Collaborative Reinforcement Learning of Autonomic Behaviour

Jim Dowling, Raymond Cunningham, Eoin Curran and Vinny Cahill
Distributed Systems Group, Trinity College Dublin
{*jpdowlin,rcnnnghm,vjcahill@cs.tcd.ie, currane@maths.tcd.ie*}

Abstract

This paper introduces Collaborative Reinforcement Learning (CRL), a coordination model for solving system-wide optimisation problems in distributed systems where there is no support for global state. In CRL the autonomic properties of a distributed system emerge from the coordination of individual agents solving discrete optimisation problems using Reinforcement Learning. In the context of an ad hoc routing protocol, we show how system-wide optimisation in CRL can be used to establish and maintain autonomic properties for decentralised distributed systems.

1 Introduction

Massive autonomic distributed computer systems, on a scale comparable with biological autonomic systems, require a decentralised, bottom-up approach to their construction. The benefits of such an approach include improved robustness and scalability, the possibility of self-regulation, self-configuration and self-organisation, the lack of centralised points of failure or attack, as well as possible evolution of the system through evolving the local rules of the agents [1].

Collaborative Reinforcement Learning (CRL) is a bottom-up approach to tackling the complex time-varying problems of engineering autonomic behaviour for distributed systems where there is no support for global state. It is an extension to Reinforcement Learning [2] (RL) for solving system-wide optimisation problems in decentralised multi-agent systems. In CRL, individual agents solve discrete optimisation problems using RL and share solution information with their neighbours, contributing towards the solution of the system-wide optimisation problem. Agents are part of a dynamic population, with support for agents joining and leaving the system and establishing connections with neighbours. CRL does not make use of system-wide knowledge and individual agents only know about and communicate with their neighbours.

It is our belief that many autonomic properties of distributed systems that can be represented as system-wide optimisation problems can be solved using CRL. Distributed systems such as ad hoc networks, peer-to-peer and grid computing applications contain optimisation problems such as how to optimise load-balancing in a grid-computing system or the optimal distribution of meta-data in a peer-to-peer system. In this paper we introduce SAMPLE, a routing protocol for ad hoc networks [3], as an implementation of CRL and describe its autonomic properties such as the adaptation of traffic flows to congestion and interference and the favouring of stable network links by network traffic.

2 CRL

Collaborative Reinforcement Learning (CRL) is an extension to RL that uses system-wide optimisation to establish and maintain system-wide properties over a group of decentralised agents, e.g. properties such as fault tolerant or load-balanced. We view an autonomic property of a distributed system as a system-wide property that contributes to the system's self-management. Existing techniques that introduce system-wide properties into distributed systems, such as group communication protocols, communicating sequential processes and dynamic software architectures, do so in a top-down manner, decomposing system behaviour and making it amenable to formal analysis. These approaches are not suitable for dynamic environments or environments that have no support for global state, such as wireless ad-hoc networks.

CRL extends RL with a coordination model that describes how agents cooperate to solve a system-wide optimisation problem composed of a set of discrete optimisation problems (DOP). It is inspired by swarm intelligence algorithms [1]. The solution of the set of DOPs that make up the system-wide optimisation problem is initiated at some starting agent or set of agents and distributed across a partially connected set of agents resulting in near-optimal use of system-wide resources. Each DOP is modelled as an absorbing Markov Decision Process (MDP).

CRL solves system-wide optimisation problems by specifying how individual agents can either solve a DOP using reinforcement learning and share their results with neighbours using localised advertisement or delegate the solution of a DOP to a neighbouring agent by transferring responsibility for the solution to the DOP, see Figure 1. DOPs can be delegated multiple

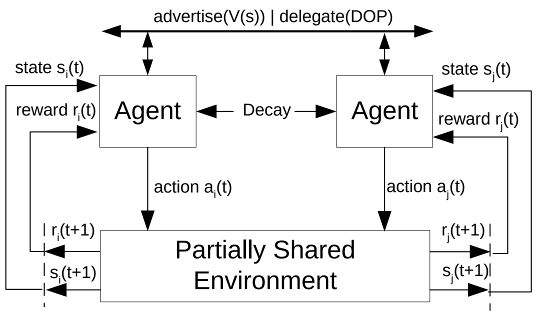


Figure 1. CRL

times across many neighbours before they are handled. An agent delegates the solution to a DOP to a neighbour when it either cannot solve the problem locally or when the estimated cost of solving it locally is higher than the estimated cost of a neighbour solving it.

2.0.1 Optimal Policy in CRL

In RL, the cost of solving a DOP is described by the estimated value function, $V(s)$ - the expected performance of the agent if it starts in state, s, and executes an optimal state transition policy thereafter. We can also use the optimal action-value function, $Q(s,a)$, instead of $V(s)$, to describe the perfomance of the agent if the optimal set of actions are executed starting from state, s. The value function and action-value function for the optimal policy are related using $V(s) = \max_a Q(s,a)$.

In CRL state transitions may be local on the current agent or be remote to a neighbouring agent. When estimating the cost of the transition to a state on a neighbouring agent we also have to take into consideration the connection cost to the neighbouring agent. For this reason, we use a different estimated optimal action-value function, $Q_i(s,a)$ at agent n_i that includes both the optimal value function for the state, $V_i(s)$, and the connection cost to the state, $D_i(s',a,s)$. The connection cost for transition to a state that is on the same agent as the agent is zero, whereas the connection cost for a transition to a state located on a neighbouring agent should reflect the underlying network cost as well as the cost of transferring control from the source agent to the target agent. The transfer of control involves terminating the DOP at the originating agent and starting the solution to a new DOP at the destination agent.

2.0.2 Advertisement

When an agent executes actions it receives reinforcements from its environment that cause updates to the V values of an agent. In CRL, neighbours are informed of changes to an agent's V values using *advertisement* to broadcast V values to neighbours, see Figure 1. Each neighbour then uses the V values and their estimated connection cost to the agent to update their cached V_i values. Examples of mechanisms for implementing advertisement updates in distributed systems include periodic broadcast of updates, conditional broadcast and event-based notification. In our ad hoc routing protocol, SAMPLE, we use the shared radio communication channel to piggy-back V updates inside 802.11 unicast and broadcast packets that are promiscuously received by neighbours.

2.0.3 Dynamic Environments

Reinforcement learning is not suitable for non-stationary (dynamic) environments. In decentralised distributed systems, it is not possible for an agent to have perfect and complete knowledge of the state of its neighbours, connections and environment. However, similar to RL, CRL models are based on MDP learning methods that require complete observability [2]. To overcome problems related to partially observable environments our statistical models favour more recent observations using a finite-history-window [2].

In decentralised distributed systems, it also is important for agents to be able to both discover new agents and to be able to 'forget' old agents and solutions as the environment changes. CRL provides additional support for a discovery action that an agent can execute in any state to attempt to find new neighbours. In SAMPLE, the discovery action is implemented using 802.11 broadcast. There is also a model for the decay of V_i information [3] where, in the absence of new advertisements of $V(s)$ values by a neighbour, an agent decays its cached $V(s)$ values. The absence of $V(s)$ value advertisements amounts to negative feedback and allows us to discard states, and hence agents, with stale values in the system. The rate of decay is configurable, with higher rates for more dynamic network topologies. As a result of the decay model in CRL, there is a requirement for a continual critical mass of advertisements to maintain knowledge of system structure. This property is similar to autopoeisis found in self-organising systems [1].

2.0.4 Model-Based RL

Reinforcement learning strategies can be either model-free or model-based [2]. We favour a model-based approach to building autonomic distributed systems using CRL, as the use of a model can be seen as allowing 'virtual experiments'. For example, if a $V(s)$ generated value changes, a model-free method may require many experiences of actions resulting in state s before this change can be incorporated into neighbouring states. However, a method that uses a model can use the observed, statistical information about state transition probabilities, $T(s,a,s')$, about which actions resulted in state s in the past to propagate this change in $V(s)$ to neighbouring states without the need for actually executing actions which result in state s. In distributed systems where real-world experience is expensive, the model based approach has a distinct advantage over model-free methods.

2.1 CRL Algorithm

The CRL algorithm can be used to solve system-wide optimisation problems that can be characterised as a multi-agent system, and where agents solve discrete optimisation problems modelled as MDPs in the following schema:

- A set of *agents* $\mathcal{N} = \{n_1, n_2, \ldots, n_M\}$ corresponding to nodes in a distributed system.
- Each agent n_i has a set, $\mathcal{V}_i$, of neighbouring agents where $\mathcal{V}_i \subseteq \mathcal{N}$ and $n_i \notin \mathcal{V}_i$.
- Each agent n_i has a set of states $\mathcal{S}_i$, where $\mathcal{S}_i \subseteq \mathcal{S}$ and $\mathcal{S}$ is the system-wide set of states.
- Nodes have both *internal* and *external states.* $Int : \mathcal{N} \rightarrow \mathcal{P}(\mathcal{S})$ is the function that maps from the set of agents to a non-empty set of internal states that are not visible by neighbouring agents. $Ext : \mathcal{N} \rightarrow \mathcal{P}(\mathcal{S})$ is the function that maps from the set of agents to a set of externally visible states. $Ext(n_i)$ is the set of states visible to neighbours of n_i. The relationship between internal and external states is the following:
$$\begin{array}{c} Int(n_i) \subset \mathcal{S}_i \\ Ext(n_i) \subset \mathcal{S}_i \end{array} \wedge \begin{array}{c} Int(n_i) \cup Ext(n_i) = \mathcal{S}_i \\ Int(n_i) \cap Ext(n_i) = \{\} \end{array}$$
- We define a set of *causally connected states* between agents n_i and n_j as: $\mathcal{C}_{n_i n_j} = Int(n_i) \cap Ext(n_j)$ where $n_j \in \mathcal{V}_i$. $s \in \mathcal{C}_{n_i n_j}$ is a causally connected state where an internal state s at n_i corresponds to an external state s at n_j. Causally connected states enable the delegation of a MDP from one node to another. In CRL a state transition to s at n_i terminates the MDP at n_i and initiates a new MDP starting at state s at n_j.
- Each agent n_i has a set of actions $\mathcal{A}_i = \mathcal{A}_{d_i} \cup \mathcal{A}_{p_i} \cup discovery$, where $\mathcal{A}_i \subseteq \mathcal{A}$. $\mathcal{A}_{d_i}$are the set of delegation actions that represent an action that attempts to delegate a MDP from $n_i$to n_j, $\mathcal{A}_{p_i}$ are the set of DOP actions that attempt to solve the MDP locally, and $discovery$ is a discovery action. $discovery$ updates the set of neighbours, $\mathcal{V}_i$, for agent, n_i, and queries if discovered neighbouring agent n_j provides the capabilities to accept a delegated MDP from n_i. If it does, $\mathcal{A}_{d_i}$ is updated to include a new delegation action that can result in a state transition to $s \in \mathcal{C}_{n_i n_j}$, delegating a MDP from n_i to n_j. The discovery action is *not* included in our calculation of the V_i values, because it should never be part of the pure exploitative strategy which the Q_i values represent.
- $D_i : \mathcal{S}_i \times \mathcal{A}_{d_i} \times \mathcal{S}_i \rightarrow \mathbb{R}$ is the connection cost function that observes the cost for the attempted use of a connection in a distributed system. The connection cost is based on a statistical model that estimates the cost of using a connection in a distributed system.
- We define a cache at n_i as $Cache_i = \{(Q_i(s,a), r_j) : r_j \in \mathbb{R} \wedge s \in \mathcal{C}_{n_i n_j}\}$. The value r_j in the pair $(Q_i(s,a_j), r_j)$ corresponds to the last advertised $V_j(s)$ received by agent n_i from agent n_j.
- For each $n_j \in \mathcal{V}_i$, $Cache_j$ is updated by a V_j *advertisement* for a shared causally connected state. The update replaces the r_j element of the pair $(Q_i(s,a), r_j)$ in $Cache_i$ with the advertised V_j value .
- $Decay(r_j) \rightarrow \mathbb{R}$ is the decay model that updates the r_j element in $Cache_i$,
$$Decay(r_j) = r_j + \rho^{td}$$
where td is the amount of time elapsed since the last received advertisement for r_j from agent n_j.
- CRL model-based learning requires learning the state transition model, $T(s,a,s')$. This can be provided by the user or estimated during the learning trial by observing actual state transitions. It is application dependent.

- The distributed model-based Q-learning algorithm is:

$$Q_i(s,a) = \begin{array}{l} R(s,a) + \gamma \sum_{s' \in S_i} T(s,a,s'). \\ (D_i(s',a,s) + Decay\,(V_j(s'))) \end{array} \quad (1)$$

  where $a \in \mathcal{A}_d$. If $a \notin \mathcal{A}_d$, this defaults to the standard model-based Q-Learning algorithm [2]. $R(s,a)$ is the MDP termination cost, $P(s'|s,a)$ is the state transition model that computes the probability of the action a resulting in a state transition to state s', $D_i(s'|s,a)$ is the estimated connection cost and $V_j(s')$ is $r_j \in (Q(s,a), r_j) \in Cache_i$ if $a \in \mathcal{A}_d$, and $V_i(s')$ otherwise.

- V_i values, at node n_i, can be calculated using the Bellman optimality equation [2]:

$$V_i(s) = \max_a \left[Q_i(s,a)\right]$$

- Finally, a *cleanup updater* is available at each agent, n_i, to remove stale elements from its set of neighbours, $\mathcal{V}_i$, delegation actions, $\mathcal{A}_{d_i}$, connected states, $\mathcal{C}_{n_i n_j}$ and its $Cache_i$. When a $(Q_i(s,a_j), r_j)$ entry in the cache drops below a specified threshold, the cleanup updater removes the delegation action a_j from $\mathcal{A}_{d_i}$, the stale connected state s from $\mathcal{C}_{n_i n_j}$, and the pair $(Q_i(s,a_j), r_j)$ from $Cache_i$. If after removing s, $c_{s_i s_j} = \{\}$ for j some neighbour of n_i, then n_j is removed from $\mathcal{V}_i$.

3 SAMPLE: Ad-hoc Routing using CRL

Ad hoc routing exhibits challenging problems such as the lack of global knowledge at any particular node in the ad hoc network and the requirement for the system-wide autonomic properties of the protocol to emerge from local routing decisions at routing agents. SAMPLE is a probabilistic on-demand ad hoc routing protocol based on CRL [3] that possesses system-wide properties, such as the adaptation of network traffic patterns around areas of congestion and wireless interference and the exploitaton of stable routes. Whereas standard ad hoc routing protocols such as Ad hoc On-Demand Distance Vector Routing (AODV) and Dynamic Source Routing protocol (DSR) use discrete models of links in the network, in SAMPLE, we use a statistical model of links based on RL and routing agents share their link information with neighbours using CRL. Routing decisions are based both on locally acquired experience using RL and information acquired from neighbours using CRL, see Figure 2.

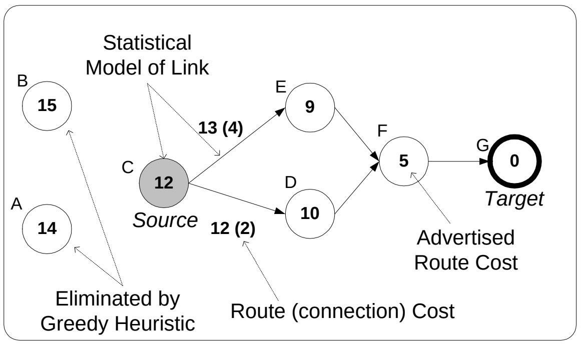
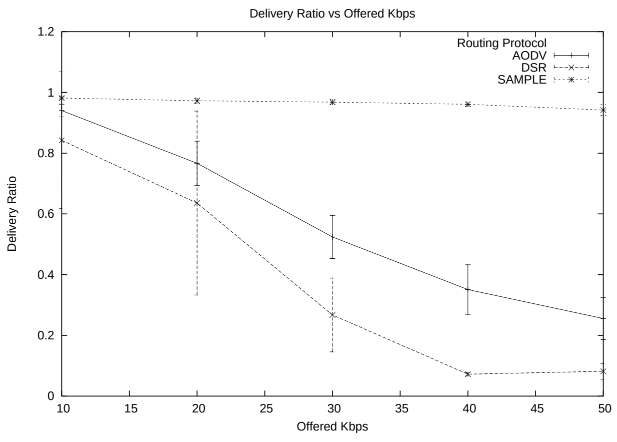


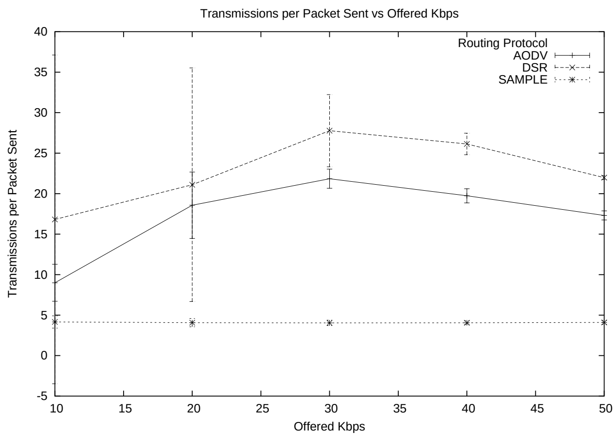
Figure 2. Routing Decision in SAMPLE

We have implemented the SAMPLE routing protocol in the NS-2 network simulator and performance results show better performance in the face of adverse network conditions than AODV and DSR [3]. In our experimental setup and simulations, there are 33 fixed nodes, 50 mobile nodes and 3 server nodes are the fixed nodes at the centre of the simulation arena. The fixed nodes in the simulation provide stable links in the network that the routing protocols could exploit. Figure 3 shows the variation in performance of SAMPLE, AODV and DSR as the number of clients in the network is increased. For these figures the packet size sent by clients was kept fixed at 64 bytes, sent 3 times a second. As the number of clients in the network is increased, the offered throughput to the routing protocols is increased. This in turn increases the level of network congestion and the amount of contention that the MAC protocol must deal with. This increased congestion reduces the ratio of packets that are successfully delivered due to an increased number of failed MAC unicasts in the network.

SAMPLE performs better than existing ad-hoc protocols in the presence of failed unicasts due to the ability of routing agents to learn that a link failed to some transient factor (and hence retrying the link may succeed) or some more serious factor such as link failure, and retrying the link is unlikely to succeed. Existing protocols assume the link has failed and perform poorly when packet error rates increase. As routing agents share their experience about links it collectively improves their rate of learning and convergence on more optimal system-wide routing behaviour. SAMPLE displays the system-wide property of routing traffic around areas of congestion or wireless interference. This property optimises throughput available in the network and emerges from the solution to and interaction of the individual routing DOPs at nodes. An important lesson from SAMPLE is the need for experimentation, as the emergence of more optimal routing properties is sensitive to tuneable parameters in CRL and the RL system model in SAMPLE.



(a) Delivery Ratio



(b) Throughput

Figure 3. SAMPLE Performance with Varying Load. 64 byte packets

3.1 Autonomic Properties of SAMPLE

From a system perspective, routing can be considered as a system-wide, continuous optimisation problem but from the perspective of an individual peer routing can be considered as a discrete optimisation problem - each packet will have start and termination states and make a discrete number of state transitions. Individual agents, from a population of agents, attempt to find a minimum cost solution to a routing problem, see Figure 2, and inform their neighbours of their lowest cost solution, who may use that information to update their solutions to the same routing problem. System-wide autonomic properties that can be observed in SAMPLE include the exploitation of stable routes by packets where possible and the re-routing of network traffic patterns around congestion and interference spots. Both of these properties emerge from the solution to and interaction of the individual discrete optimisation problems, i.e. routing decisions, that operate only on local statistical models. As can be seen from the results in figure 3, the autonomic properties of the routing protocol improve routing performance, particularly in congested and lossy wireless networks.

3.2 Discussion and Future Work

We are currently investigating the relationship between system-wide optimisation and the establishment and maintanance of various autonomic properties in decentralised distributed systems. As a different application of CRL, we are building a system that optimises resource utilisation in an ad hoc networks using load-balancing. While certain autonomic properties, such as self-optimisation can often be cast as system-wide optimisation problems, it is less clear how self-healing, self-configuration and self-protection can fit into this scheme. CRL relies on agents being able to represent an activity using a cost, and we are investigating how we can represent different autonomic properties using the cost model.

4 Conclusions

This paper introduces Collaborative Reinforcement Learning, a coordination model for solving system-wide optimisation problems in distributed systems where there is no support for global state. It can be used as a technique for establishing autonomic distributed system properties using only local discrete optimisation rules and information sharing among agents. We have shown in SAMPLE that autonomic behaviour can be introduced into distributed systems using CRL.

References

[1] J. Kennedy and R. C. Eberhart, *Swarm Intelligence*. San Francisco, California: Morgan Kaufmann, 2001.

[2] R. Sutton and A. Barto, *Reinforcement Learning*. MIT Press, 1998.

[3] E. Curran and J. Dowling, "Sample: An on-demand probabilistic routing protocol for ad-hoc networks," *Technical Report: Dept. of Computer Science, Trinity College Dublin*, 2004.