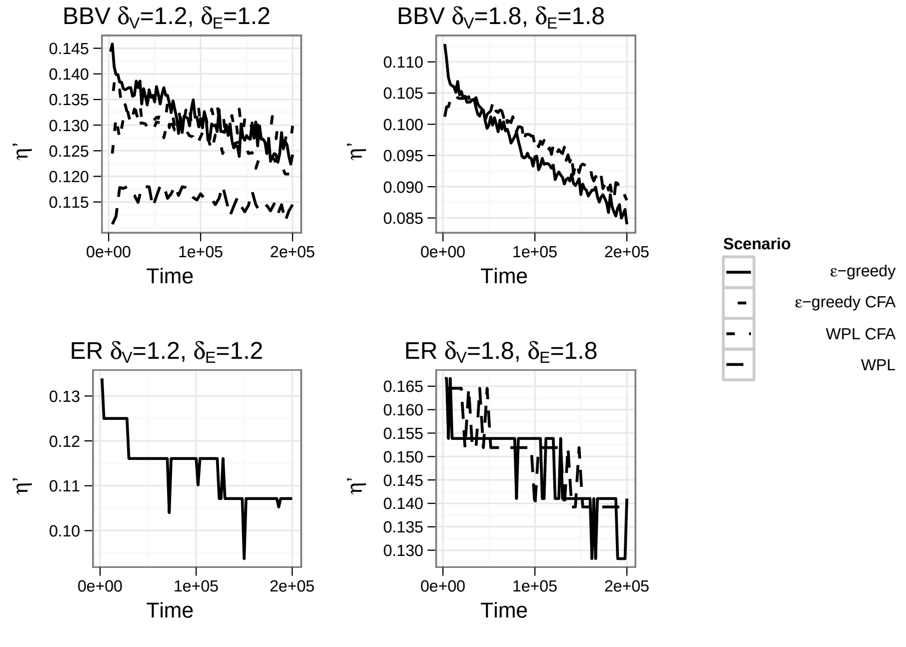


Figure 7.13: Global Concentration

These results are a first attempt of accounting for network effects in analysing multi-agent reinforcement learning algorithms.

7.4 Summary

This chapter set out an agenda to analyse the non-linear dynamics of intelligent agent learning behaviour. This agenda was delivered in three main parts that all illuminate different aspects of the learning dynamics. First, metrics based on queueing performance were covered. These include simple diagnostic plots that illustrate the equilibration of utilisation rate, response time, and delay in queue. Additionally, a metric was introduced that measures the distance from uniformly random decision policies. While it does not provide a qualitative statement about the “goodness” of the learnt behaviour, it indicates, however, the stochastic difference between different scenarios. The distance metric is also capable of identifying fluctuating learning performance. The second approach provides a perspective on received rewards over time. These measures are a first step of providing some qualitative feedback on the learning performance. However, the expert metrics do not give a notion of optimality. This is achieved with regret and incentive to deviate. Regret considers how an agent would have done had it selected the optimal action while all the other agents choose the same action again. A positive regret value indicates that an agent would have been better off following the pure strategy. Complementary to regret is the incentive to deviate, which can attain positive and negative values. If positive, then the agent would be better off selecting the best response action. Finally, a networks perspective was introduced that evaluates control structures. This methodology assists in identifying and extracting the backbone of complex networks that comprise weighted directed links. The nodes are associated with a non-topological scalar quantity, which has a physical connotation of producing mass and transferring it to the nodes upstream. In the context of queueing systems, the mass can be the delay encountered in the queues. That way, the delay accrued by the nodes upstream can be calculated. Applying this methodology to queueing systems, the procedure identifies the backbone as the sub-network where most delay is experienced in the network. Analysing this backbone over time in the context of adaptive queueing networks provides a deeper understanding of the impact of dynamically changing learning policies, which is reflected in the routing probabilities.

Observation 7. *In conjunction with the metric that measures the distance from uniformly random policies, it can be observed that WPL under low load dynamics is much closer to uniformly random policies. This implies that some agents have action spaces with almost equal selection probabilities.*

Observation 7 can be attributed to the mechanism by which the policies are updated. Due to the fact that changes in the gradient direction results in accelerated learning an increase in fluctuating learning dynamics may be the cause of why WPL does not seem to exploit better than the ϵ-greedy policy.

Observation 8. *High load queueing behaviour results in more distinctive learning policies compared to low load scenarios.*

Observation 8 can be attributed to the non-linear queueing performance for utilisation rates above $\approx 80\%$ (see Figure 3.1). In this regime, a relatively small difference in utilisation of the servers downstream results in vast differences in queueing behaviours. As a result, agents adapt to this change to reflect the relative performance, which can be measured using the metric given in equation (7.1).

Observation 9. *The positive expert measure attains values much higher than the negative expert measure.*

Assumption 9.1. *Throughout the simulation, the expert metrics were collected according to equations* (7.4) *and* (7.5)*, which accumulate the positive and negative expert values over time. In order to calculate to calculate averages the values were calculated for* 100 *intervals.*

Observation 10. *The* ϵ*-greedy policy attains a much wider spread between the average positive and negative expert metrics, than the WPL policy.*

Observation 11. *In all but one scenario are the positive expert values higher than the negative expert values.*

Observation 12. *All scenarios illustrate that positive expert values are maximised and negative expert values minimised.*

These observations are expected, because the learning objective is to minimise the response time. This implies that, as intelligent agents adapt to an environment they gain knowledge about which actions are favourable over others. Selecting favourable actions will ultimately lead to positive reinforcement signals.

Observation 13. *The* SARSA(0) *reinforcement learning method with an* ϵ*-greedy policy exhibits the lowest regret for the BBV and the ER models.*

Observation 14. *The* SARSA(0) *reinforcement learning method with an* ϵ*-greedy policy exhibits the lowest incentive to deviate from current action selection policy for the BBV and the ER models in most scenarios.*

Observation 15. *The* SARSA(0) *reinforcement learning method with the weighted policy learner is able to disperse the control within the backbone and shows a lower concentration of high delay values compared to the* ϵ*-greedy policy.*

Observation 16. *Given the BBV network model, the global concentration level of high delay values in the queues of the network reduces as the queueing load decreases.*

Adaptive methods that are able to disperse control may also be more likely to perform well, because delay in the queues have a smaller impact upstream. Interestingly, if the queueing system is under stress, i.e., experiences a high load, more distinct policies are learnt which may impact the robustness properties of certain networks. Evidently, a failure of a node in a system under stress which lacks strong robustness guarantees potentially leads to cascading failures. Future work will investigate, how learning agents are able to cope with failures and to what degree those failures affect the system as a whole.

These observations indicate that WPL outperforms the ϵ-policy in most scenarios. However, a more thorough analysis is required to evaluate the uncertainty associated with those results. Since, the focus of this chapter was to illustrate learning behaviour without smoothing the results over many replications this is left for future work.

— 42.

Douglas Adams (The hitchhiker's guide to the galaxy)

8 Conclusions

The research conducted for this thesis set out to investigate the modelling and analysis of cooperative adaptive queueing networks and their learning behaviour. It emphasises on the importance of including topological features of the underlying interaction pattern of agents among each other. In the context of distributed task assignment in particular, and queueing networks in general, this required the construction of different network topologies. The decentralised control mechanism was based on SARSA(0) temporal-difference reinforcement learning using neural network function approximation to estimate the state-action value function. This way, a generic layer for state representation was adopted to facilitate an efficient study on which state variables provide the best information to learn the state-action mapping. The cooperative behaviour was designed through a reward function that measures the negative response value for a given task completion. This encourages the multi-agent system to act in concert to achieve a global goal, which is the minimisation of the total time of the task in the system. The learning methods were evaluated to find the best parametrisation to achieve globally optimal behaviours. Additionally, behavioural analyses were conducted to demonstrate the qualitative differences in the learning behaviours. The efficient analysis of stochastic computer simulation formed one important objective of this thesis. Kriging metamodelling and the canonical analysis of response surfaces was chosen as an appropriate means to synthesise and generalise statistical findings so that they can inform a system designer for distributed task assignment problems.

In this chapter, the contents of this thesis are summarised and an overview of the contributions with respect to the state of the art are presented. To conclude, future work and research ideas are discussed based on the work carried out up to date.

8.1 Summary

As simulation studies are becoming more prevalent in computer science, systematically controlling the experiment or a thorough analysis of the relationship of the explanatory variables and the response variable is often neglected. In particular, the nature of this thesis has a strong empirical character which necessitates a statistical framework to support a qualitative comparison of the decentralised control algorithms. Chapter 2 presented response surface methodology (RSM) comprising of a collection of well-known mathematical and statistical techniques to design, analyse, and optimise simulation experiments which are drawn from published literature. This thesis evaluates stochastic computer experiments and accordingly the focus was placed on statistical tools that treat simulations as a black box and that allow the construction of response surfaces that integrate the stochastic nature of the simulation output. The construction of a response surface is achieved using stochastic Kriging, which is an extension to regression that accounts for spatial correlation among the data points. A fully Bayesian interpretation of stochastic Kriging was given to capture the various sources of uncertainty including the variance of the computer simulation output, the uncertainty about predicting unobserved data locations, and the uncertainty resulting from the Markov Chain Monte Carlo inference of the Kriging model parameters. Canonical analysis was presented to aid in the mathematical study of the resulting response surface and to provide a means of analysing computer simulations in higher dimensional design domains.

Chapter 3 introduced network evolution models that adhere to queueing-theoretic semantics known from operations research. The purpose of network evolution models in the context of this thesis is their integration into larger simulation studies that investigate adaptive algorithms to cope with defined traffic patterns. With the inclusion of network evolution models different network sizes and different topological features can be represented. Two network evolution models known from the complex networks domain were adapted to comply with stable Jackson networks. Their respective topologies resemble that of random networks and scale-free networks. Random networks exhibit a lower clustering coefficient, a higher average path length between reachable nodes, and a higher diameter compared to scale-free networks. The ability to cope with task requests entering the system, two parameters govern the assignment of the capacity to nodes. This is a simple model that allows the investigation of traffic patterns solely depending on those two parameters while keeping the external arrival rate fixed.

Chapter 4 presented Markov decision processes and their elementary solution concepts based on solving value functions. In particular the value-iteration and policy-iteration algorithms known from dynamic programming were presented to solve MDPs. The disadvantage of dynamic programming is that those methods require a full specification of the reward and transition functions, which makes them prohibitively expensive in most practical scenarios. Monte Carlo methods were then introduced which do not rely on prior knowledge of the environment, but instead operate in a trial-and-error fashion to sample the environment in order to provide estimates. The main difference between these methods were pointed out and how temporal difference reinforcement learning methods combine the best of both dynamic programming and Monte Carlo methods. The first part of this chapter focused on the theoretical underpinnings of the respective methods. In the second part, which focused on multi-agent reinforcement learning, the implications of the convergence guarantees for single-agent stationary MDPs on multi-agent systems were set forth. Recently, research synthesised the game-theoretic framework with multi-agent systems to address some of the issues related to finding optimal joint actions. The game-theoretic framework provides a natural extension to multi-agent reinforcement learning, because it models the behaviour of one agent in the face of others in the environment. Recent literature on cooperative and competitive reinforcement learning was reviewed.

Sequential distributed task assignment problems in this thesis were modelled with a utilitarian stance to avoid the "Tragedy of the Commons", which can often be observed when selfish agents operate in an environment. Decentralised task assignment was formalised using the general framework of decentralised MDPs. The cooperative nature of the multi-agent system is designed around the reward structure. More specifically, the reward is made up of the negative value of the response time of (sub-) task completion.

Given a specification of the cooperative distributed task assignment problem SARSA(0) reinforcement learning agents endowed with the undirected ϵ-greedy policy and a directed weighted policy learner were investigated. In order to simulate different environmental settings the task network was instantiated over two different task topologies, which were introduced in Chapter 3. Additionally, since the performance modelling was based on the mathematical framework of queueing theory two different queueing load levels were investigated to find out how agents adapt in the face of stress. The main contribution of this Chapter 6 is the empirical analysis using the response surface methodology presented in Chapter 2. The main findings are that carefully calibrating all free parameters of a learning method outperforms uniformly random decision policies and the weighted policy learner is superior in most instances to the ϵ-greedy policy. This in itself is not a very strong statement about the quality of the learning method. However, the method of calibration and consequently understanding which learning parameters work best are interesting results. Another aspect explored in this chapter is the utility of collaboratively training the Q-value function approximator. This way, agents can efficiently utilise knowledge about a neighbour's queueing state. In the scenarios considered, the collaborative function approximators did not exhibit a discernible impact on the overall system performance.

Chapter 7 set forth an agenda for examining the dynamic learning behaviour of multi-agent systems. A new simple metric was introduced to measure the distance in probability space from uniformly random decision policies. While it does not provide a qualitative statement about the performance, it complements the reward-based expert metrics. The expert metrics accumulate positive and negative rewards respectively and it was shown that learning agents minimise negative rewards while maximising positive ones. Further, to elucidate optimality criteria of learning behaviour, the concept of regret and incentive to deviate was measured. In the single simulation runs covered in this chapter, SARSA(0) reinforcement learning equipped with the ϵ-greedy policy shows significantly less regret and an incentive to deviate from the current action selection policy than the weighted policy learner. Following this analysis, a network perspective on adaptive queueing performance was introduced. This approach identifies the backbone of a sub-network where most of the control resides. Extracting the backbone facilitates the identification of local control pattern and global concentration of control. It was shown that the ϵ-greedy policy is more able to disperse local control patterns, which has an impact on the overall system performance, because a reduced influence of control implies a more even distribution of the network flow.

8.2 Contributions and Related Works

Lipson, van den Herik et al. also conducted empirical analyses of reinforcement-learning methods to compare different learning methods [88, 161].

The contributions of this thesis can roughly be classified in four disciplines. First, the simulation environment which includes the response surface methodology. Second, the network evolution models for queueing networks. Third, the modelling of sequential distributed task assignment problems using decentralised control mechanisms. And finally, the behavioural analysis of adaptive forces on queueing networks using analytical tools derived from complex networks research.

An agent-based simulation platform implementing parallel algorithms for high-performance computing clusters was developed, where the assignment of computing nodes to certain simulation runs requires the maintenance of a list of idle and inactive computing nodes. Computing nodes may become inactive, if a particular simulation run is completed and therefore frees up its resources. Since the number of replications for a simulation run is adjusted at runtime, the inactive nodes can be reassigned to other simulation runs that have not achieved the desired confidence levels yet. That way the computing cluster remains at high efficiency rates. While this implementation is a master-slave architecture, which is quite simple to implement, the integration of response surface methodology into agent-based simulators has not received much attention.

Casting network evolution models into a queueing-theoretic framework is a novel approach that enables large-scale queueing network simulations. Two network evolution models were adapted, the Erdős-Rényi (ER) random graph and the Barrat, Barthèlemy, and Vespignani model (BBV) social graph models. Their purpose is to present entirely different topological features to the modelling of task hierarchies and to study the effects of topological structure on the learning behaviour of agent-based simulation. Also, the reverse effect of the weighted versus the unweighted assortativity metric for both network evolution models is a new insightful result, which emphasises the importance of characterising the underlying graph structure. It is shown that the dichotomous interpretation of the arcs in a large network may lead to entirely different interpretations than a more comprehensive perspective including the weights of the arcs as well.

Sequential distributed task assignment problems as presented in this thesis rely on two building blocks to go beyond illustrative academic scenarios. First, the network topology needs to be specified and secondly the autonomous agents need to be modelled such that the global goal of minimising the total time of task completion times is achieved in concert. Otherwise, selfish agents may succeed in achieving their own local goals at the expense of the whole task network. In queueing networks the response time of task completion is a natural choice for the reward function, because this function discerns an agent's own contribution on the overall effort of all agents involved in accomplishing a task. The agents are endowed with SARSA(0) reinforcement learning and a neural network function approximator to generalise over the state-action value space. To improve scalability of cooperative multi-agent systems, the state-action value function approximators were outsourced to the neighbouring agents in order to facilitate faster learning. One direct and one indirect policy were employed and the effect of network structure was studied on the learning methods. Additionally, globally optimal learning parameters were found and their sensitivities to the learning objective investigated.

The behavioural analysis was conducted using queueing, expert, and influence metrics. These metrics concentrate on certain concepts based on graph-theoretical principles and analysis their evolution through time. For the queueing metrics a snapshot of the queueing network is taken and its analytical equilibrium performance measures calculated. Additionally, a measure of total distance from random behaviour visualises the system's runtime behaviour. Fluctuations may arise which can be attributed to adapting to dynamically changing queueing conditions. The learning processes take effect in the assignment of probabilities to activating links in the queueing network. If this assignment fluctuates it may lead to cascading effects in the network and challenges the agents upstream to learn near optimal policies, because the variance of the performance of the agents downstream may be too high to discern a clear trend. The expert metrics take into account the rewards received over time and demonstrates that fluctuating behaviours are observed as well. The positive and negative expert metrics also show that positive experience is maximised while negative experience is minimised over time. The influence metrics provide a perspective on the network effects integrating measures of control that go beyond first-order neighbours. Instantaneous control structures can be derived and examined how they evolve through time. It was shown that learning behaviour can lead to an increased concentration of high delay values in the network which means that very few nodes experience high delays in their queues. This may have negative consequences if failures arise causing a few nodes to transition into an instable regime.

8.3 Future Work

The scope of the different domains covered by this thesis has led to many interesting research questions that could not be fully investigated in this thesis due to their lack of immediate relevance, or their potential to open up larger research problems. A number of areas of potential future research have been identified in the four major domains covered by this thesis.

8.3.1 Response Surface Methodology

The statistical and mathematical tools used for response surface methodology originate from published research literature. However, there are a few directions which would advance the general methodology. First, the Kriging surfaces are improved based on a mean-squared error criterion. This criterion discards knowledge about rapid changes in certain regions of the surface. Consequently, a better and more efficient means of exploring the surface could be investigated in the future, such as regression trees with criteria for bounded sub-regions of the space that capture the uncertainty about the encompassing data points. From a software engineering perspective, it would be interesting to investigate graphics processing units as a platform to conduct the Bayesian inference and validation of the Kriging model parameters.

8.3.2 Queueing Network Modelling

The queueing network evolution model presented in this thesis is relatively simple and can be taken forward in several ways. First, the arrival and service rates come from stationary probability distributions, which implies that there are no time-varying characteristics of the arrival and service processes. This, however, is rarely the case in reality. Consequently, the arrival rate could be employed as a non-homogeneous Poisson distribution to imitate the time-varying evolution of task requests. Secondly, services generally are associated with a failure rate with a stochastic repair time. Extending the current model to include service failures would introduce a more realistic interaction pattern which would further demonstrate the viability of reinforcement learning in those scenarios. Thirdly, the networks evolved for this thesis are static once established. That means that existing service nodes are not joining or leaving the network, or re-wiring their current interconnections with neighbouring services. Introducing dynamically evolving task structures would be highly beneficial and one of the most pressing research directions. The current model of the task hierarchy does not allow loops within the network to avoid infinitely cycling task requests. If the structural embedding of services are more dynamic then precautions have to be taken to account for loops. This can possibly be achieved by accounting for multiple classes of task requests. Multi-class queueing networks can also be exploited to introduce priority of requests. A higher prioritised task request would be assigned a higher-ranked position in the waiting queue compared to lower prioritised task requests. Finally, only two network evolution models were covered in this thesis. There are many more existing network evolution models that can be adapted with relative ease and would provide more comprehensive simulation settings.

8.3.3 Decentralised Control Mechanisms

One focus of this thesis was to model distributed task assignment problems using cooperative multi-agent reinforcement learning methods. SARSA(0) reinforcement learning coupled with neural network function approximation was chosen, because of its crucial ability to generalise observed data to unseen states. It is well-known that off-policy sampling and non-linear function approximators can introduce some instabilities in the learning behaviour with diverging worst-case behaviour. Neural networks appear to work better in on-policy learning algorithms, which is why Q-learning was not investigated. It would be interesting to explore other state of the art learning algorithms that provide stronger convergence guarantees in single agent as well as multi-agent systems. In particular the machine learning community concentrated on developing algorithms that avoid difficult to tune parameters altogether, such as the learning rate and the dilemma of exploration and exploitation. Mostly, these algorithms have not been explored in multi-agent settings yet and it would be interesting to proceed with this agenda.

8.3.4 Behavioural Analysis

The behavioural analysis of data captured on graphs is a fascinating research domain, because it can be applied to many fields that admit abstractions over graphs. The metrics used in this thesis are investigated over evolving graphs in order to capture the (non-linear) dynamics and their study was limited to graphical interpretations. It would be interesting to advance this analyse to include numeric fitting of time series models to be able to compare the non-linear dynamics more rigorously.

Appendices

A
MCMC Inference of the Kriging Parameters

This appendix contains the results of the MCMC inference of the Kriging model parameters corresponding to the theoretical treatment of the subject presented in Section 2.2. In particular this appendix presents the trace plots, equilibration plots, and the autocorrelation plots.

A.1 BBV DTAPs

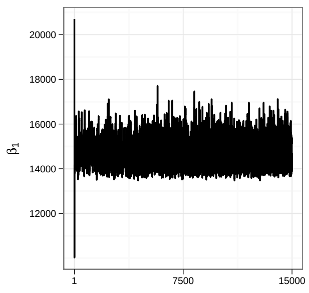
A.1.1 Uniformly Random Decision Policy

The summary of the Kriging estimation is given in Table A.1 with a MCMC chain size of 161. The Kriging model is assessed using 100 validation locations selected using a space-filling LHS design.

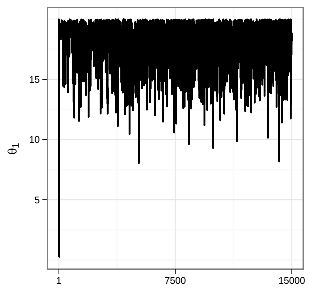
Table A.1: Kriging Summary of BBV DTAPs with a uniform random decision Policy

	$\delta_V, \delta_E \in [1.2, 1.3]$		$\delta_V, \delta_E \in [1.7, 1.8]$	
Variable	Value	95% CI half-width	Value	95% CI half-width
β_0	14548.52	101.86	7837.30	26.5
θ_{δ_V}	18.02	0.27	11.37	0.88
θ_{δ_E}	10.18	0.94	9.97	0.91
σ^2	17057226.14	308883.55	265423.15	28501.07
R^2	0.878		0.999	
RMSE	217.98		80.369	

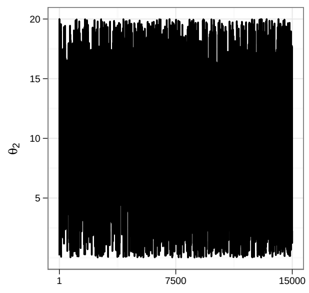
A.1.1.1 $\delta_V \in [1.2, 1.3], \delta_E \in [1.2, 1.3]$



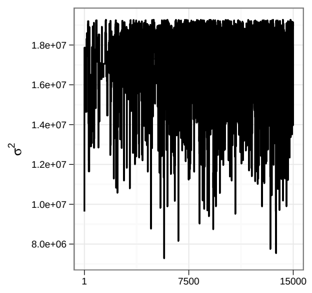
(a) Trace for β_0



(b) Trace for θ_{δ_V}

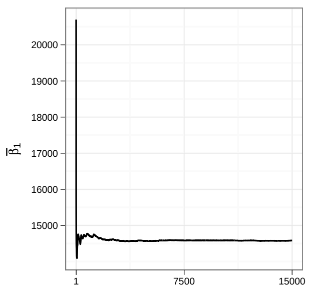


(c) Trace for θ_{δ_E}

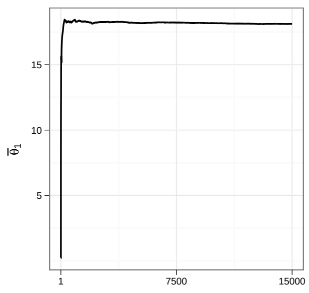


(d) Trace for σ^2

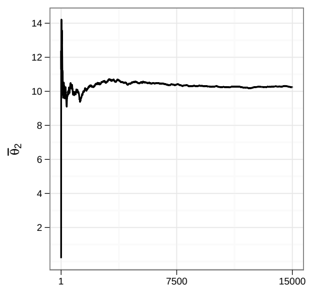
Figure A.1: MCMC Trace for BBV DTAPs with a uniformly random decision Policy with $\delta_V \in [1.2, 1.3], \delta_E \in [1.2, 1.3]$



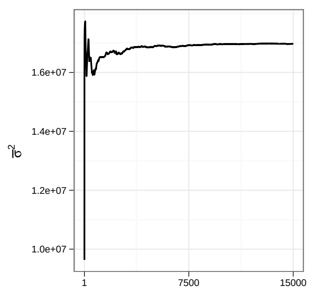
(a) Equilibration for β_0



(b) Equilibration for θ_{δ_V}

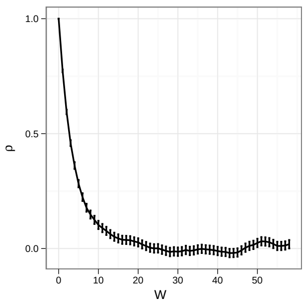


(c) Equilibration for θ_{δ_E}

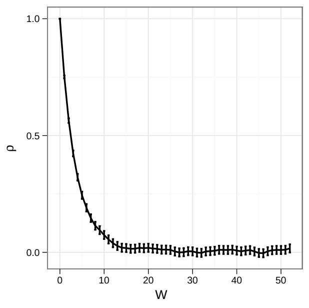


(d) Equilibration for σ^2

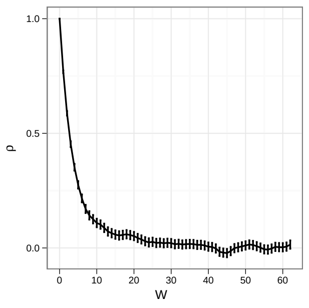
Figure A.2: MCMC Mean for BBV DTAPs with a uniformly random decision Policy with $\delta_V \in [1.2, 1.3], \delta_E \in [1.2, 1.3]$



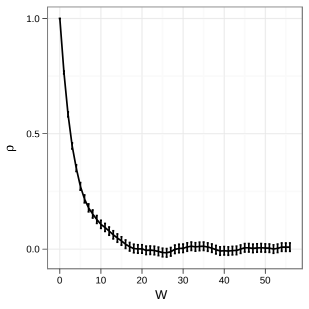
(a) Autocorrelation for β_0



(b) Autocorrelation for θ_{δ_V}

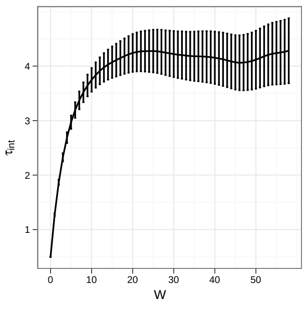


(c) Autocorrelation for θ_{δ_E}

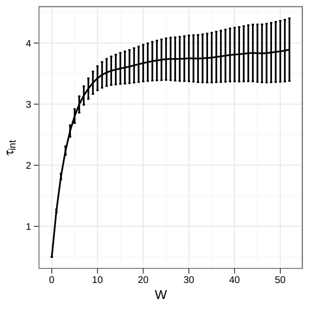


(d) Autocorrelation for σ^2

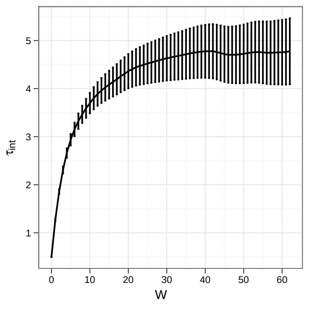
Figure A.3: MCMC Autocorrelation for BBV DTAPs with a uniformly random decision Policy with $\delta_V \in [1.2, 1.3]$, $\delta_E \in [1.2, 1.3]$



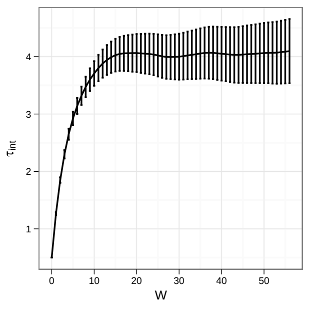
(a) $\tau_{\mathtt{int}}$ vs. W for β_0



(b) $\tau_{\mathtt{int}}$ vs. W for θ_{δ_V}



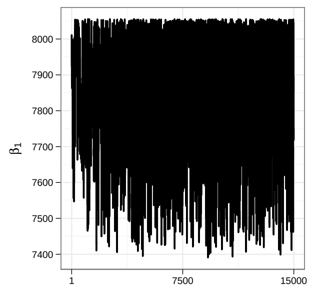
(c) $\tau_{\mathtt{int}}$ vs. W for θ_{δ_E}



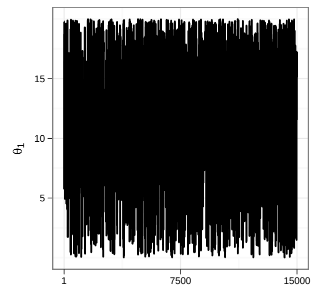
(d) $\tau_{\mathtt{int}}$ vs. W for σ^2

Figure A.4: MCMC $\tau_{\mathtt{int}}$ vs. W for BBV DTAPs with a uniformly random decision Policy with $\delta_V \in [1.2, 1.3]$, $\delta_E \in [1.2, 1.3]$

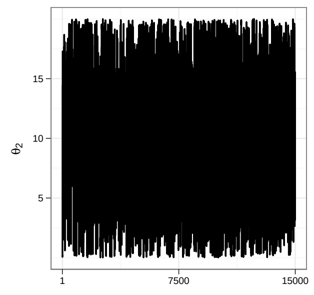
A.1.1.2 $\delta_V \in [1.7, 1.8], \delta_E \in [1.7, 1.8]$



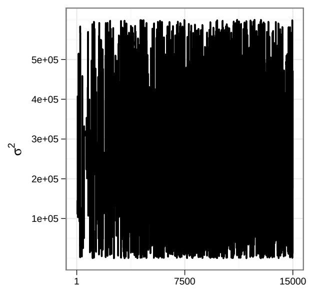
(a) Trace for β_0



(b) Trace for θ_{δ_V}

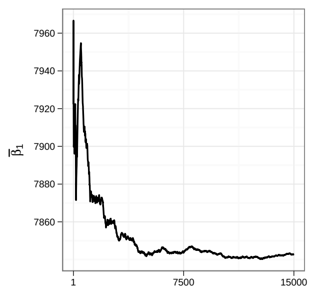


(c) Trace for θ_{δ_E}

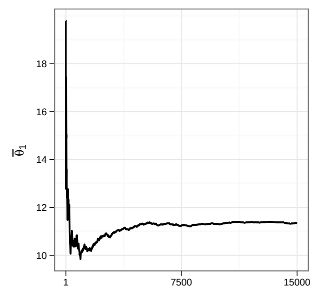


(d) Trace for σ^2

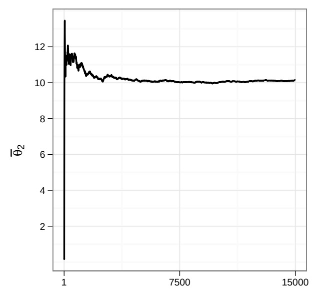
Figure A.5: MCMC Trace for BBV DTAPs with a uniformly random decision Policy with $\delta_V \in [1.7, 1.8], \delta_E \in [1.7, 1.8]$



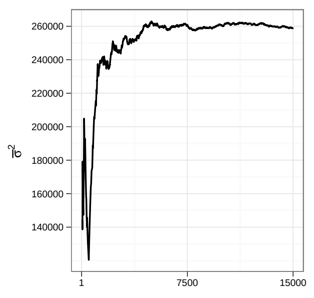
(a) Equilibration for β_0



(b) Equilibration for θ_{δ_V}

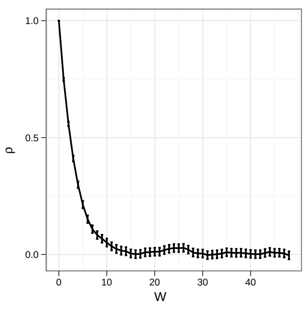


(c) Equilibration for θ_{δ_E}

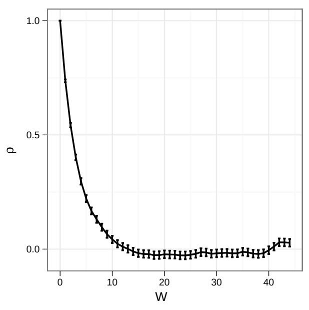


(d) Equilibration for σ^2

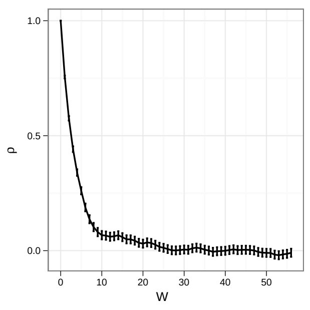
Figure A.6: MCMC Mean for BBV DTAPs with a uniformly random decision Policy with $\delta_V \in [1.7, 1.8], \delta_E \in [1.7, 1.8]$



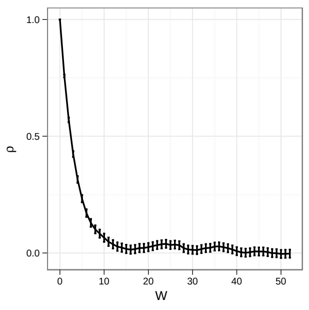
(a) Autocorrelation for β_0



(b) Autocorrelation for θ_{δ_V}

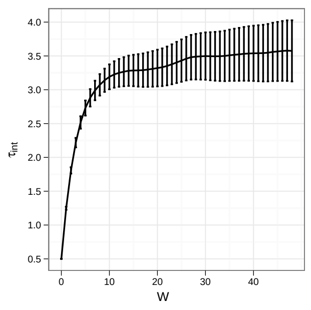


(c) Autocorrelation for θ_{δ_E}

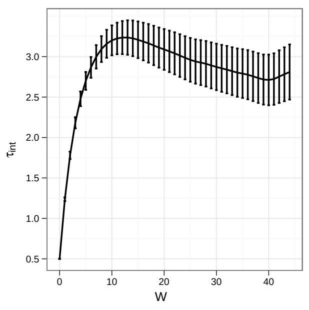


(d) Autocorrelation for σ^2

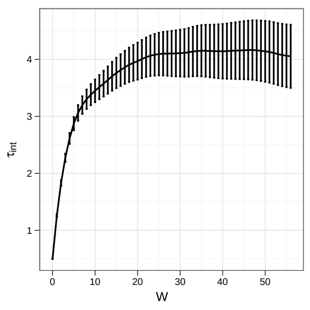
Figure A.7: MCMC Autocorrelation for BBV DTAPs with a uniformly random decision Policy with $\delta_V \in [1.7, 1.8], \delta_E \in [1.7, 1.8]$



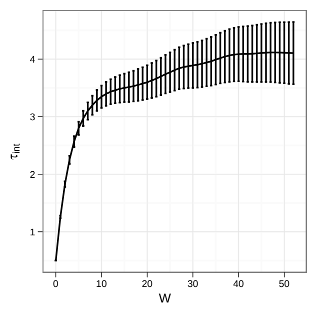
(a) $\tau_{\mathtt{int}}$ vs. W for β_0



(b) $\tau_{\mathtt{int}}$ vs. W for θ_{δ_V}



(c) $\tau_{\mathtt{int}}$ vs. W for θ_{δ_E}



(d) $\tau_{\mathtt{int}}$ vs. W for σ^2

Figure A.8: MCMC $\tau_{\mathtt{int}}$ vs. W for BBV DTAPs with a uniformly random decision Policy with $\delta_V \in [1.7, 1.8], \delta_E \in [1.7, 1.8]$

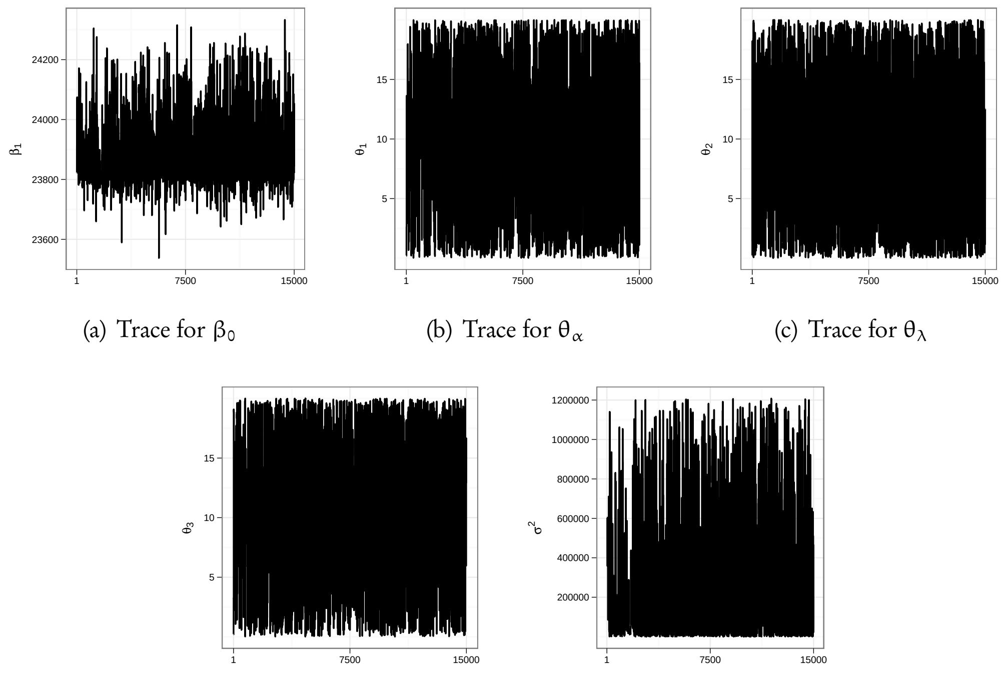
A.1.2 SARSA(0) and ϵ-greedy Policy

A.1.2.1 $\delta_V = 1.2, \delta_E = 1.2$

The summary of the Kriging estimation is given in Table A.2 with a MCMC chain size of 91. The Kriging model is assessed using 100 validation locations selected using a space-filling LHS design.

Table A.2: Kriging Summary of BBV DTAPs with SARSA(0) and ϵ-greedy Policy with $\delta_V = 1.2, \delta_E = 1.2$

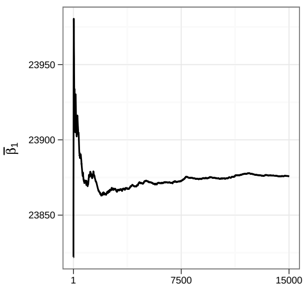
Parameter	Value	95% CI half-width
β_0	23885.56	18.5
θ_α	9.9	1.26
θ_λ	8.78	1.24
θ_η	11.44	1.17
σ^2	257315.7	61752.0
R^2	0.999	
RMSE	195.32	

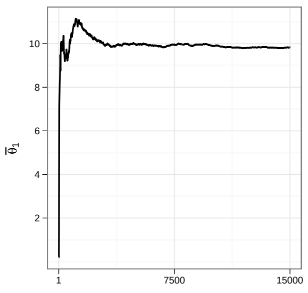


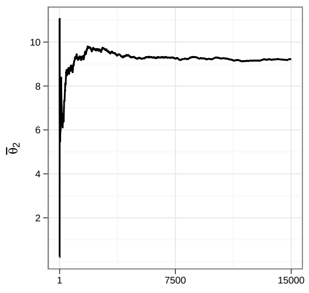
(a) Trace for β_0 (b) Trace for θ_α (c) Trace for θ_λ

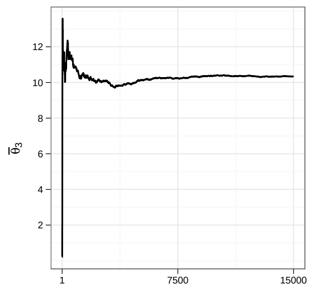
(d) Trace for θ_η (e) Trace for σ^2

Figure A.9: MCMC Trace for DTAPs with SARSA(0) and ϵ-greedy Policy with $\delta_V = 1.2, \delta_E = 1.2$


(a) Equilibration for β_0


(b) Equilibration for θ_α


(c) Equilibration for θ_λ


(d) Equilibration for θ_η

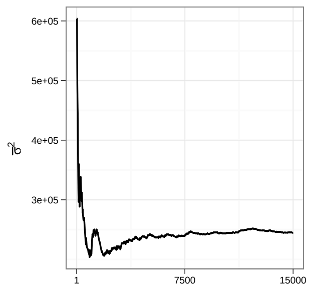
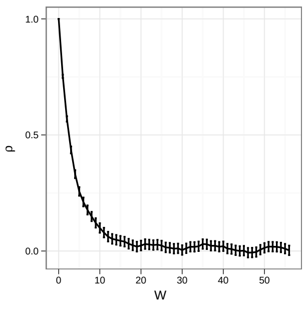
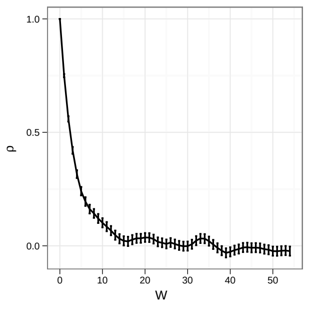
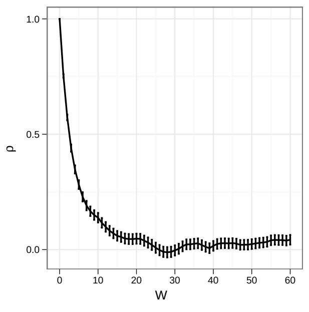
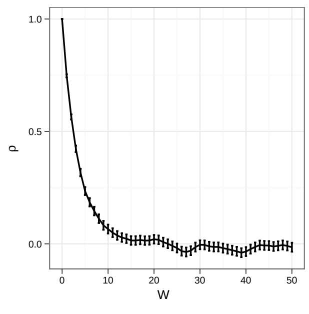

(e) Equilibration for σ^2

Figure A.10: MCMC Mean for DTAPs with $SARSA(0)$ and ϵ-greedy Policy with $\delta_V = 1.2, \delta_E = 1.2$


(a) Autocorrelation for β_0


(b) Autocorrelation for θ_α


(c) Autocorrelation for θ_λ


(d) Autocorrelation for θ_η

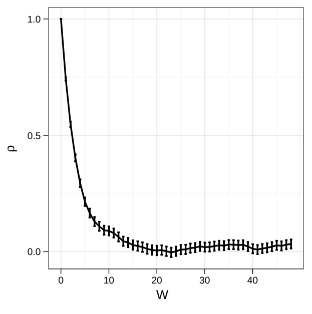
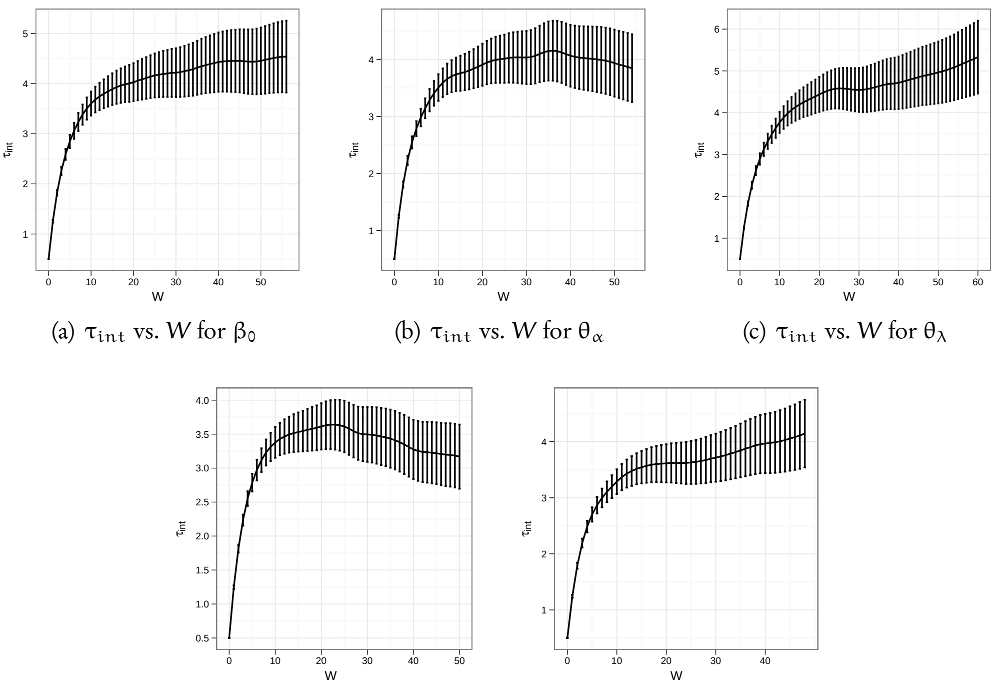

(e) Autocorrelation for σ^2

Figure A.11: MCMC Autocorrelation for DTAPs with $SARSA(0)$ and ϵ-greedy Policy with $\delta_V = 1.2, \delta_E = 1.2$



(a) $\tau_{\mathtt{int}}$ vs. W for β_0 (b) $\tau_{\mathtt{int}}$ vs. W for θ_α (c) $\tau_{\mathtt{int}}$ vs. W for θ_λ

(d) $\tau_{\mathtt{int}}$ vs. W for θ_η (e) $\tau_{\mathtt{int}}$ vs. W for σ^2

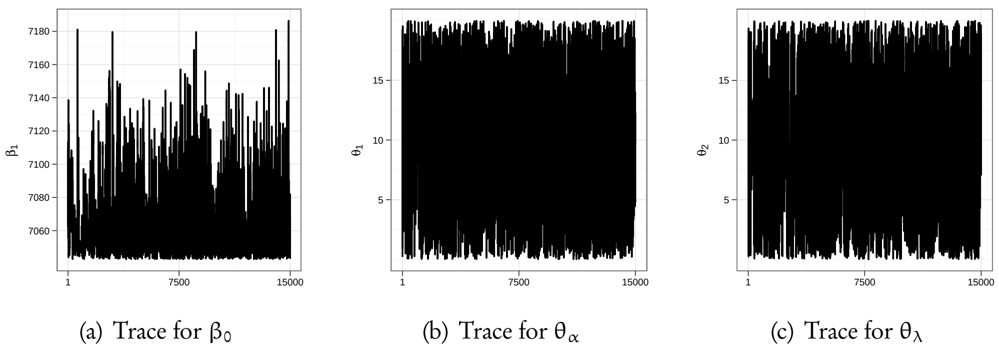
Figure A.12: MCMC $\tau_{\mathtt{int}}$ vs. W for DTAPs with $SARSA(0)$ and ϵ-greedy Policy with $\delta_V = 1.2, \delta_E = 1.2$

A.1.2.2 $\delta_V = 1.8, \delta_E = 1.8$

The summary of the Kriging estimation is given in Table A.3 with a MCMC chain size of 161. The Kriging model is assessed using 100 validation locations selected using a space-filling LHS design.

Table A.3: Kriging Summary of BBV DTAPs with $SARSA(0)$ and ϵ-greedy Policy with $\delta_V = 1.8, \delta_E = 1.8$

Parameter	Value	95% CI half-width
β_0	7062.11	2.93
θ_α	9.61	0.94
θ_λ	9.12	0.88
θ_η	9.1	0.93
σ^2	18471.18	4084.35
R^2	0.999	
RMSE	58.58	


(a) Trace for β_0 (b) Trace for θ_α (c) Trace for θ_λ

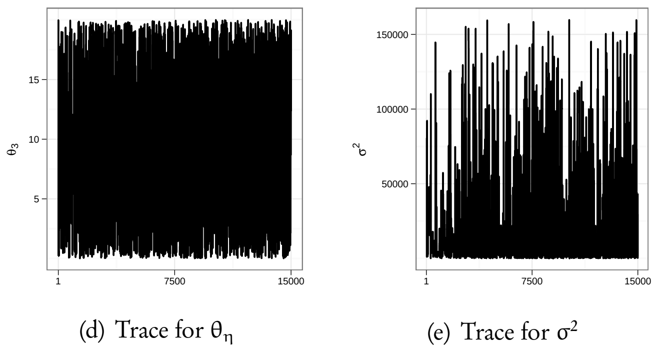
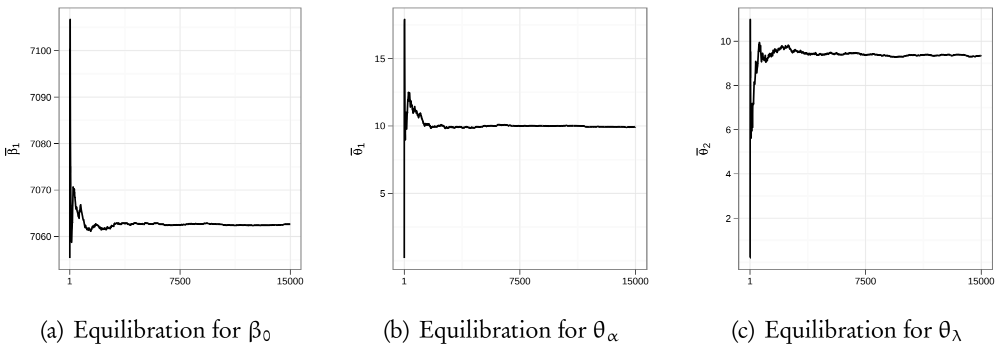

(d) Trace for θ_η (e) Trace for σ^2

Figure A.13: MCMC Trace for DTAPs with $SARSA(0)$ and ϵ-greedy Policy with $\delta_V = 1.8, \delta_E = 1.8$


(a) Equilibration for β_0 (b) Equilibration for θ_α (c) Equilibration for θ_λ

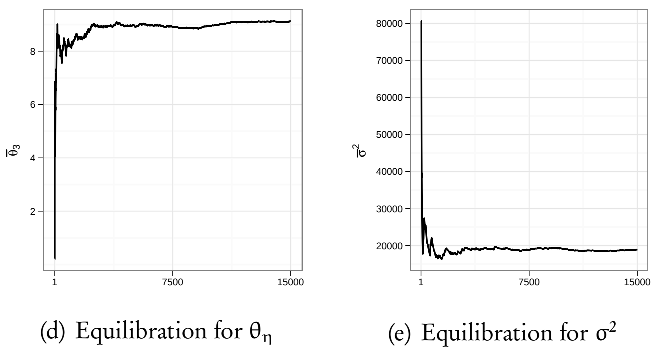
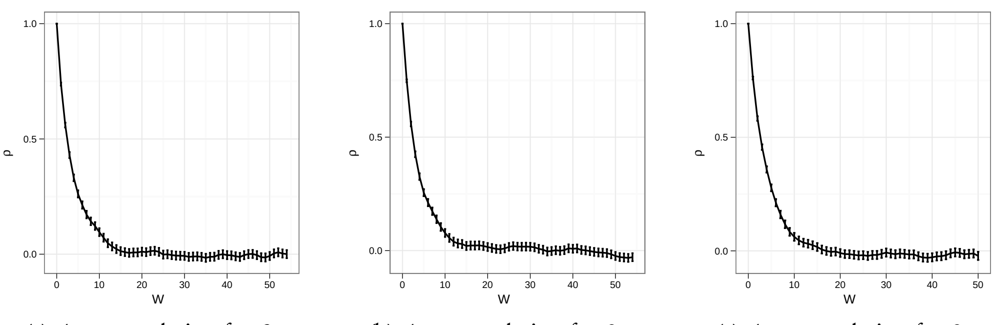
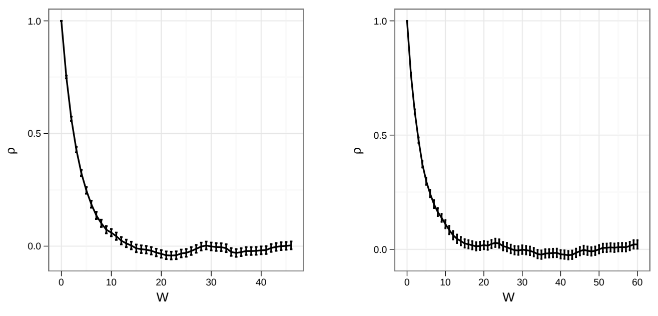

(d) Equilibration for θ_η (e) Equilibration for σ^2

Figure A.14: MCMC Mean for DTAPs with $SARSA(0)$ and ϵ-greedy Policy with $\delta_V = 1.8, \delta_E = 1.8$

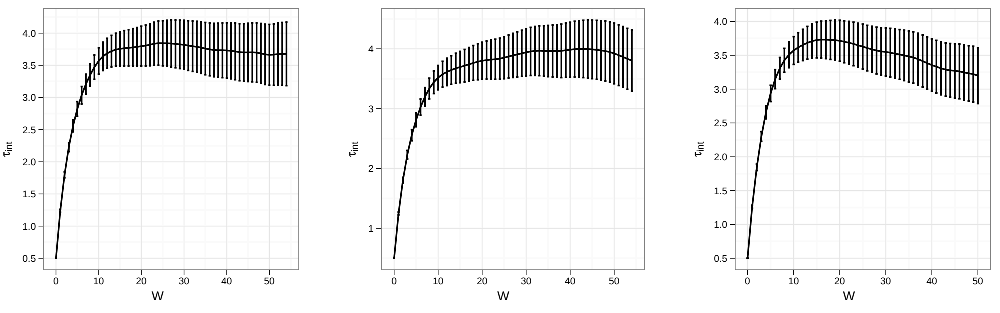


(a) Autocorrelation for β_0 (b) Autocorrelation for θ_α (c) Autocorrelation for θ_λ

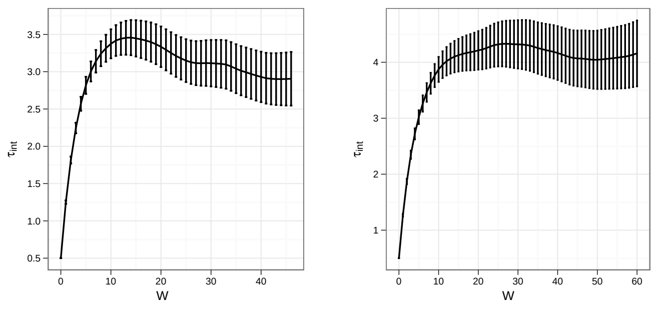


(d) Autocorrelation for θ_η (e) Autocorrelation for σ^2

Figure A.15: MCMC Autocorrelation for DTAPs with SARSA(0) and ϵ-greedy Policy with $\delta_V = 1.8, \delta_E = 1.8$



(a) τ_{int} vs. W for β_0 (b) τ_{int} vs. W for θ_α (c) τ_{int} vs. W for θ_λ



(d) τ_{int} vs. W for θ_η (e) τ_{int} vs. W for σ^2

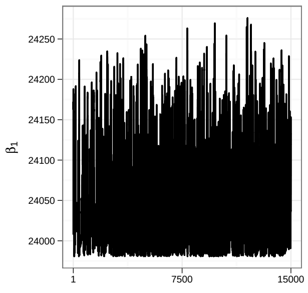
Figure A.16: MCMC τ_{int} vs. W for DTAPs with SARSA(0) and ϵ-greedy Policy with $\delta_V = 1.8, \delta_E = 1.8$

A.1.2.3 $\delta_V = 1.2, \delta_E = 1.2$ and outsourced Function Approximators

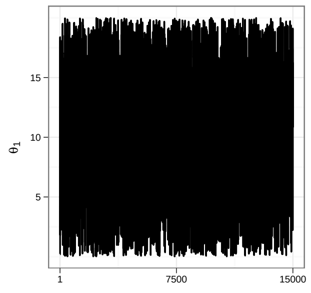
The summary of the Kriging estimation is given in Table A.4 with a MCMC chain size of 161. The Kriging model is assessed using 100 validation locations selected using a space-filling LHS design.

Table A.4: Kriging Summary of BBV DTAPs with SARSA(0) and ϵ-greedy Policy with $\delta_V = 1.2, \delta_E = 1.2$

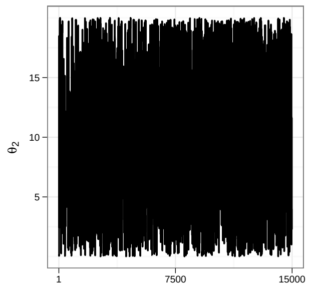
Parameter	Value	95% CI half-width
β_0	24048.4	9.1
θ_α	9.37	0.91
θ_λ	9.26	0.90
θ_η	10.97	0.95
σ^2	188532.33	35739.2
R^2	0.999	
RMSE	185.83	



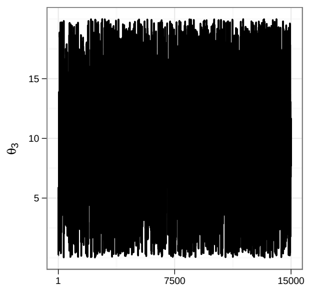
(a) Trace for β_0



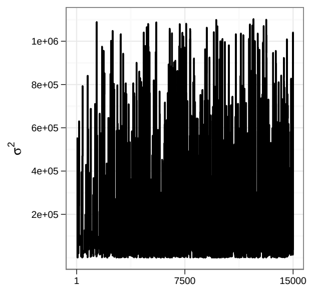
(b) Trace for θ_α



(c) Trace for θ_λ

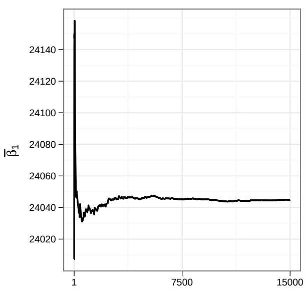


(d) Trace for θ_η

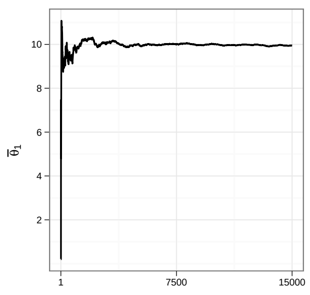


(e) Trace for σ^2

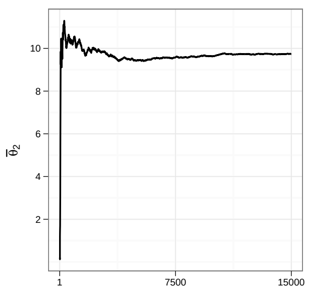
Figure A.17: MCMC Trace for DTAPs with SARSA(0) and ϵ-greedy Policy with $\delta_V = 1.2, \delta_E = 1.2$ and outsourced Function Approximators



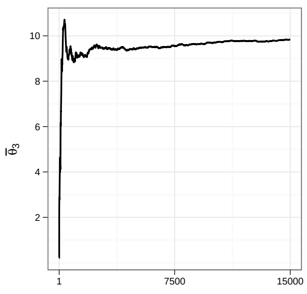
(a) Equilibration for β_0



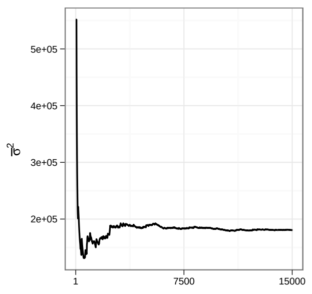
(b) Equilibration for θ_α



(c) Equilibration for θ_λ

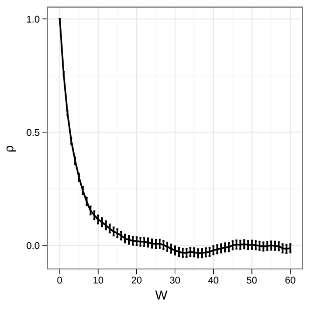


(d) Equilibration for θ_η

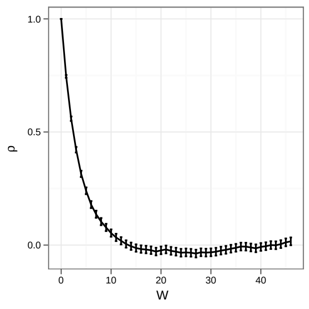


(e) Equilibration for σ^2

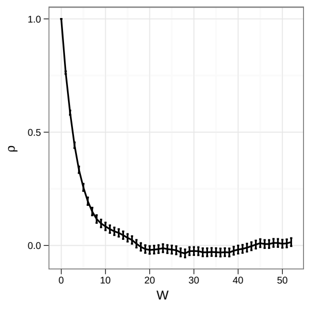
Figure A.18: MCMC Mean for DTAPs with $SARSA(0)$ and ϵ-greedy Policy with $\delta_V = 1.2, \delta_E = 1.2$ and outsourced Function Approximators



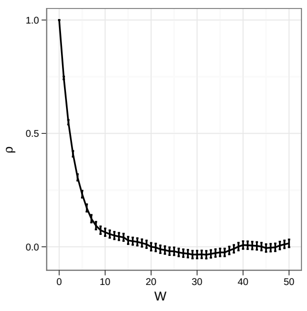
(a) Autocorrelation for β_0



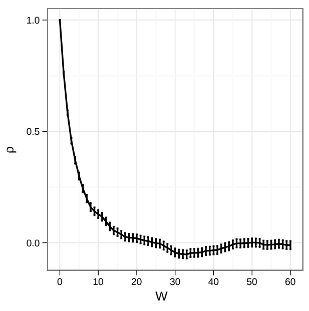
(b) Autocorrelation for θ_α



(c) Autocorrelation for θ_λ

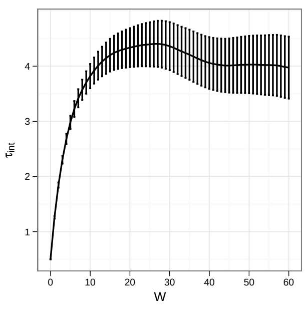


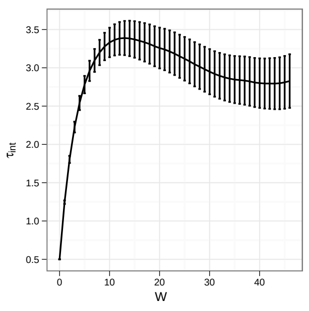
(d) Autocorrelation for θ_η

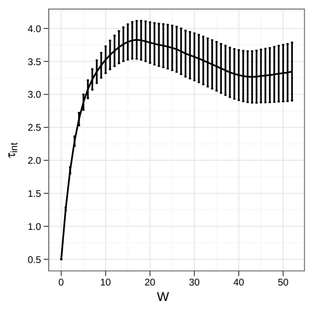


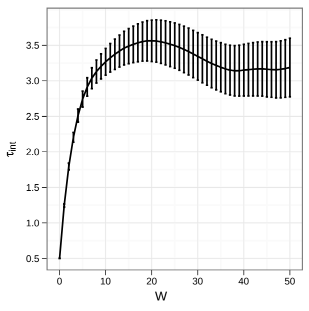
(e) Autocorrelation for σ^2

Figure A.19: MCMC Autocorrelation for DTAPs with $SARSA(0)$ and ϵ-greedy Policy with $\delta_V = 1.2, \delta_E = 1.2$ and outsourced Function Approximators


(a) τ_{int} vs. W for β_0


(b) τ_{int} vs. W for θ_α


(c) τ_{int} vs. W for θ_λ


(d) τ_{int} vs. W for θ_η

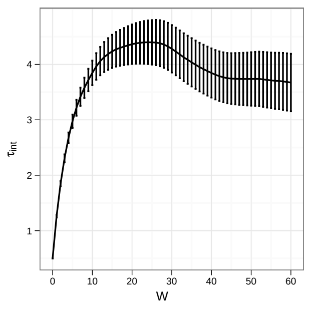

(e) τ_{int} vs. W for σ^2

Figure A.20: MCMC τ_{int} vs. W for DTAPs with $\mathsf{SARSA}(0)$ and ϵ-greedy Policy with $\delta_V = 1.2, \delta_E = 1.2$ and outsourced Function Approximators

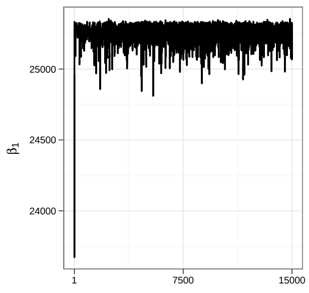
A.1.3 $\mathsf{SARSA}(0)$ and WPL Policy

A.1.3.1 $\delta_V = 1.2, \delta_E = 1.2$

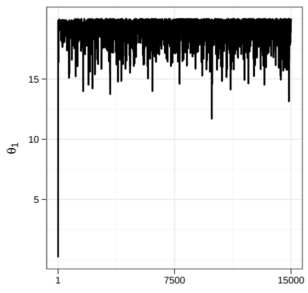
The summary of the Kriging estimation is given in Table A.5 with a MCMC chain size of 161. The Kriging model is assessed using 100 validation locations selected using a space-filling LHS design.

Table A.5: Kriging Summary of BBV DTAPs with $\mathsf{SARSA}(0)$ and WPL Policy with $\delta_V = 1.2, \delta_E = 1.2$

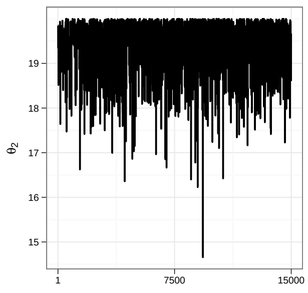
Parameter	Value	95% CI half-width
β_0	25261.39	8.95
θ_α	19.11	0.12
θ_λ	19.41	0.085
θ_η	18.62	0.22
θ_ζ	17.65	0.37
σ^2	7191733.2	40746.63
R^2	0.999	
RMSE	57.28	



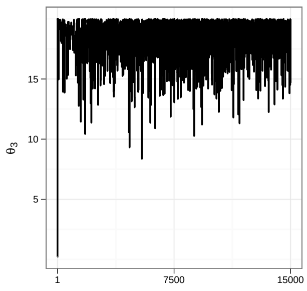
(a) Trace for β_0



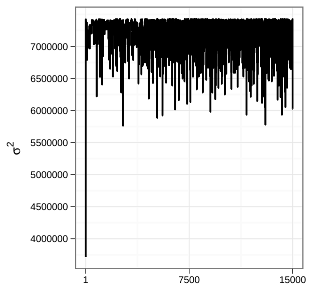
(b) Trace for θ_α



(c) Trace for θ_λ

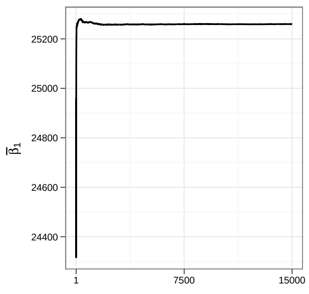


(d) Trace for θ_η

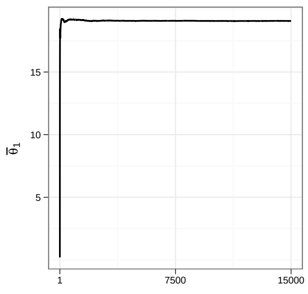


(e) Trace for σ^2

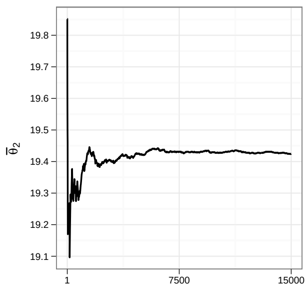
Figure A.21: MCMC Trace for DTAPs with $\mathsf{SARSA(0)}$ and WPL Policy with $\delta_V = 1.2, \delta_E = 1.2$



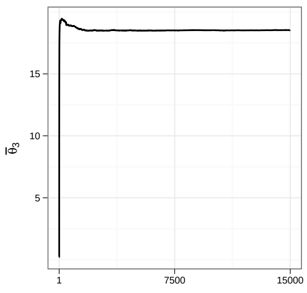
(a) Equilibration for β_0



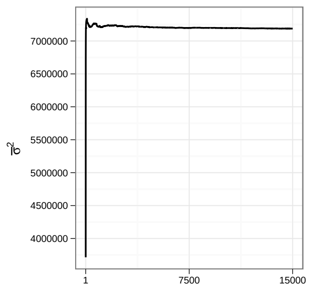
(b) Equilibration for θ_α



(c) Equilibration for θ_λ

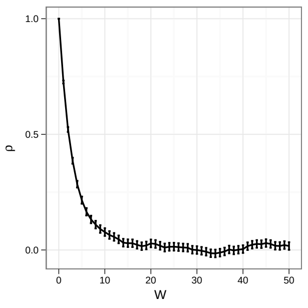


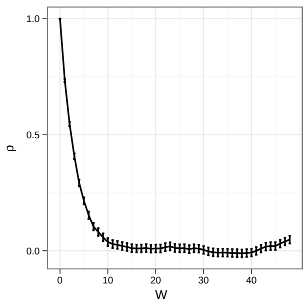
(d) Equilibration for θ_η

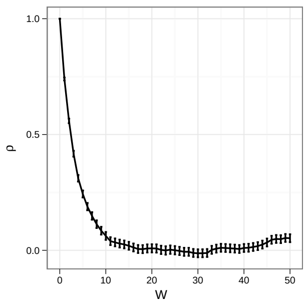


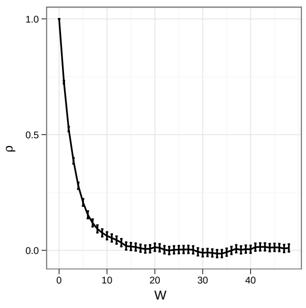
(e) Equilibration for σ^2

Figure A.22: MCMC Mean for DTAPs with $\mathsf{SARSA(0)}$ and WPL Policy with $\delta_V = 1.2, \delta_E = 1.2$


(a) Autocorrelation for β_0


(b) Autocorrelation for θ_α


(c) Autocorrelation for θ_λ


(d) Autocorrelation for θ_η

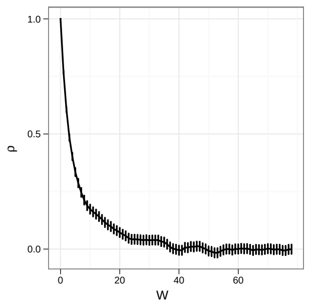
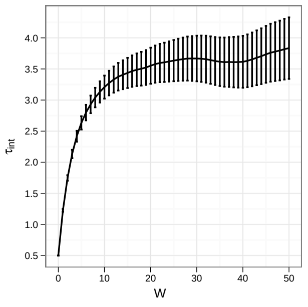
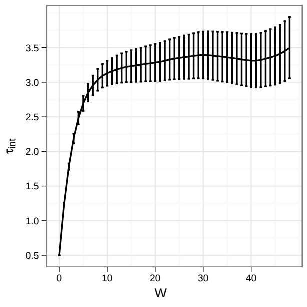
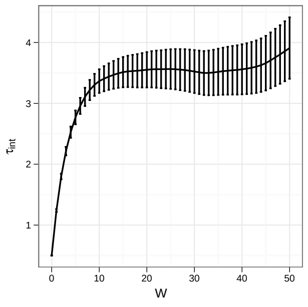
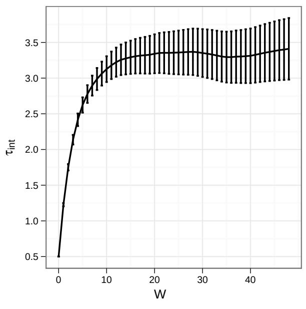

(e) Autocorrelation for σ^2

Figure A.23: MCMC Autocorrelation for DTAPs with $\mathsf{SARSA}(0)$ and WPL Policy with $\delta_V = 1.2, \delta_E = 1.2$


(a) $\tau_{\mathtt{int}}$ vs. W for β_0


(b) $\tau_{\mathtt{int}}$ vs. W for θ_α


(c) $\tau_{\mathtt{int}}$ vs. W for θ_λ


(d) $\tau_{\mathtt{int}}$ vs. W for θ_η

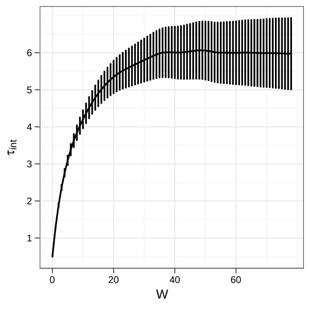

(e) $\tau_{\mathtt{int}}$ vs. W for σ^2

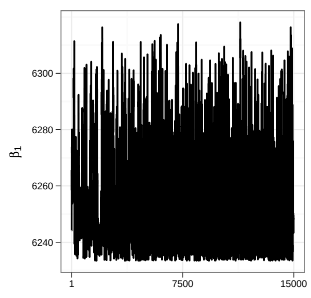
Figure A.24: MCMC $\tau_{\mathtt{int}}$ vs. W for DTAPs with $\mathsf{SARSA}(0)$ and WPL Policy with $\delta_V = 1.2, \delta_E = 1.2$

A.1.3.2 $\delta_V = 1.8, \delta_E = 1.8$

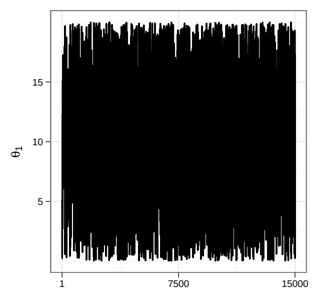
The summary of the Kriging estimation is given in Table A.6 with a MCMC chain size of 161. The Kriging model is assessed using 100 validation locations selected using a space-filling LHS design.

Table A.6: Kriging Summary of BBV DTAPs with SARSA(0) and WPL Policy with $\delta_V = 1.8, \delta_E = 1.8$

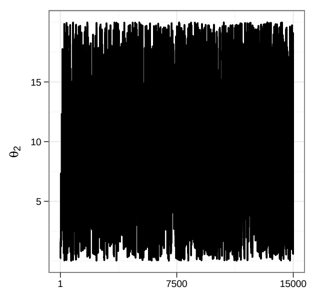
Parameter	Value	95% CI half-width
β_0	6253.96	2.59
θ_α	9.72	0.92
θ_λ	10.5	0.89
θ_η	9.04	0.91
θ_ζ	10.44	0.94
σ^2	13573.12	2262.88
R^2	0.999	
RMSE	57.28	



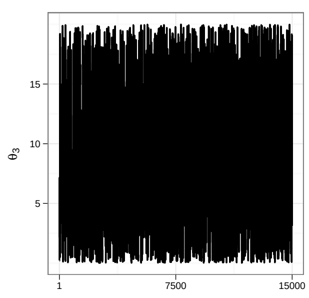
(a) Trace for β_0



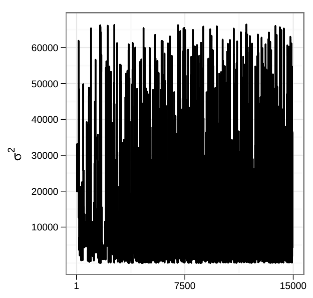
(b) Trace for θ_α



(c) Trace for θ_λ

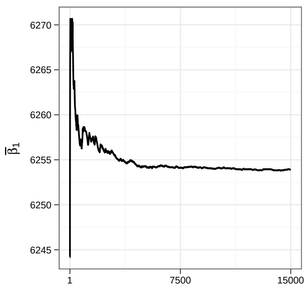


(d) Trace for θ_η

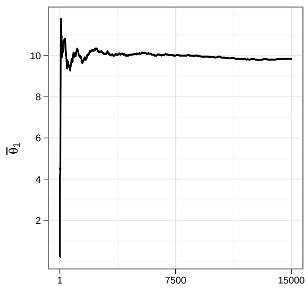


(e) Trace for σ^2

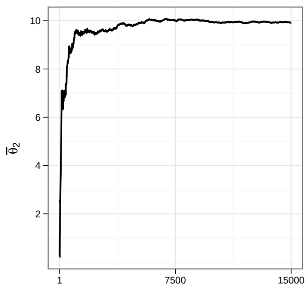
Figure A.25: MCMC Trace for DTAPs with SARSA(0) and WPL Policy with $\delta_V = 1.8, \delta_E = 1.8$



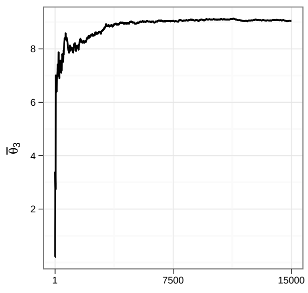
(a) Equilibration for β_0



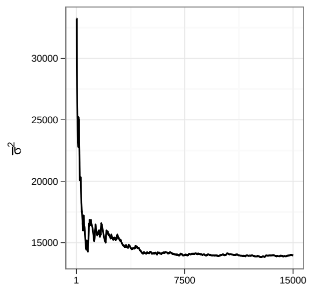
(b) Equilibration for θ_α



(c) Equilibration for θ_λ

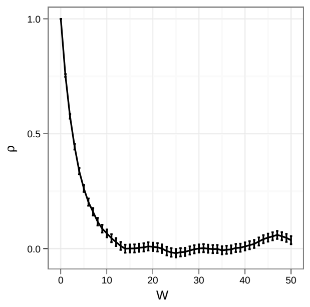


(d) Equilibration for θ_η

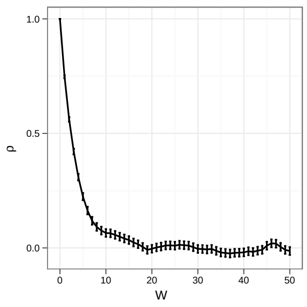


(e) Equilibration for σ^2

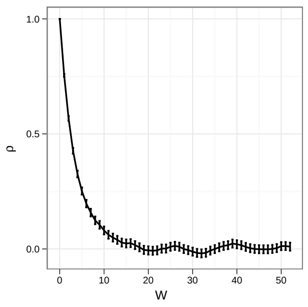
Figure A.26: MCMC Mean for DTAPs with $SARSA(0)$ and WPL Policy with $\delta_V = 1.8, \delta_E = 1.8$



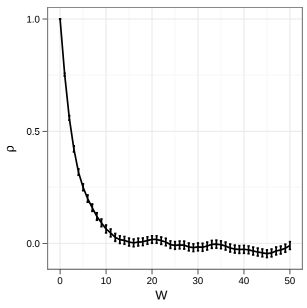
(a) Autocorrelation for β_0



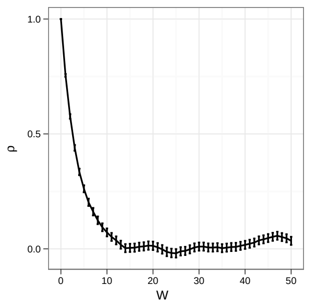
(b) Autocorrelation for θ_α



(c) Autocorrelation for θ_λ

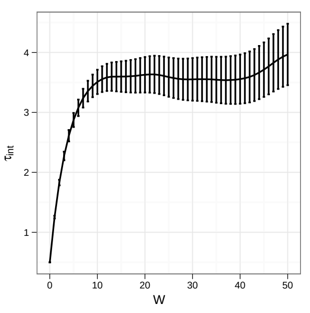


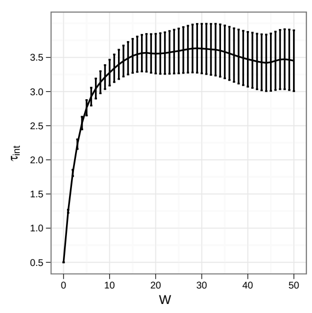
(d) Autocorrelation for θ_η

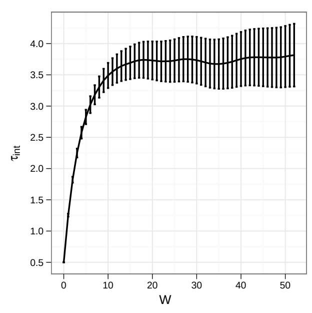


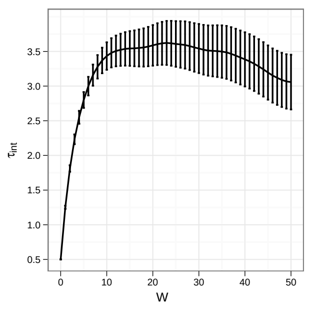
(e) Autocorrelation for σ^2

Figure A.27: MCMC Autocorrelation for DTAPs with $SARSA(0)$ and WPL Policy with $\delta_V = 1.8, \delta_E = 1.8$


(a) τ_{int} vs. W for β_0


(b) τ_{int} vs. W for θ_α


(c) τ_{int} vs. W for θ_λ


(d) τ_{int} vs. W for θ_η

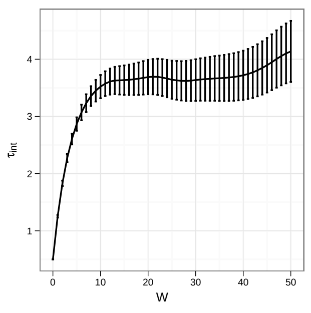

(e) τ_{int} vs. W for σ^2

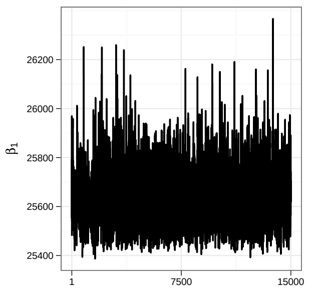
Figure A.28: MCMC τ_{int} vs. W for DTAPs with SARSA(0) and WPL Policy with $\delta_V = 1.8, \delta_E = 1.8$

A.1.3.3 $\delta_V = 1.2, \delta_E = 1.2$ and outsourced Function Approximators

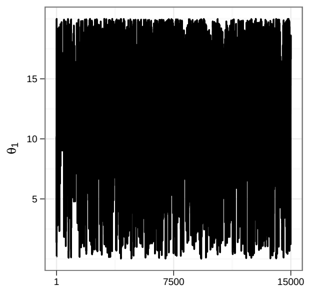
The summary of the Kriging estimation is given in Table A.7 with a MCMC chain size of 161. The Kriging model is assessed using 100 validation locations selected using a space-filling LHS design.

Table A.7: Kriging Summary of BBV DTAPs with SARSA(0) and ϵ-greedy Policy with $\delta_V = 1.2, \delta_E = 1.2$

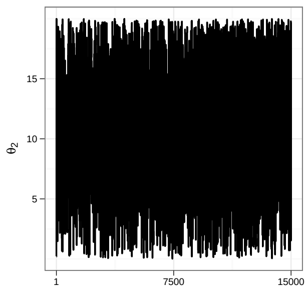
Parameter	Value	95% CI half-width
β_0	25604.71	17.57
θ_α	12.88	0.87
θ_λ	10.57	0.78
θ_η	18.02	0.29
θ_ζ	15.75	0.55
σ^2	2021790	38364.47
R^2	0.999	
RMSE	318.69	



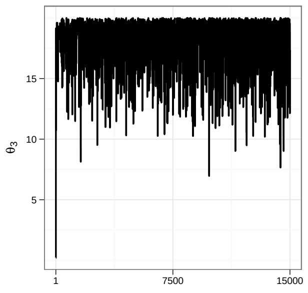
(a) Trace for β_0



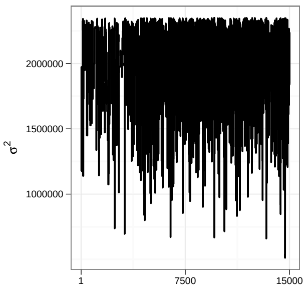
(b) Trace for θ_α



(c) Trace for θ_λ

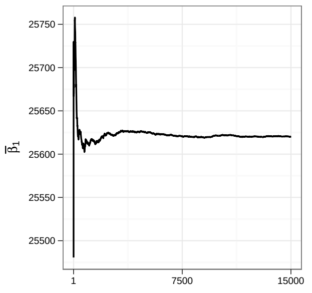


(d) Trace for θ_η

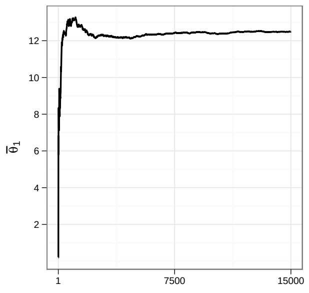


(e) Trace for σ^2

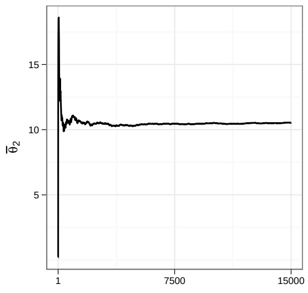
Figure A.29: MCMC Trace for DTAPs with SARSA(0) and ϵ-greedy Policy with $\delta_V = 1.2, \delta_E = 1.2$ and outsourced Function Approximators



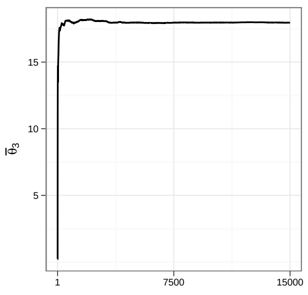
(a) Equilibration for β_0



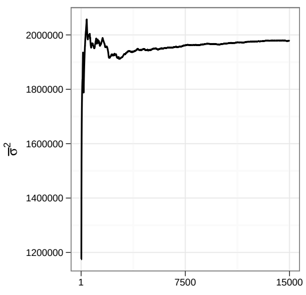
(b) Equilibration for θ_α



(c) Equilibration for θ_λ

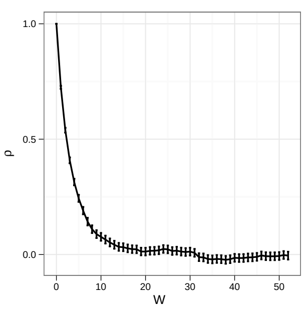


(d) Equilibration for θ_η

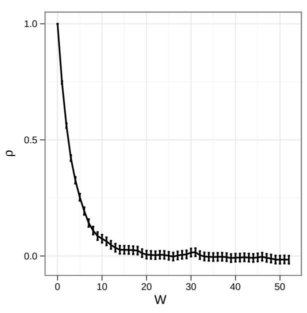


(e) Equilibration for σ^2

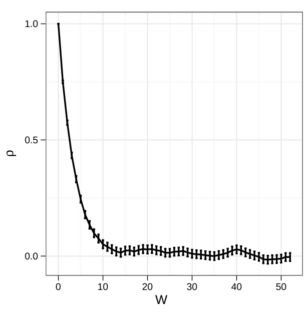
Figure A.30: MCMC Mean for DTAPs with SARSA(0) and ϵ-greedy Policy with $\delta_V = 1.2, \delta_E = 1.2$ and outsourced Function Approximators



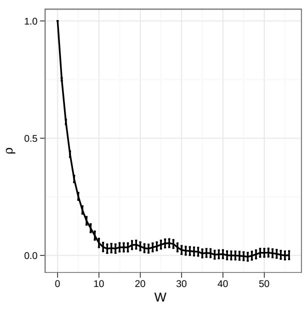
(a) Autocorrelation for β_0



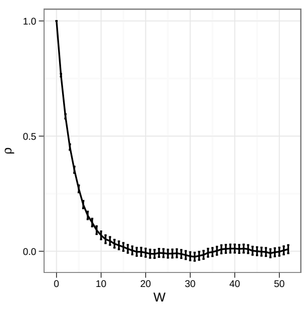
(b) Autocorrelation for θ_α



(c) Autocorrelation for θ_λ

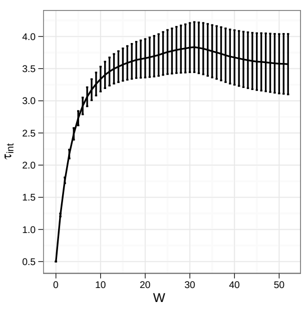


(d) Autocorrelation for θ_η

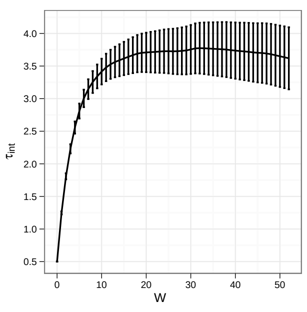


(e) Autocorrelation for σ^2

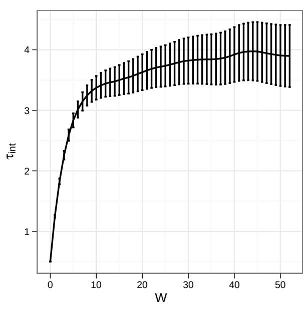
Figure A.31: MCMC Autocorrelation for DTAPs with SARSA(0) and ϵ-greedy Policy with $\delta_V = 1.2, \delta_E = 1.2$ and outsourced Function Approximators



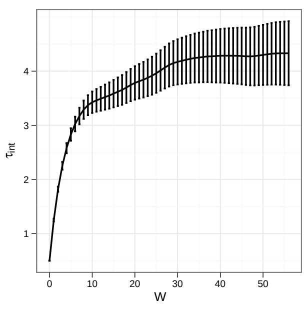
(a) τ_{int} vs. W for β_0



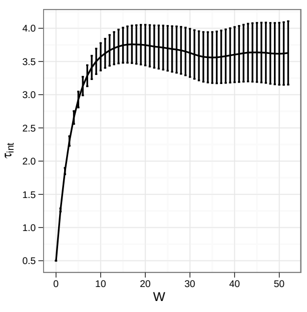
(b) τ_{int} vs. W for θ_α



(c) τ_{int} vs. W for θ_λ



(d) τ_{int} vs. W for θ_η



(e) τ_{int} vs. W for σ^2

Figure A.32: MCMC τ_{int} vs. W for DTAPs with SARSA(0) and ϵ-greedy Policy with $\delta_V = 1.2, \delta_E = 1.2$ and outsourced Function Approximators

A.2 ER DTAPs

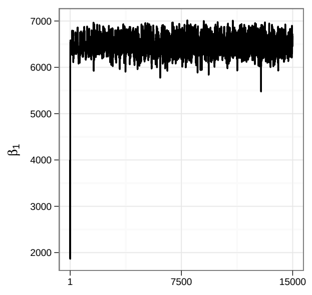
A.2.1 Uniformly Random Decision Policy

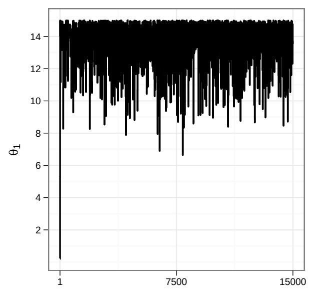
The summary of the Kriging estimation is given in Table A.8 with a MCMC chain size of 161. The Kriging model is assessed using 100 validation locations selected using a space-filling LHS design.

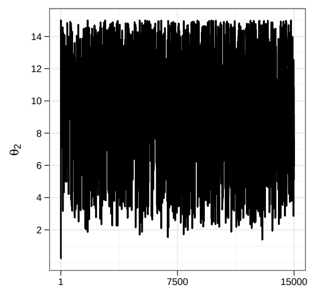
Table A.8: Kriging Summary of ER DTAPs with a uniform random decision Policy

	$\delta_V, \delta_E \in [1.2, 1.3]$		$\delta_V, \delta_E \in [1.7, 1.8]$	
Variable	Value	95% CI half-width	Value	95% CI half-width
β_0	6454.67	28.04	1692.75	1.9
θ_{δ_V}	13.65	0.182	19.71	0.057
θ_{δ_E}	9.496	0.536	19.86	0.029
σ^2	2472331.63	39279.84	53527.07	173.378
R^2	0.80		0.815	
RMSE	60.88		4.676	

A.2.1.1 $\delta_V \in [1.2, 1.3], \delta_E \in [1.2, 1.3]$


(a) Trace for β_0


(b) Trace for θ_{δ_V}


(c) Trace for θ_{δ_E}

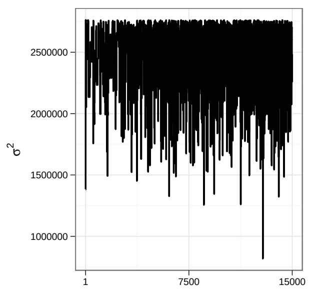
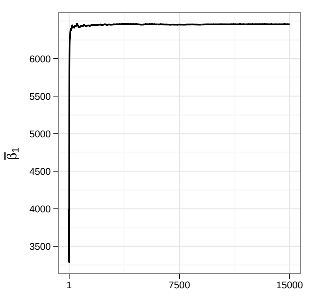
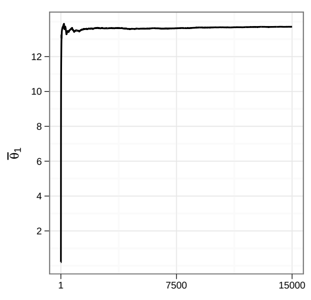

(d) Trace for σ^2

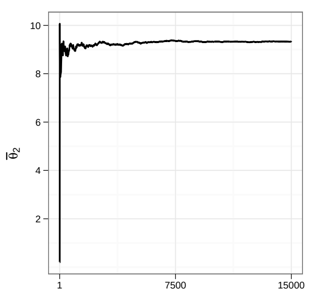
Figure A.33: MCMC Trace for ER DTAPs with a uniformly random decision Policy with $\delta_V \in [1.2, 1.3], \delta_E \in [1.2, 1.3]$



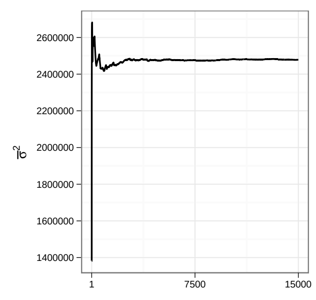
(a) Equilibration for β_0



(b) Equilibration for θ_{δ_V}

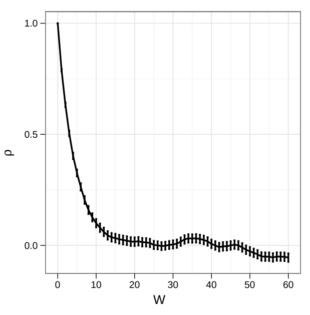


(c) Equilibration for θ_{δ_E}

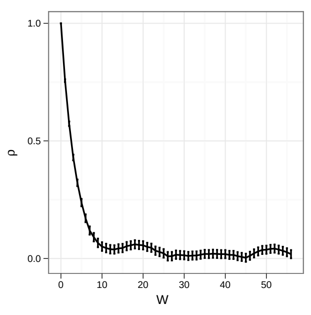


(d) Equilibration for σ^2

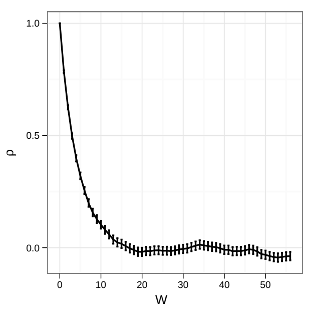
Figure A.34: MCMC Mean for ER DTAPs with a uniformly random decision Policy with $\delta_V \in [1.2, 1.3], \delta_E \in [1.2, 1.3]$



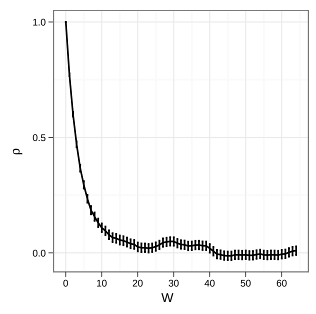
(a) Autocorrelation for β_0



(b) Autocorrelation for θ_{δ_V}

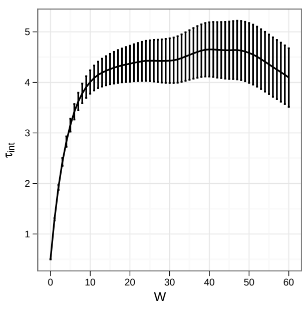


(c) Autocorrelation for θ_{δ_E}

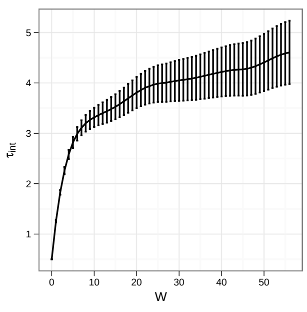


(d) Autocorrelation for σ^2

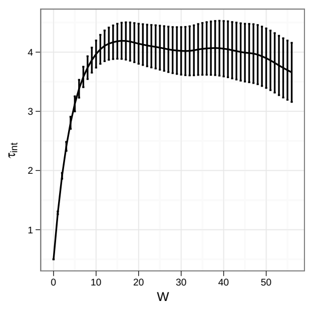
Figure A.35: MCMC Autocorrelation for ER DTAPs with a uniformly random decision Policy with $\delta_V \in [1.2, 1.3], \delta_E \in [1.2, 1.3]$



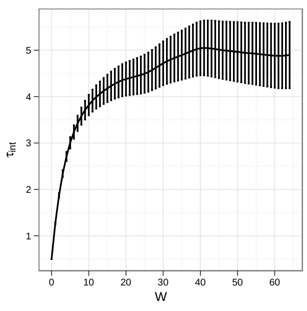
(a) τ_{int} vs. W for β_0



(b) τ_{int} vs. W for θ_{δ_V}



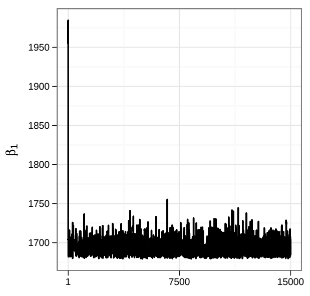
(c) τ_{int} vs. W for θ_{δ_E}



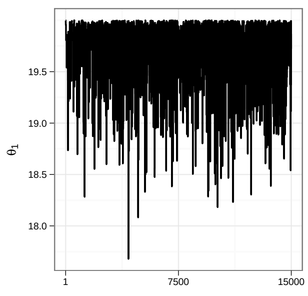
(d) τ_{int} vs. W for σ^2

Figure A.36: MCMC τ_{int} vs. W for ER DTAPs with a uniformly random decision Policy with $\delta_V \in [1.2, 1.3], \delta_E \in [1.2, 1.3]$

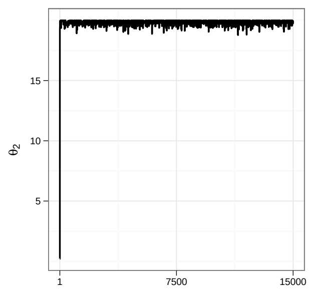
A.2.1.2 $\delta_V \in [1.7, 1.8], \delta_E \in [1.7, 1.8]$



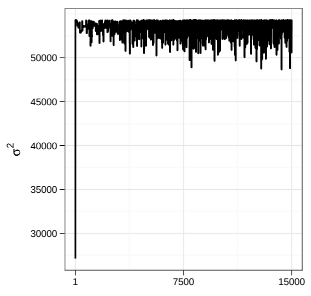
(a) Trace for β_0



(b) Trace for θ_{δ_V}

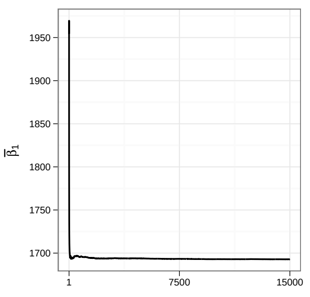


(c) Trace for θ_{δ_E}

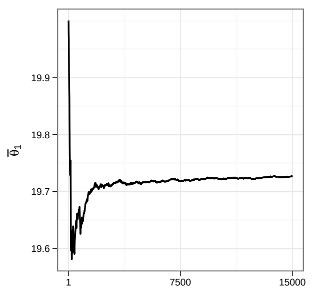


(d) Trace for σ^2

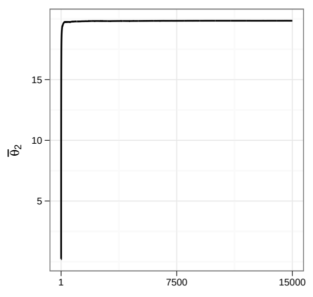
Figure A.37: MCMC Trace for ER DTAPs with a uniformly random decision Policy with $\delta_V \in [1.7, 1.8], \delta_E \in [1.7, 1.8]$



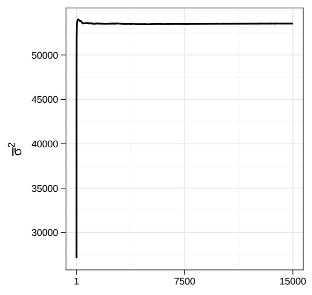
(a) Equilibration for β_0



(b) Equilibration for θ_{δ_V}

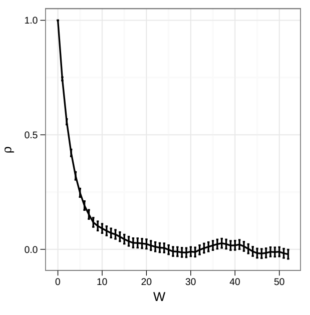


(c) Equilibration for θ_{δ_E}

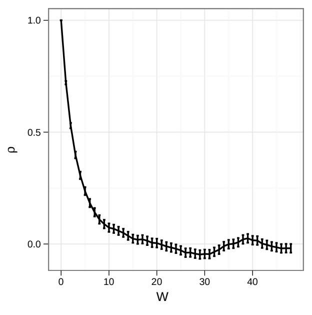


(d) Equilibration for σ^2

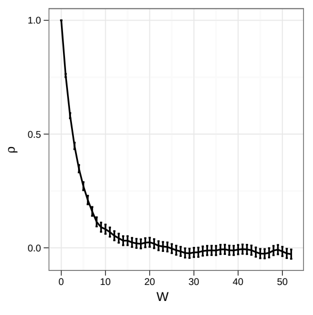
Figure A.38: MCMC Mean for ER DTAPs with a uniformly random decision Policy with $\delta_V \in [1.7, 1.8], \delta_E \in [1.7, 1.8]$



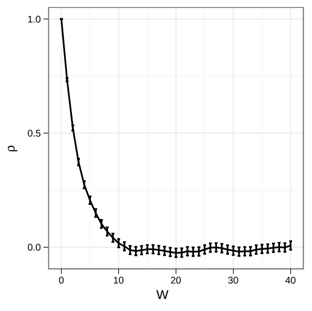
(a) Autocorrelation for β_0



(b) Autocorrelation for θ_{δ_V}

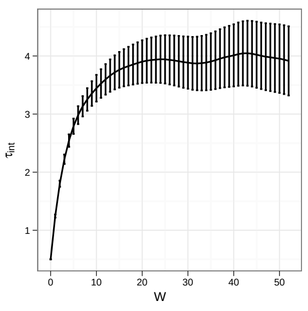


(c) Autocorrelation for θ_{δ_E}

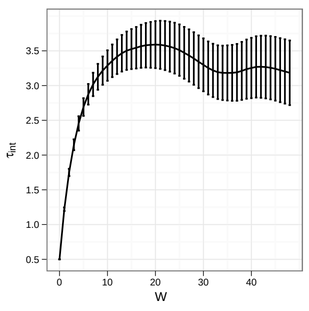


(d) Autocorrelation for σ^2

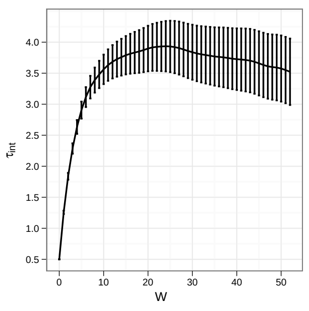
Figure A.39: MCMC Autocorrelation for ER DTAPs with a uniformly random decision Policy with $\delta_V \in [1.7, 1.8], \delta_E \in [1.7, 1.8]$



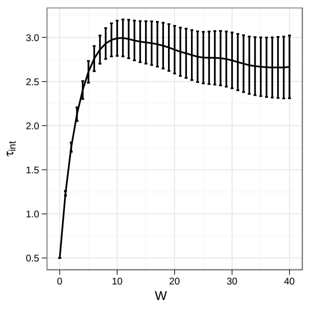
(a) $\tau_{\mathtt{int}}$ vs. W for β_0



(b) $\tau_{\mathtt{int}}$ vs. W for θ_{δ_V}



(c) $\tau_{\mathtt{int}}$ vs. W for θ_{δ_E}



(d) $\tau_{\mathtt{int}}$ vs. W for σ^2

Figure A.40: MCMC $\tau_{\mathtt{int}}$ vs. W for ER DTAPs with a uniformly random decision Policy with $\delta_V \in [1.7, 1.8], \delta_E \in [1.7, 1.8]$

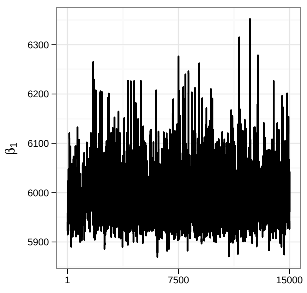
A.2.2 $\mathsf{SARSA}(0)$ and ϵ-greedy Policy

A.2.2.1 $\delta_V = 1.2, \delta_E = 1.2$

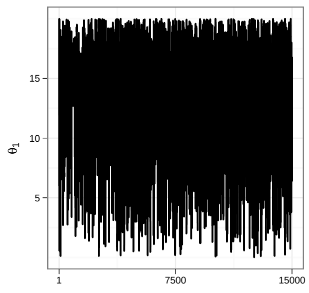
The summary of the Kriging estimation is given in Table A.9 with a MCMC chain size of 161. The Kriging model is assessed using 100 validation locations selected using a space-filling LHS design.

Table A.9: Kriging Summary of ER DTAPs with $\mathsf{SARSA}(0)$ and ϵ-greedy Policy with $\delta_V = 1.2, \delta_E = 1.2$

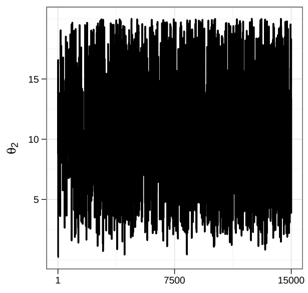
Parameter	Value	95% CI half-width
β_0	5994.95	8.9
θ_α	12.45	0.79
θ_λ	9.96	0.68
θ_η	12.08	0.67
σ^2	183858.46	6630.16
R^2	0.999	
RMSE	84.85	



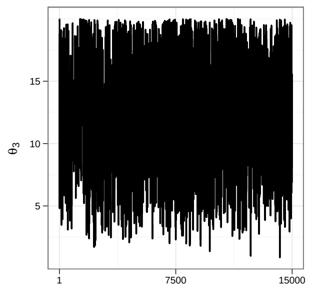
(a) Trace for β_0



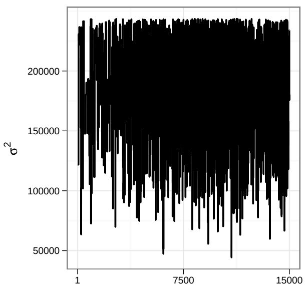
(b) Trace for θ_α



(c) Trace for θ_λ

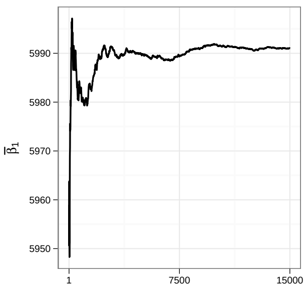


(d) Trace for θ_η

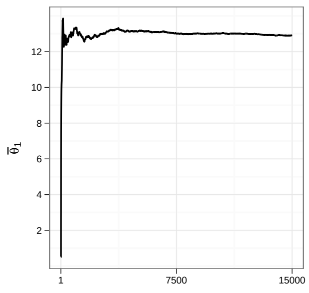


(e) Trace for σ^2

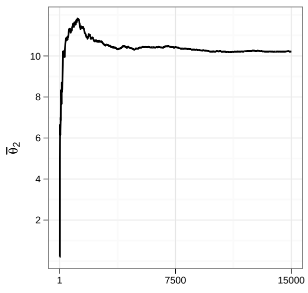
Figure A.41: MCMC Trace for ER DTAPs with SARSA(0) and ϵ-greedy Policy with $\delta_V = 1.2, \delta_E = 1.2$



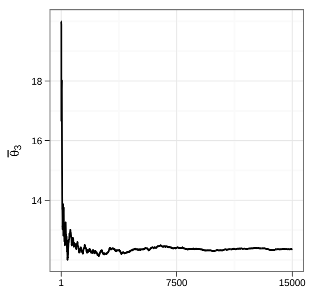
(a) Equilibration for β_0



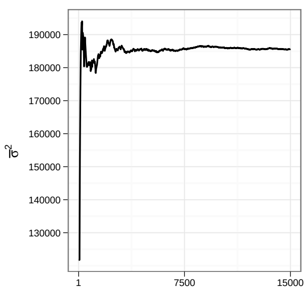
(b) Equilibration for θ_α



(c) Equilibration for θ_λ

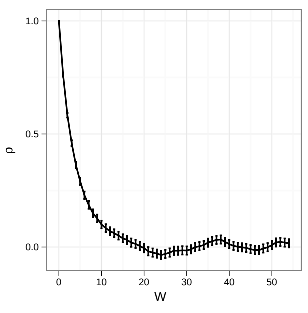


(d) Equilibration for θ_η

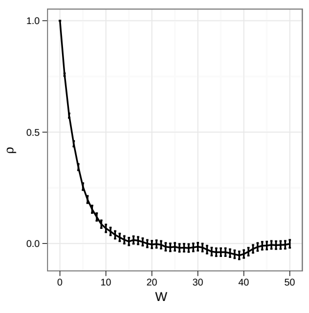


(e) Equilibration for σ^2

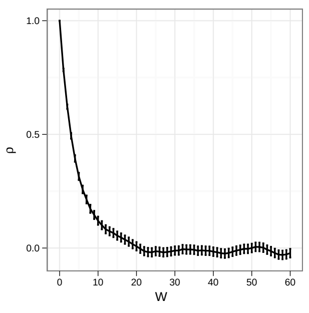
Figure A.42: MCMC Mean for ER DTAPs with SARSA(0) and ϵ-greedy Policy with $\delta_V = 1.2, \delta_E = 1.2$



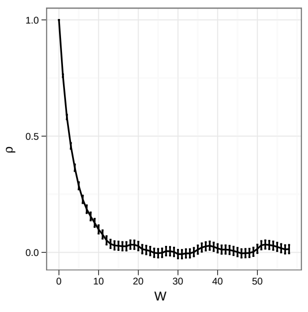
(a) Autocorrelation for β_0



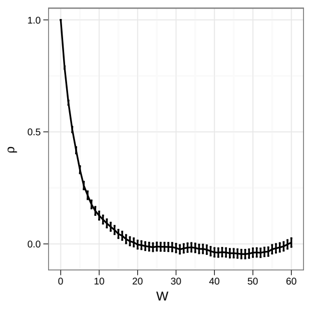
(b) Autocorrelation for θ_α



(c) Autocorrelation for θ_λ

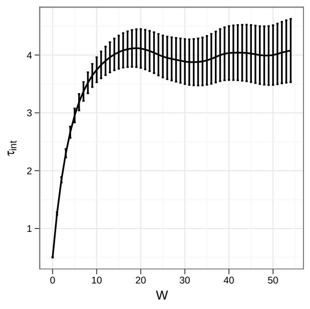


(d) Autocorrelation for θ_η

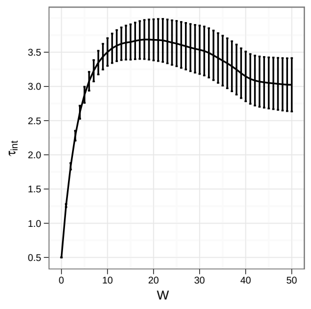


(e) Autocorrelation for σ^2

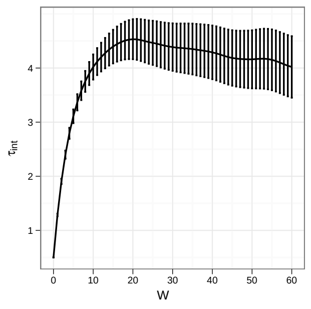
Figure A.43: MCMC Autocorrelation for DTAPs with $SARSA(0)$ and ϵ-greedy Policy with $\delta_V = 1.2, \delta_E = 1.2$



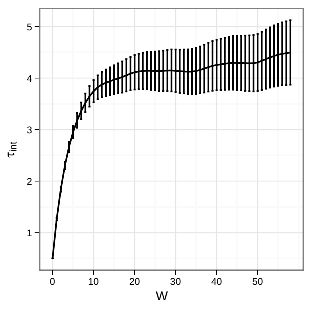
(a) τ_{int} vs. W for β_0



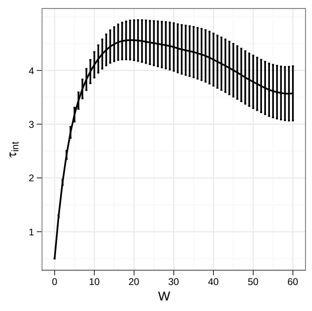
(b) τ_{int} vs. W for θ_α



(c) τ_{int} vs. W for θ_λ



(d) τ_{int} vs. W for θ_η



(e) τ_{int} vs. W for σ^2

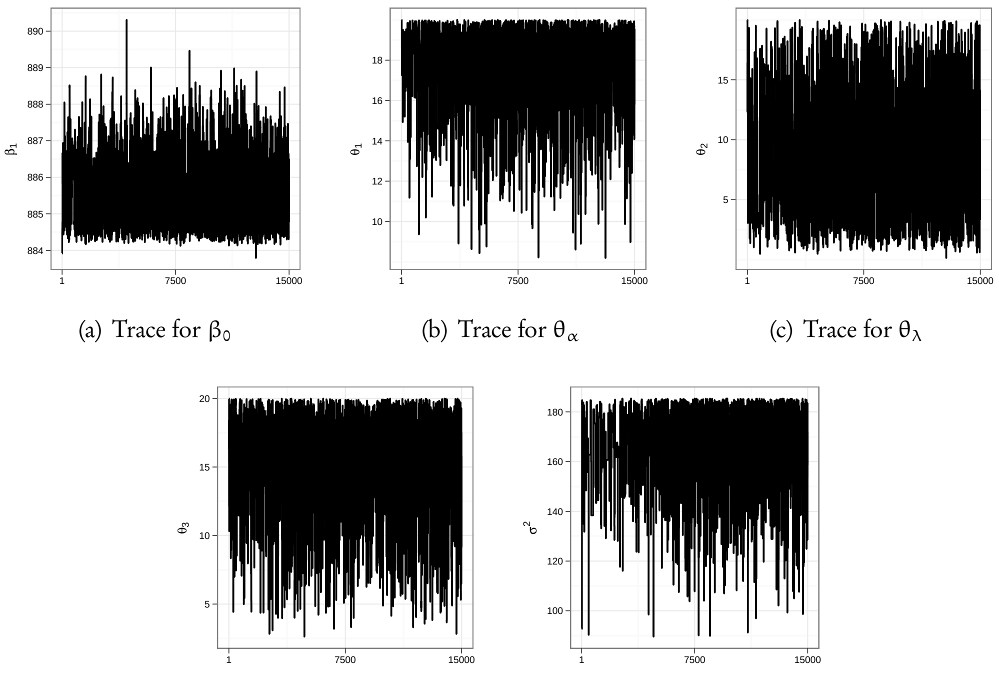
Figure A.44: MCMC τ_{int} vs. W for ER DTAPs with $SARSA(0)$ and ϵ-greedy Policy with $\delta_V = 1.2, \delta_E = 1.2$

A.2.2.2 $\delta_V = 1.8, \delta_E = 1.8$

The summary of the Kriging estimation is given in Table A.10 with a MCMC chain size of 161. The Kriging model is assessed using 100 validation locations selected using a space-filling LHS design.

Table A.10: Kriging Summary of ER DTAPs with SARSA(0) and ϵ-greedy Policy with $\delta_V = 1.8, \delta_E = 1.8$

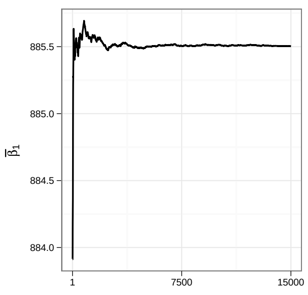
Parameter	Value	95% CI half-width
β_0	885.54	0.161
θ_α	17.88	0.334
θ_λ	8.12	0.841
θ_η	15.05	0.65
σ^2	166.74	2.554
R^2	0.999	
RMSE	1.79	



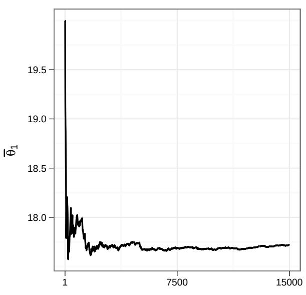
(a) Trace for β_0 (b) Trace for θ_α (c) Trace for θ_λ

(d) Trace for θ_η (e) Trace for σ^2

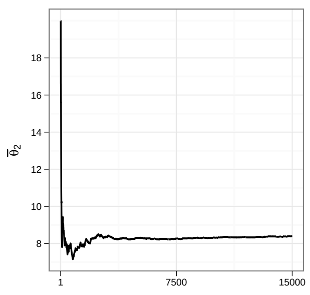
Figure A.45: MCMC Trace for ER DTAPs with SARSA(0) and ϵ-greedy Policy with $\delta_V = 1.8, \delta_E = 1.8$



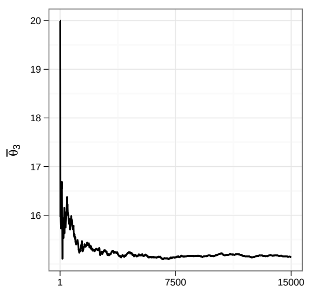
(a) Equilibration for β_0



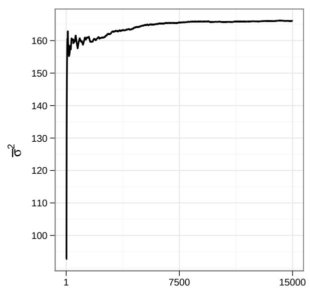
(b) Equilibration for θ_α



(c) Equilibration for θ_λ

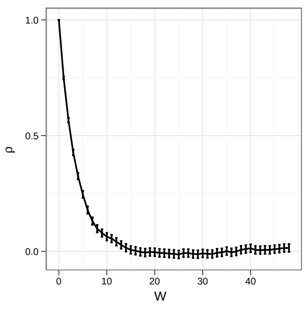


(d) Equilibration for θ_η

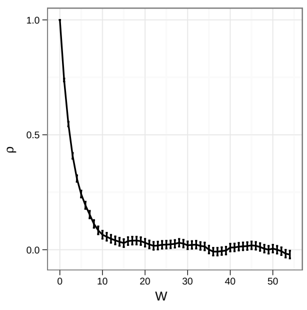


(e) Equilibration for σ^2

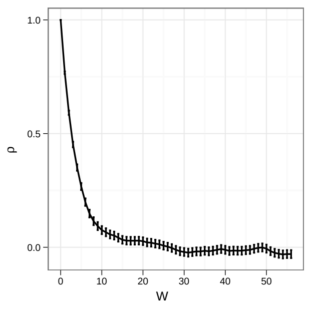
Figure A.46: MCMC Mean for ER DTAPs with SARSA(0) and ϵ-greedy Policy with $\delta_V = 1.8, \delta_E = 1.8$



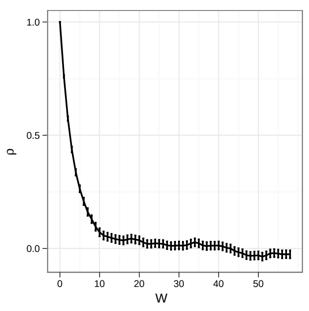
(a) Autocorrelation for β_0



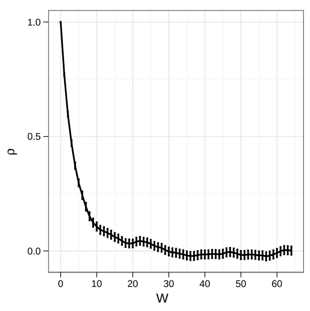
(b) Autocorrelation for θ_α



(c) Autocorrelation for θ_λ

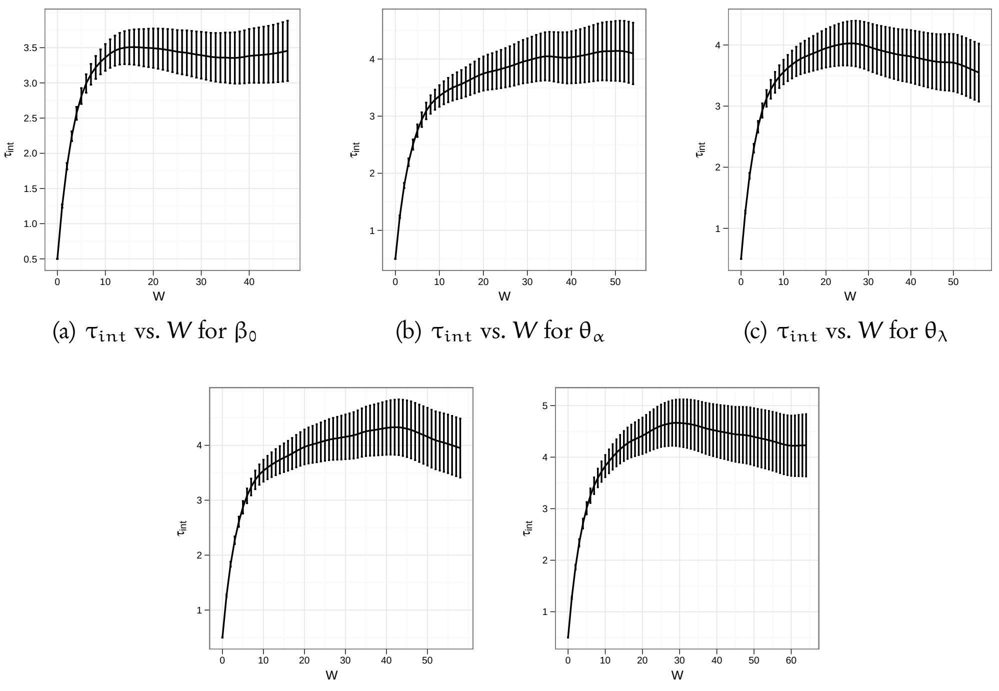


(d) Autocorrelation for θ_η



(e) Autocorrelation for σ^2

Figure A.47: MCMC Autocorrelation for DTAPs with SARSA(0) and ϵ-greedy Policy with $\delta_V = 1.8, \delta_E = 1.8$



(a) τ_{int} vs. W for β_0 (b) τ_{int} vs. W for θ_α (c) τ_{int} vs. W for θ_λ

(d) τ_{int} vs. W for θ_η (e) τ_{int} vs. W for σ^2

Figure A.48: MCMC τ_{int} vs. W for ER DTAPs with $\mathsf{SARSA}(0)$ and ϵ-greedy Policy with $\delta_V = 1.8, \delta_E = 1.8$

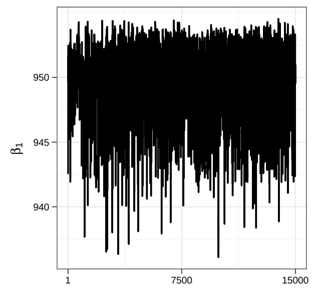
A.2.3 $\mathsf{SARSA}(0)$ and WPL Policy

A.2.3.1 $\delta_V = 1.2, \delta_E = 1.2$

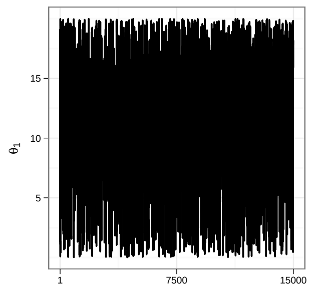
The summary of the Kriging estimation is given in Table A.11 with a MCMC chain size of 161. The Kriging model is assessed using 100 validation locations selected using a space-filling LHS design.

Table A.11: Kriging Summary of ER DTAPs with $\mathsf{SARSA}(0)$ and WPL Policy with $\delta_V = 1.8, \delta_E = 1.8$

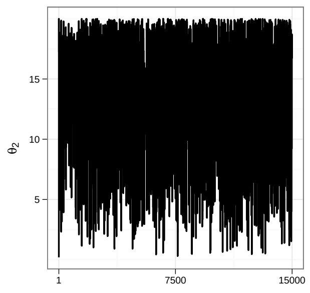
Parameter	Value	95% CI half-width
β_0	949.91	0.38
θ_α	11.62	0.83
θ_λ	13.83	0.7
θ_η	14.67	0.6
θ_ζ	9.08	0.85
σ^2	287.41	15.26
R^2	0.999	
RMSE	5.23	



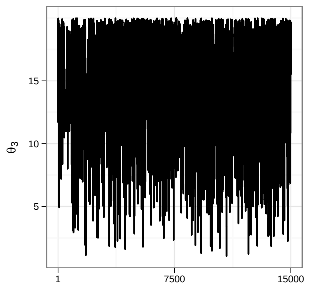
(a) Trace for β_0



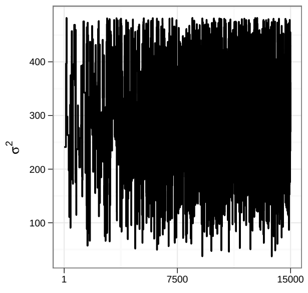
(b) Trace for θ_α



(c) Trace for θ_λ

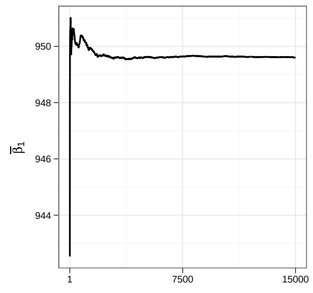


(d) Trace for θ_η

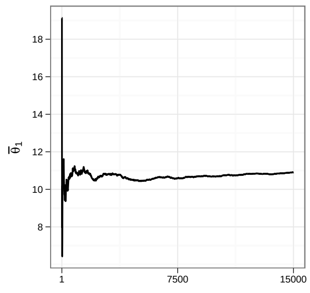


(e) Trace for σ^2

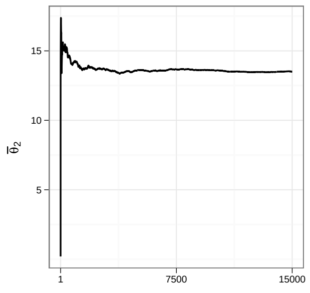
Figure A.49: MCMC Trace for DTAPs with $\mathsf{SARSA(0)}$ and WPL Policy with $\delta_V = 1.8, \delta_E = 1.8$



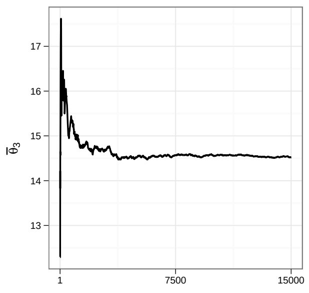
(a) Equilibration for β_0



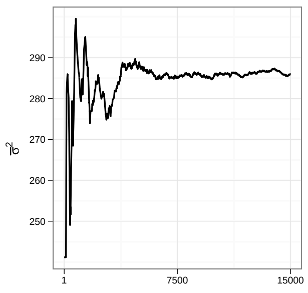
(b) Equilibration for θ_α



(c) Equilibration for θ_λ

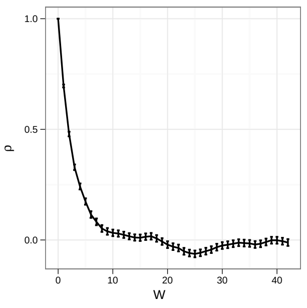


(d) Equilibration for θ_η

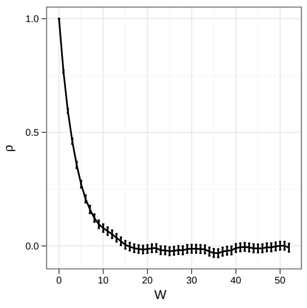


(e) Equilibration for σ^2

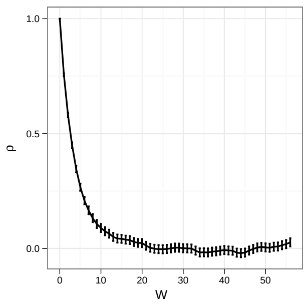
Figure A.50: MCMC Mean for DTAPs with $\mathsf{SARSA(0)}$ and WPL Policy with $\delta_V = 1.8, \delta_E = 1.8$



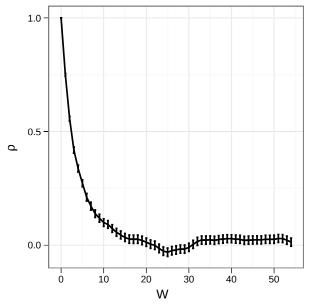
(a) Autocorrelation for β_0



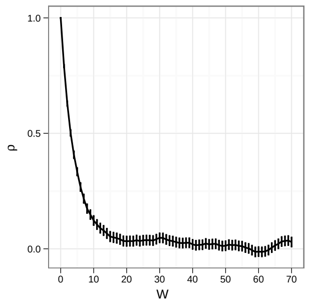
(b) Autocorrelation for θ_α



(c) Autocorrelation for θ_λ

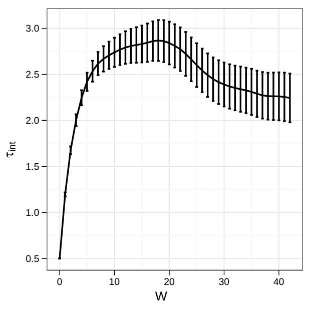


(d) Autocorrelation for θ_η

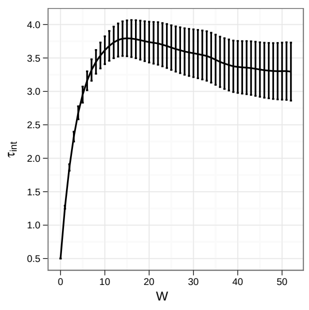


(e) Autocorrelation for σ^2

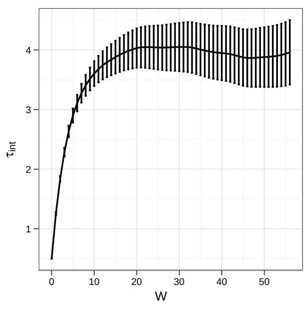
Figure A.51: MCMC Autocorrelation for DTAPs with $SARSA(0)$ and WPL Policy with $\delta_V = 1.8, \delta_E = 1.8$



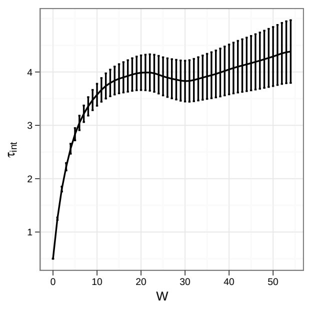
(a) τ_{int} vs. W for β_0



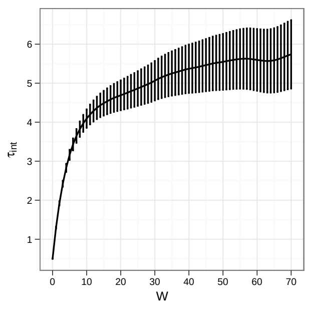
(b) τ_{int} vs. W for θ_α



(c) τ_{int} vs. W for θ_λ



(d) τ_{int} vs. W for θ_η



(e) τ_{int} vs. W for σ^2

Figure A.52: MCMC τ_{int} vs. W for DTAPs with $SARSA(0)$ and WPL Policy with $\delta_V = 1.8, \delta_E = 1.8$

Bibliography

[1] Sherief Abdallah and Victor Lesser. Multiagent reinforcement learning and self-organization in a network of agents. In *AAMAS '07: Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, pages 1–8, New York, NY, USA, 2007. ACM. ISBN 978-81-904262-7-5.

[2] Sherief Abdallah and Victor Lesser. Learning the task allocation game. In *AAMAS '06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 850–857, New York, NY, USA, 2006. ACM. ISBN 1-59593-303-4.

[3] O. Abul, F. Polat, and R. Alhajj. Multiagent reinforcement learning using function approximation. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 30(4):485–497, 2000.

[4] M. N. Ahmadabadi and M. Asadpour. Expertness based cooperative q-learning. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 32(1):66–76, 2002.

[5] S. E. Ahnert, D. Garlaschelli, T. M. A. Fink, and G. Caldarelli. Ensemble approach to the analysis of weighted networks. *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)*, 76(1), 2007.

[6] S. E. Ahnert, D. Garlaschelli, T. M. A. Fink, and G. Caldarelli. Applying weighted network measures to microarray distance matrices. *Journal of Physics A: Mathematical and Theoretical*, 41(22):224011+, 2008.

[7] Bruce Ankenman, Barry L. Nelson, and Jeremy Staum. Stochastic kriging for simulation metamodeling. In *2008 Winter Simulation Conference (WSC)*, pages 362–370. IEEE, December 2008. ISBN 978-1-4244-2707-9.

[8] Kenneth J. Arrow, A. K. Sen, and Kotaro Suzumura, editors. *Handbook of Social Choice and Welfare, Volume 1 (Handbooks in Economics)*. North Holland, 1 edition, August 2002. ISBN 0444829148.

[9] W. Brian Arthur. Inductive reasoning and bounded rationality. *The American Economic Review*, 84(2):406–411, 1994. ISSN 00028282.

[10] John Asmuth, Lihong Li, Michael L. Littman, Ali Nouri, and David Wingate. A bayesian sampling approach to exploration in reinforcement learning. In *Proceedings of The 25th Conference on Uncertainty in Artificial Intelligence (UAI-09)*, June 2009.

[11] Robert Axelrod and William D. Hamilton. The evolution of cooperation. *Science*, 211(4489): 1390–1396, 1981. ISSN 00368075.

[12] Babaoğlu, H. Meling, and A. Montresor. Anthill: a framework for the development of agent-based peer-to-peer systems. In *22nd International Conference on Distributed Computing Systems*, pages 15–22. IEEE Comput. Soc, July 2002. ISBN 0-7695-1585-1.

[13] B. Baddeley. Reinforcement learning in continuous time and space: Interference and not ill conditioning is the main problem when using distributed function approximators. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 38(4):950–956, 2008.

[14] Sudipto Banerjee, Bradley, and Alan E. Gelfand. *Hierarchical Modeling and Analysis for Spatial Data (Monographs on Statistics and Applied Probability).* Chapman & Hall/CRC, 1 edition, December 2003. ISBN 158488410X.

[15] Albert-Lászlό Barabási and Réka Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, October 1999.

[16] Alain Barrat, Marc Barthélemy, and Alessandro Vespignani. Weighted evolving networks: Coupling topology and weight dynamics. *Physical Review Letters*, 92(22), 2004.

[17] Alain Barrat, Marc Barthélemy, and Alessandro Vespignani. Modeling the evolution of weighted networks. *Physical Review E*, 70(6):066149+, December 2004.

[18] Jordi Bascompte. Disentangling the web of life. *Science*, 325(5939):416–419, July 2009.

[19] S. Battiston. Inner structure of capital control networks. *Physica A: Statistical Mechanics and its Applications*, 338(1-2):107–112, July 2004. ISSN 03784371.

[20] Daniel S. Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. The complexity of decentralized control of markov decision processes. *Mathematics of Operations Research*, 27 (4):819–840, 2002. ISSN 0364765X.

[21] Gunter Bolch, Stefan Greiner, Hermann de Meer, and Kishor S. Trivedi. *Queueing Networks and Markov Chains: Modeling and Performance Evaluation with Computer Science Applications.* WileyBlackwell, 2nd edition edition, May 2006. ISBN 0471565253.

[22] Eric Bonabeau, Marco Dorigo, and Guy Theraulaz. *Swarm Intelligence: From Natural to Artificial Systems (Santa Fe Institute Studies in the Sciences of Complexity Proceedings).* Oxford University Press, USA, 1 edition, September 1999. ISBN 0195131592.

[23] Michael Bowling. Convergence and no-regret in multiagent learning. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 209–216. MIT Press, Cambridge, MA, 2005.

[24] George E. P. Box and Norman R. Draper. *Response Surfaces, Mixtures, and Ridge Analyses (Wiley Series in Probability and Statistics).* Wiley-Interscience, 2 edition, March 2007. ISBN 0470053577.

[25] Michael Burl and Esther Wang. Active learning for directed exploration of complex systems. In Léon Bottou and Michael Littman, editors, *Proceedings of the 26th International Conference on Machine Learning*, pages 89–96, Montreal, June 2009. Omnipress.

[26] L. Busoniu, R. Babuska, and B. De Schutter. A comprehensive survey of multiagent reinforcement learning. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 38(2):156–172, 2008.

[27] Carter T. Butts. Revisiting the foundations of network analysis. *Science*, 325(5939):414–416, July 2009. ISSN 1095-9203.

[28] Rich Caruana and Alexandru N. Mizil. An empirical comparison of supervised learning algorithms. In *ICML '06: Proceedings of the 23rd international conference on Machine learning*, pages 161–168, New York, NY, USA, 2006. ACM. ISBN 1-59593-383-2.

[29] Georgios Chalkiadakis and Craig Boutilier. Coordination in multiagent reinforcement learning: a bayesian approach. In *AAMAS '03: Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 709–716, New York, NY, USA, 2003. ACM Press. ISBN 1581136838.

[30] Georgios Chalkiadakis and Craig Boutilier. Bayesian reinforcement learning for coalition formation under uncertainty. In *AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1090–1097, Washington, DC, USA, 2004. IEEE Computer Society. ISBN 1581138644.

[31] Yann Chevaleyre, Paul E. Dunne, Ulle Endriss, Jérôme Lang, Michel Lemaître, Nicolas Maudet, Julian Padget, Steve Phelps, Juan A. Rodríguez-aguilar, and Paulo Sousa. Issues in multiagent resource allocation. *Informatica*, 30, 2006.

[32] Ka-Po Chow and Yu-Kwong Kwok. On load balancing for distributed multiagent computing. *Parallel and Distributed Systems, IEEE Transactions on*, 13(8):787–801, November 2002.

[33] Caroline Claus and Craig Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. In *AAAI '98/IAAI '98: Proceedings of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence*, pages 746–752, Menlo Park, CA, USA, 1998. American Association for Artificial Intelligence. ISBN 0-262-51098-7.

[34] Aaron Clauset, Cosma R. Shalizi, and M. E. J. Newman. Power-law distributions in empirical data. *SIAM Review*, 51(4):661+, Feb 2009. ISSN 0036-1445.

[35] David A. Cohn, Zoubin Ghahramani, and Michael I. Jordan. Active learning with statistical models. *Journal of Artificial Intelligence Research*, 4:129–145, 1995.

[36] V. Colizza, A. Barrat, M. Barthélemy, and A. Vespignani. The role of the airline transportation network in the prediction and predictability of global epidemics. *Proceedings of the National Academy of Sciences of the United States of America*, 103(7):2015–2020, February 2006. ISSN 0027-8424.

[37] Rémi Coulom. Feedforward neural networks in reinforcement learning applied to high-dimensional motor control. In *Algorithmic Learning Theory - 13th International Conference, ALT 2002 Lübeck, Germany, November 24–26, 2002 Proceedings*, pages 403–414. Springer Berlin / Heidelberg, November 2009.

[38] Noel Cressie. The origins of kriging. *Mathematical Geology*, 22(3):239–252, April 1990.

[39] Noel A. C. Cressie. *Statistics for Spatial Data (Wiley Series in Probability and Statistics)*. Wiley-Interscience, rev sub edition, January 1993. ISBN 0471002550.

[40] Carla Currin, Toby Mitchell, Max Morris, and Don Ylvisaker. Bayesian prediction of deterministic functions, with applications to the design and analysis of computer experiments. *Journal of the American Statistical Association*, 86(416):953–963, 1991. ISSN 01621459.

[41] Dominik Dahlem and William Harrison. Waiting time sensitivities of social and random graph models. *Social Network Analysis and Mining, International Conference on Advances in*, 0: 176–181, July 2009.

[42] Dominik Dahlem and William Harrison. Globally optimal multi-agent reinforcement learning parameters in distributed task assignment. *Web Intelligence and Intelligent Agent Technology, IEEE/WIC/ACM International Conference on*, 2:28–35, 2009.

[43] Dominik Dahlem, David McKitterick, Lotte Nickel, Jim Dowling, Bartek Biskupski, and René Meier. Binding- and port-agnostic service composition using a p2p soa. In Kunal Verma, Amit Sheth, Michal Zaremba, and Christoph Bussler, editors, *International Workshop on Dynamic Web Processes DWP 2005, at ICSOC 2005*, pages 61–72, P.O. Box 218, Yorktown Heights, NY 10598 USA, December 2005. IBM T.J. Watson Research Center.

[44] Dominik Dahlem, Lotte Nickel, Jan Sacha, Bartosz Biskupski, Jim Dowling, and René Meier. Towards improving the availability of service compositions. In *Digital EcoSystems and Technologies Conference, 2007. DEST '07. Inaugural IEEE-IES*, pages 67–70, June 2007.

[45] Elizabeth M. Daly and Mads Haahr. Social network analysis for information flow in disconnected delay-tolerant manets. *IEEE Transactions on Mobile Computing*, 8(5):606–621, May 2009. ISSN 1536-1233.

[46] Gianni Di Caro and Marco Dorigo. Antnet: Distributed stigmergetic control for communications networks. *Journal of Artificial Intelligence Research*, 9:317–365, December 1998.

[47] M. Dorigo, G. D. Di Caro, and L. Gambardella. Ant colony optimization: A new meta-heuristic. In Peter J. Angeline, Zbyszek Michalewicz, Marc Schoenauer, Xin Yao, and Ali Zalzala, editors, *Proceedings of the Congress on Evolutionary Computation*, volume 2, pages 1470–1477, Mayflower Hotel, Washington D.C., USA, June-September 1999. IEEE Press.

[48] Marco Dorigo and Luca M. Gambardella. Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1(1):53–66, April 1997.

[49] J. Dowling, E. Curran, R. Cunningham, and V. Cahill. Using feedback in collaborative reinforcement learning to adaptively optimize manet routing. *Systems, Man and Cybernetics, Part A, IEEE Transactions on*, 35(3):360–372, 2005.

[50] John Duchi, Shai S. Shwartz, Yoram Singer, and Tushar Chandra. Efficient projections onto the ℓ_1-ball for learning in high dimensions. In *ICML '08: Proceedings of the 25th international conference on Machine learning*, pages 272–279, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-205-4.

[51] Pál Erdős and Alfréd Rényi. On the evolution of random graphs. *Publications of the Mathematical Institute of the Hungarian Academy of Sciences*, 5:17–61, 1960.

[52] Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi. *Reasoning About Knowledge*. The MIT Press, 1st mit press paperback ed edition, December 2003. ISBN 0262562006.

[53] Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. On power-law relationships of the internet topology. In *SIGCOMM '99: Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication*, pages 251–262, New York, New York, USA, 1999. ACM Press. ISBN 1581131356.

[54] Andrew O. Finley, Sudipto Banerjee, and Bradley P. Carlin. spbayes: An r package for univariate and multivariate hierarchical point-referenced spatial models, April 2007.

[55] Ian Foster, Yong Zhao, Ioan Raicu, and Shiyong Lu. Cloud computing and grid computing 360-degree compared. In *2008 Grid Computing Environments Workshop*, pages 1–10. IEEE, November 2008. ISBN 978-1-4244-2860-1.

[56] Linton C. Freeman. *The Development of Social Network Analysis: A Study in the Sociology of Science*. Empirical Press, July 2004. ISBN 1594577145.

[57] Wai-Tat Fu and John R. Anderson. From recurrent choice to skill learning: A reinforcement-learning model. *Journal of Experimental Psychology: General*, 135(2):184–206, May 2006.

[58] Aram Galstyan and Kristina Lerman. *Analysis of a Stochastic Model of Adaptive Task Allocation in Robots*, volume 3464 of *Lecture Notes in Computer Science*, pages 167–179. Springer Berlin / Heidelberg, Berlin / Heidelberg, May 2005.

[59] Shawn Gano, John Renaud, Jay Martin, and Timothy Simpson. Update strategies for kriging models used in variable fidelity optimization. *Structural and Multidisciplinary Optimization*, 32(4):287–298, October 2006.

[60] Marie Gaudard, Marvin Karson, Ernst Linder, and Debajyoti Sinha. Bayesian spatial prediction. *Environmental and Ecological Statistics*, 6(2):147–171, June 1999.

[61] Andrew Gelman, John B. Carlin, Hal S. Stern, Donald B. Rubin, and A. Gelman. *Bayesian Data Analysis*. Chapman & Hall/CRC, 1st edition, June 1995. ISBN 0412039915.

[62] David Ginsbourger, Delphine Dupuy, Anca Badea, Laurent Carraro, and Olivier Roustant. A note on the choice and the estimation of kriging models for the analysis of deterministic computer experiments. *Applied Stochastic Models in Business and Industry*, 25(2):115–131, 2009. ISSN 1526-4025.

[63] J. B. Glattfelder and S. Battiston. Backbone of complex networks of corporations: The flow of control. *Physical Review E*, 80(3), September 2009. ISSN 1550-2376.

[64] William L. Goffe, Gary D. Ferrier, and John Rogers. Global optimization of statistical functions with simulated annealing. *Journal of Econometrics*, 60:65–99, 1994.

[65] Eduardo R. Gomes and Ryszard Kowalczyk. Dynamic analysis of multiagent Q-learning with ϵ-greedy exploration. In Léon Bottou and Michael Littman, editors, *Proceedings of the 26th International Conference on Machine Learning*, pages 369–376, Montreal, 2009. Omnipress.

[66] Richard L. Graham and Rainer Keller. *Dynamic Communicators in MPI*, volume 5759, chapter 18, pages 116–123. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. ISBN 978-3-642-03769-6.

[67] Robert B. Gramacy and Herbert K. H. Lee. Adaptive design and analysis of supercomputer experiments, 2009.

[68] Robert B. Gramacy, Herbert K. H. Lee, and William G. Macready. Parameter space exploration with gaussian process trees. In *ICML '04: Proceedings of the twenty-first international conference on Machine learning*, New York, NY, USA, 2004. ACM. ISBN 1581138285.

[69] Garrett Hardin. The tragedy of the commons. *Science*, 162(3859):1243–1248, December 1968.

[70] Kenneth A. Hawick, P. D. Coddington, and H. A. James. Distributed frameworks and parallel algorithms for processing large-scale geographic data. *Parallel Computing*, 29(10):1297–1333, October 2003. ISSN 01678191.

[71] Pieter J. Hoen, Karl Tuyls, Liviu Panait, Sean Luke, and J. A. La Poutré. *An Overview of Cooperative and Competitive Multiagent Learning*. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, March 2006.

[72] Roger W. Hoerl. Ridge analysis 25 years later. *The American Statistician*, 39(3):186–192, 1985. ISSN 00031305.

[73] Ronald A. Howard. *Dynamic Programming and Markov Process (Technology Press Research Monographs)*. The MIT Press, first edition edition, June 1960. ISBN 0262080095.

[74] Junling Hu and Michael P. Wellman. Nash q-learning for general-sum stochastic games. *J. Mach. Learn. Res.*, 4:1039–1069, 2003. ISSN 1533-7928.

[75] Tommi Jaakkola, Michael I. Jordan, and Satinder P. Singh. On the convergence of stochastic iterative dynamic programming algorithms. *Neural Comput.*, 6(6):1185–1201, 1994. ISSN 0899-7667.

[76] James R. Jackson. Jobshop-like queueing systems. *Manage. Sci.*, 50(12 Supplement):1796–1802, 2004. ISSN 0025-1909.

[77] T. Kamada and S. Kawai. An algorithm for drawing general undirected graphs. *Inf. Process. Lett.*, 31(1):7–15, April 1989. ISSN 0020-0190.

[78] Spiros Kapetanakis and Daniel Kudenko. Reinforcement learning of coordination in cooperative multi-agent systems. In *Eighteenth national conference on Artificial intelligence*, pages 326–331, Menlo Park, CA, USA, 2002. American Association for Artificial Intelligence. ISBN 0-262-51129-0.

[79] K. Kerry and K. Hawick. Kriging interpolation on high-performance computers. In *Proceedings of the International Conference and Exhibition on High-Performance Computing and Networking*, pages 429–438. Springer Berlin / Heidelberg, April 1998.

[80] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, May 1983. ISSN 1095-9203.

[81] J. Kleijnen. Kriging metamodeling in simulation: A review. *European Journal of Operational Research*, 192(3):707–716, February 2009. ISSN 03772217.

[82] Kyriakos Kritikos and Dimitris Plexousakis. Mixed-integer programming for qos-based web service matchmaking. *Services Computing, IEEE Transactions on*, 2(2):122–139, 2009. ISSN 1939-1374.

[83] Takayasu Kumano, Shinkyu Jeong, Shigeru Obayashi, Yasushi Ito, Keita Hatanaka, and Hiroyuki Morino. Multidisciplinary design optimization of wing shape for a small jet aircraft using kriging model. In *44th AIAA Aerospace Sciences Meeting and Exhibit*. The American Institute of Aeronautics and Astronautics (AIAA), January 2006.

[84] Michail G. Lagoudakis and Ronald Parr. Least-squares policy iteration. *J. Mach. Learn. Res.*, 4: 1107–1149, 2003. ISSN 1533-7928.

[85] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, July 1982. ISSN 0164-0925.

[86] Averill M. Law and David W. Kelton. *Simulation Modelling and Analysis*. McGraw-Hill Education - Europe, April 2000. ISBN 0071165371.

[87] Lihong Li, Michael L. Littman, and Christopher R. Mansley. Online exploration in least-squares policy iteration. In *Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009)*, May 2009.

[88] Asher Lipson. Empirically evaluating multiagent reinforcement learning algorithms in repeated games. Master's thesis, University of British Columbia, November 2005.

[89] Michael L. Littman. Friend-or-foe q-learning in general-sum games. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 322–328. Morgan Kaufmann, 2001.

[90] Michael L. Littman and Csaba Szepesvári. A generalized reinforcement-learning model: Convergence and applications. Technical report, Brown University, Providence, RI, USA, 1996.

[91] P. J. Macdonald, E. Almaas, and A. L. Barabási. Minimum spanning trees on weighted scale-free networks, May 2004.

[92] Hamid Maei, Csaba Szepesvari, Shalabh Bhatnagar, Doina Precup, David Silver, and Rich Sutton. Convergent temporal-difference learning with arbitrary smooth function approximation. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 1204–1212. 2009.

[93] S. Mannor and J. Shamma. Multi-agent learning for engineers. *Artificial Intelligence*, 171(7): 417–422, May 2007. ISSN 00043702.

[94] Jay D. Martin. *A methodology for evaluating system-level uncertainty in the conceptual design of complex multidisciplinary systems*. PhD thesis, Dept. of Mechanical Engineering, The Pennsylvania State University, May 2005.

[95] Jay D. Martin and Timothy W. Simpson. Use of kriging models to approximate deterministic computer models. *AIAA Journal*, 43(4):853–863, 2005.

[96] Maja J. Matarić. Reinforcement learning in the multi-robot domain. *Autonomous Robots*, 4(1): 73–83, March 1997. ISSN 09295593.

[97] Michael D. Mckay. Latin hypercube sampling as a tool in uncertainty analysis of computer models. In *WSC '92: Proceedings of the 24th conference on Winter simulation*, pages 557–564, New York, NY, USA, 1992. ACM Press. ISBN 0780307984.

[98] J. I. L. Miguéns and J. F. F. Mendes. *Weighted and Directed Network on Traveling Patterns*, volume 5151 of *Lecture Notes in Computer Science*, chapter 13, pages 145–154. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. ISBN 978-3-540-92190-5.

[99] Stanley Milgram. The small-world problem. *Psychology Today*, 1(1):60–67, 1967.

[100] Melanie Mitchell. Complex systems: Network thinking. *Artificial Intelligence*, 170(18): 1194–1212, December 2006.

[101] Toby J. Mitchell and Max D. Morris. Bayesian design and analysis of computer experiments: Two examples. *Statistica Sinica*, 2(2):359–379, July 1992. ISSN 1017-0405.

[102] Douglas C. Montgomery. *Design and Analysis of Experiments*. Wiley, December 2004. ISBN 047148735X.

[103] Alberto Montresor, Hein Meling, and Özalp Babaoğlu. Messor: Load-balancing through a swarm of autonomous agents. In *In Proceedings of 1st Workshop on Agent and Peer-to-Peer Systems*, pages 125–137, 2002.

[104] Oskar Morgenstern and John Von Neumann. *Theory of Games and Economic Behavior*. Princeton University Press, May 1944. ISBN 0691003629.

[105] Herve Moulin. *Axioms of Cooperative Decision Making (Econometric Society Monographs)*. Cambridge University Press, July 1991. ISBN 0521424585.

[106] Rana Moyeed and Andreas Papritz. An empirical comparison of kriging methods for nonlinear spatial point prediction. *Mathematical Geology*, 34(4):365–386, May 2002.

[107] Raymond H. Myers, Douglas C. Montgomery, and Christine M. Anderson-Cook. *Response Surface Methodology: Process and Product Optimization Using Designed Experiments (Wiley Series in Probability and Statistics)*. Wiley, 3 edition, January 2009. ISBN 0470174463.

[108] Radford M. Neal. Monte carlo implementation of gaussian process models for bayesian regression and classification, Jan 1997.

[109] M. E. J. Newman. Scientific collaboration networks. ii. shortest paths, weighted networks, and centrality. *Physical Review E*, 64(1):016132+, Jun 2001.

[110] M. E. J. Newman. Mixing patterns in networks. *Physical Review E*, 67(2):026126+, Feb 2003.

[111] M. E. J. Newman. Power laws, pareto distributions and zipf's law. *Contemporary Physics*, 46 (5):323–351, May 2006. ISSN 0010-7514.

[112] Ali Nouri and Michael L. Littman. Multi-resolution exploration in continuous spaces. In Daphne Koller, Dale Schuurmans, Yoshua Bengio, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 21, Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 8-11, 2008*, pages 1209–1216. MIT Press, December 2008.

[113] M. A. Oliver and R. Webster. Kriging: a method of interpolation for geographical information systems. *International Journal of Geographical Information Science*, 4(3):313–332, 1990.

[114] Tore Opsahl and Pietro Panzarasa. Clustering in weighted networks. *Social Networks*, 31(2): 155–163, May 2009. ISSN 03788733.

[115] Andy Oram. *Peer-to-Peer : Harnessing the Power of Disruptive Technologies*. O'Reilly, March 2001. ISBN 059600110X.

[116] Elinor Ostrom. A general framework for analyzing sustainability of social-ecological systems. *Science*, 325(5939):419–422, July 2009.

[117] Liviu Panait and Sean Luke. Cooperative multi-agent learning: The state of the art. *Autonomous Agents and Multi-Agent Systems*, 11(3):387–434, November 2005. ISSN 1387-2532.

[118] H. T. Papadopoulos and C. Heavey. Queueing theory in manufacturing systems analysis and design: A classification of models for production and transfer lines. *European Journal of Operational Research*, 92(1):1–27, July 1996. ISSN 03772217.

[119] L. Peshkin and V. Savova. Reinforcement learning for adaptive routing. In *2002 International Joint Conference on Neural Networks (IJCNN)*, volume 2, pages 1825–1830. IEEE, 2002. ISBN 0-7803-7278-6.

[120] Carl E. Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, December 2005. ISBN 026218253X.

[121] Christian P. Robert and George Casella. *Monte Carlo Statistical Methods (Springer Texts in Statistics)*. Springer, July 2005. ISBN 0387212396.

[122] Gareth O. Roberts and Jeffrey S. Rosenthal. Examples of adaptive mcmc. *Journal of Computational and Graphical Statistics*, 18(2):349–367, June 2009.

[123] Raul Rojas. *Neural Networks: A Systematic Introduction*. Springer, 1 edition, July 1996. ISBN 3540605053.

[124] Jeffrey S. Rosenthal. Amcmc: An r interface for adaptive mcmc. *Computational Statistics & Data Analysis*, 51(12):5467–5470, 2007.

[125] Sheldon M. Ross. *Simulation*. Academic Press, 4 edition, August 2006. ISBN 0125980639.

[126] R. Rudek, L. Koszalka, and I. P. Koszalka. Introduction to multi-agent modified q-learning routing for computer networks. In *AICT-SAPIR-ELETE '05: Proceedings of the Advanced Industrial Conference on Telecommunications/Service Assurance with Partial and Intermittent Resources Conference/E-Learning on Telecommunications Workshop*, pages 408–413, Washington, DC, USA, 2005. IEEE Computer Society. ISBN 0-7695-2388-9.

[127] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. *Learning internal representations by error propagation*, chapter 8, pages 318–362. MIT Press, Cambridge, MA, USA, 1986. ISBN 0-262-68053-X.

[128] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: Modern Approach*. Prentice Hall, 1st edition, January 1995. ISBN 0131038052.

[129] Jan Sacha, Bartosz Biskupski, Dominik Dahlem, Raymond Cunningham, Jim Dowling, and René Meier. A service-oriented peer-to-peer architecture for a digital ecosystem. In *Digital EcoSystems and Technologies Conference, 2007. DEST '07. Inaugural IEEE-IES*, pages 205–210, June 2007.

[130] Jan Sacha, Bartosz Biskupski, Dominik Dahlem, Raymond Cunningham, René Meier, Jim Dowling, and Mads Haahr. Decentralising a service-oriented architecture. *Peer-to-Peer Networking and Applications*, October 2009. ISSN 1936-6450.

[131] Jerome Sacks, Susannah B. Schiller, and William J. Welch. Designs for computer experiments. *Technometrics*, 31(1):41–47, 1989. ISSN 00401706.

[132] A. Sadek and N. Basha. Self-learning intelligent agents for dynamic traffic routing on transportation networks. In *Proceedings of the 6th International Conference on Complex Systems (ICCS), June 25-30, 2006; Boston, MA*, June 2006.

[133] Jari Saramäki, Mikko Kivelä, Jukka P. Onnela, Kimmo Kaski, and János Kertész. Generalizations of the clustering coefficient to weighted complex networks. *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)*, 75(2):027105+, 2007.

[134] Andrea Schaerf, Yoav Shoham, and Moshe Tennenholtz. Adaptive load balancing: a study in multi-agent learning. *J. Artif. Int. Res.*, 2(1):475–500, 1994. ISSN 1076-9757.

[135] Frank Schweitzer, Giorgio Fagiolo, Didier Sornette, Fernando Vega-Redondo, Alessandro Vespignani, and Douglas R. White. Economic networks: the new challenges. *Science (New York, N.Y.)*, 325(5939):422–425, July 2009. ISSN 1095-9203.

[136] Niranjan G. Shivaratri, Phillip Krueger, and Mukesh Singhal. Load distributing for locally distributed systems. *Computer*, 25(12):33–44, 1992. ISSN 0018-9162.

[137] Y. Shoham, R. Powers, and T. Grenager. If multi-agent learning is the answer, what is the question? *Artificial Intelligence*, 171(7):365–377, May 2007. ISSN 00043702.

[138] Yoav Shoham and Kevin Leyton-Brown. *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press, December 2008. ISBN 0521899435.

[139] Wasserman Stanley and Faust Katherine. *Social Network Analysis: Methods and Applications (Structural Analysis in the Social Sciences)*. Cambridge University Press, 1 edition, November 1994. ISBN 0521387078.

[140] Jeremy Staum. Better simulation metamodeling: The why, what, and how of stochastic kriging. In *Simulation Conference, 2009. WSC 2009. Winter*, December 2009.

[141] Michael L. Stein. *Interpolation of Spatial Data: Some Theory for Kriging (Springer Series in Statistics)*. Springer, 1 edition, June 1999. ISBN 0387986294.

[142] E. Stinstra and D. Denhertog. Robust optimization using computer experiments. *European Journal of Operational Research*, 191(3):816–837, December 2008. ISSN 03772217.

[143] Peter Stone and Manuela Veloso. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, 8(3):345–383, June 2000. ISSN 09295593.

[144] R. S. Sutton, A. G. Barto, and R. J. Williams. Reinforcement learning is direct adaptive optimal control. *Control Systems Magazine, IEEE*, 12(2):19–22, 1992.

[145] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning)*. The MIT Press, March 1998. ISBN 0262193981.

[146] Bosiljka Tadić, G. J. Rodgers, and Stefan Thurner. Transport on complex networks: Flow, jamming and optimization, Jul 2006.

[147] Milind Tambe, Jafar Adibi, Yaser Al-Onaizan, Ali Erdem, Gal A. Kaminka, Stacy C. Marsella, and Ion Muslea. Building agent teams using an explicit teamwork model and learning. *Artificial Intelligence*, 110(2):215–239, June 1999. ISSN 0004-3702.

[148] G. Tesauro, N. K. Jong, R. Das, and M. N. Bennani. A hybrid reinforcement learning approach to autonomic resource allocation. In *Proceedings of the IEEE International Conference on Autonomic Computing, 2006. ICAC '06*, pages 65–73, 2006.

[149] Gerald Tesauro. Td-gammon, a self-teaching backgammon program, achieves master-level play. *Neural Computation*, 6(2):215–219, March 1994. ISSN 0899-7667.

[150] Gerald Tesauro. Online resource allocation using decompositional reinforcement learning. In *AAAI'05: Proceedings of the 20th national conference on Artificial intelligence*, pages 886–891. AAAI Press, 2005. ISBN 1-57735-236-x.

[151] Gerald Tesauro, Nicholas Jong, Rajarshi Das, and Mohamed Bennani. Improvement of systems management policies using hybrid reinforcement learning. In *Machine Learning: ECML 2006 - 17th European Conference on Machine Learning, Berlin, Germany, September 18-22, 2006, Proceedings*, pages 783–791, 2006.

[152] Gerald Tesauro, Nicholas Jong, Rajarshi Das, and Mohamed Bennani. On the use of hybrid reinforcement learning for autonomic resource allocation. *Cluster Computing*, 10(3):287–299, 2007.

[153] Sebastian Thrun. The role of exploration in learning control. In *Handbook for Intelligent Control: Neural, Fuzzy and Adaptive Approaches*. Van Nostrand Reinhold, Florence, Kentucky, 1992.

[154] V. A. Traag and Jeroen Bruggeman. Community detection in networks with positive and negative links. *Physical Review E*, 80(3), September 2009. ISSN 1550-2376.

[155] John N. Tsitsiklis. Asynchronous stochastic approximation and q-learning. *Machine Learning*, 16(3):185–202, September 1994.

[156] Kagan Tumer and David Wolpert, editors. *Collectives and the Design of Complex Systems*. Springer, 1 edition, May 2004. ISBN 0387401652.

[157] Kagan Tumer and David Wolpert. Collective intelligence and braess' paradox. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 104–109. AAAI Press / The MIT Press, 2000. ISBN 0-262-51112-6.

[158] Karl Tuyls, Katja Verbeeck, and Tom Lenaerts. A selection-mutation model for q-learning in multi-agent systems. In *AAMAS '03: Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 693–700, New York, NY, USA, 2003. ACM. ISBN 1-58113-683-8.

[159] Wim C. M. van Beers. Kriging metamodeling in discrete-event simulation: an overview. In *WSC '05: Proceedings of the 37th conference on Winter simulation*, pages 202–208. Winter Simulation Conference, 2005. ISBN 0780395190.

[160] Wim C. M. van Beers and Jack P. C. Kleijnen. Customized sequential designs for random simulation experiments: Kriging metamodeling and bootstrapping. *European Journal of Operational Research*, 186(3):1099–1113, May 2008. ISSN 03772217.

[161] H. van den Herik, D. Hennes, M. Kaisers, K. Tuyls, and K. Verbeeck. Multi-agent learning dynamics: A survey. In *Cooperative Information Agents XI*, Lecture Notes in Computer Science, pages 36–56. Springer Berlin / Heidelberg, September 2007.

[162] N. Vandaele, T. V. Woensel, and A. Verbruggen. A queueing based traffic flow model. *Transportation Research Part D: Transport and Environment*, pages 121–135, March 2000. ISSN 1361-9209.

[163] Jay M. Ver Hoef, Noel Cressie, Robert N. Fisher, and Ted J. Case. *Uncertainty and Spatial Linear Models for Ecological Data*, chapter 10, pages 214–237. Springer, 1 edition, June 2001. ISBN 0387951296.

[164] Katja Verbeeck, Johan Parent, and Ann Nowé. Homo egualis reinforcement learning agents for load balancing. In *Innovative Concepts for Agent-Based Systems*, pages 81–91. Springer Berlin / Heidelberg, 2003.

[165] Katja Verbeeck, Ann Nowé, Johan Parent, and Karl Tuyls. Exploring selfish reinforcement learning in repeated games with stochastic rewards. *Autonomous Agents and Multi-Agent Systems*, 14(3):239–269, November 2006. ISSN 1573-7454.

[166] Alessandro Vespignani. Predicting the behavior of techno-social systems. *Science*, 325(5939): 425–428, July 2009.

[167] José M. Vidal. *Fundamentals of Multiagent Systems: Using NetLogo Models*. Unpublished, 2006.

[168] Wen X. Wang, Bing H. Wang, Bo Hu, Gang Yan, and Qing Ou. General dynamics of topology and traffic on weighted technological networks. *Physical Review Letters*, 94(18), 2005.

[169] Xiaofeng Wang and Tuomas Sandholm. Reinforcement learning to play an optimal nash equilibrium in team markov games. In *in Advances in Neural Information Processing Systems*, volume 15, pages 1571–1578, 2002.

[170] Christopher J. C. H. Watkins and Peter Dayan. Technical note: Q-learning. *Machine Learning*, 8(3):279–292, May 1992.

[171] Duncan J. Watts and Steven H. Strogatz. Collective dynamics of small-world networks. *Nature*, 393(6684):440–442, June 1998. ISSN 0028-0836.

[172] Gerhard Weiss, editor. *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. The MIT Press, March 1999. ISBN 0262232030.

[173] B. P. Welford. Note on a method for calculating corrected sums of squares and products. *Technometrics*, 4(3):419–420, 1962. ISSN 00401706.

[174] Ward Whitt. Deciding which queue to join: Some counterexamples. *Operations Research*, 34 (1):55–62, 1986. ISSN 0030364X.

[175] Wayne Winston. Optimality of the shortest line discipline. *Journal of Applied Probability*, 14 (1):181–189, 1977. ISSN 00219002.

[176] U. Wolff. Monte carlo errors with less errors. *Computer Physics Communications*, 156(2): 143–153, January 2004. ISSN 00104655.

[177] D. H. Wolpert, K. R. Wheeler, and K. Tumer. Collective intelligence for control of distributed dynamical systems. *EPL (Europhysics Letters)*, 49(6):708–714, 2000.

[178] David H. Wolpert and William G. Macready. No free lunch theorems for search. Technical Report SFI-TR-95-02-010, Santa Fe Institute, Santa Fe, NM, February 1995.

[179] David H. Wolpert and Kagan Tumer. Optimal payoff functions for members of collectives. *Advances in Complex Systems (ACS)*, 4(02):265–279, 2001.

[180] David H. Wolpert and Kagan Tumer. Collective intelligence, data routing and braess' paradox. *J. Artif. Int. Res.*, 16(1):359–387, 2002. ISSN 1076-9757.

[181] David H. Wolpert, Kagan Tumer, and Jeremy Frank. Using collective intelligence to route internet traffic. In *Proceedings of the 1998 conference on Advances in neural information processing systems II*, pages 952–958, Cambridge, MA, USA, 1999. MIT Press. ISBN 0-262-11245-0.

[182] J. W. Wong. Queueing network modeling of computer communication networks. *ACM Comput. Surv.*, 10(3):343–351, 1978. ISSN 0360-0300.

[183] Dan Wu, Bijan Parsia, Evren Sirin, James Hendler, and Dana Nau. Automating daml-s web services composition using shop2. In *The Semantic Web - ISWC 2003*, pages 195–210. Springer Berlin / Heidelberg, 2003.

[184] Ying Xiong, Wei Chen, Daniel Apley, and Xuru Ding. A non-stationary covariance-based kriging method for metamodelling in engineering design. *International Journal for Numerical Methods in Engineering*, 71(6):733–756, 2007.

[185] Kenny Q. Ye. Orthogonal column latin hypercubes and their application in computer experiments. *Journal of the American Statistical Association*, 93(444):1430–1439, 1998. ISSN 01621459.

[186] C. Y. Yin, B. H. Wang, W. X. Wang, G. Yan, and H. J. Yang. Traffic dynamics based on an efficient routing strategy on scale free networks. *The European Physical Journal B - Condensed Matter and Complex Systems*, 49(2):205–211, January 2006.

[187] Chongjie Zhang, Victor Lesser, and Prashant Shenoy. A multi-agent learning approach to online distributed resource allocation. In *IJCAI 2009, Proceedings of the Twenty-first International Joint Conference on Artificial Intelligence*, pages 361–366, July 2009.

[188] George K. Zipf. *Human Behaviour and the Principle of Least Effort*. Hafner Publishing Co Ltd, new issue of 1949 ed edition, 1949. ISBN 0028558308.