The Knowledge Engineering Review, Vol. 21:3, 231–238.
doi:10.1017/S0269888906000956 Printed in the United Kingdom

Building autonomic systems using collaborative reinforcement learning

JIM DOWLING, RAYMOND CUNNINGHAM, EOIN CURRAN and VINNY CAHILL

Distributed Systems Group, Department of Computer Science, Trinity College, Dublin; e-mail: jdowling@cs.tcd.ie, rcnnnghm@cs.tcd.ie, currane@maths.tcd.ie, vjcahill@cs.tcd.ie

Abstract

This paper presents Collaborative Reinforcement Learning (CRL), a coordination model for online system optimization in decentralized multi-agent systems. In CRL system optimization problems are represented as a set of discrete optimization problems, each of whose solution cost is minimized by model-based reinforcement learning agents collaborating on their solution. CRL systems can be built to provide autonomic behaviours such as optimizing system performance in an unpredictable environment and adaptation to partial failures. We evaluate CRL using an *ad hoc* routing protocol that optimizes system routing performance in an unpredictable network environment.

1 Introduction

Massive autonomic distributed computer systems, on a scale comparable with biological autonomic systems, require a decentralized, 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-organization, the lack of centralized points of failure or attack, as well as possible evolution of the system through evolving the local rules of the agents (Kennedy & Eberhart, 2001).

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 (RL) (Sutton & Barto, 1998) for solving system optimization problems in decentralized multi-agent systems represented as a set of discrete optimization problems (DOPs). CRL agents collaboratively solve DOPs using RL and share solution information with their neighbours, contributing towards the solution of the system optimization 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 knowledge and individual agents only know about and communicate with their neighbours.

Many desired autonomic properties of distributed systems that can be represented as system optimization problems can be solved using CRL. Distributed systems such as mobile *ad hoc* networks (MANETs), peer-to-peer (P2P) and Grid computing systems contain optimization problems such as load-balancing (Dowling, 2004), resource allocation and the optimal distribution of data in a P2P system (Sacha & Dowling, 2005). In this paper we present a routing protocol for *ad hoc* networks called SAMPLE 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 Distributed reinforcement learning challenges

RL is an unsupervised learning technique, where a single agent learns to optimize its interaction with an uncertain environment. RL agents are usually modelled as *Markov decision processes* (MDPs) (Sutton & Barto, 1998). A MDP consists of a set of states, $\mathcal{S}=\{s_1, s_2, \ldots, s_N\}$, a set of actions, $\mathcal{A}=\{a_1, a_2, \ldots, a_M\}$, a reward (or reinforcement) function $R(\mathcal{S}, \mathcal{A})\rightarrow\mathbb{R}$ that is provided by the agent's environment. RL agents observe their current state, s, and execute an action a, and their environment returns an immediate reward, $R(s,a)$. Agents learn the *optimal policy* for interacting with their environment by taking actions that are expected to maximize the total reward an agent receives over some time horizon. An agent may also maintain a model of its environment, known as model-based RL, using a state transition distribution function, $P(\mathcal{S}\times\mathcal{A})\rightarrow\Pi(\mathcal{S})$, where $\Pi(\mathcal{S})$ is the set of probability distributions over the set $\mathcal{S}$. We write $P(s'|s,a)$ as the probability of making a transition from state s to state s' when action a is executed. Agents can also explore their environment for better actions by selecting actions in a probabilistic manner. The effect of these features in RL means that agents can take actions that give a poor reward in the short-term in the anticipation of higher rewards in the medium/longer term. In autonomic systems, actions can be considered to be any decision that an agent may need to learn how to make, while states can be any information that is useful in making those decisions.

One of the main problems in applying RL to autonomic computing is that many problems that we wish to solve are distributed, as they require redundant agents to provide high levels of system availability and scalability. In particular, in P2P and MANETs there is a lack of global knowledge, preventing the use of a global reward function that agents would use to collectively learn a policy to optimize some *desired system property*, such as system performance or reliability. This implies that agents must deal with *system-level uncertainty* regarding the state of the system. In any reasonably complex system it is also reasonable to assume that agents will not be fully connected, but rather connected to only a small subset of agents in the systems, i.e. the agent's neighbours. Agents can communicate with their neighbours through message passing, but knowledge of the state and policies of those neighbours can become stale and agents must deal with *neighbour-level uncertainty* in the current state and policy of neighbours. To reduce these uncertainties and build self-optimizing systems using RL, agents must coordinate their policies. Agents cannot learn an optimal policy for the system, if they ignore these uncertainties and base actions on only locally acquired knowledge.

The challenge of distributed reinforcement learning is how to design mechanisms that reduce system- and neighbour-level uncertainties, and enable agents to collectively learn actions that optimize desired system properties. CRL, presented in the next section, supports the construction of agents that addresses these challenges.

3 Collaborative reinforcement learning

CRL extends RL with a coordination model that specifies how agents collaborate to perform online system optimization (Dowling *et al*., 2005). System optimization problems are modelled as a set of concurrent DOPs or combinatorial optimization problems that can be initiated at any agent and solved at any agent in the system. A DOP is defined as the selection of minimal cost alternatives from among a finite set of feasible solutions, as defined by an objective function, (Dorigo & Di Caro, 1999). In order to solve a DOP optimally, the agent where the DOP was initiated delegates its solution to a target agent in the system that can solve it with minimal cost, although the agent may also solve it locally if its estimated cost is lower. Agents in CRL do not use a global objective function to determine the minimum cost agent in the system. Instead, they choose the neighbour with the estimated minimal cost to solve the DOP. This is possible as neighbouring agents share information with one another about their estimated cost for solving the DOP. DOPs can be delegated multiple times across many neighbours before they are handled, enabling improved utilization of resources in the system.

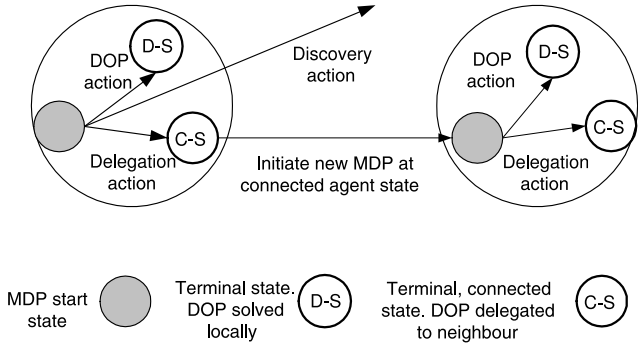

Figure 1 Collaborating agents in CRL

3.1 Optimal policy in collaborative reinforcement learning

DOPs are modelled at each agent as an absorbing MDP, i.e. a MDP that is guaranteed to enter a terminal state after a finite amount of time. CRL agents are model-based RL agents, and the estimated cost of an agent solving a DOP is described by the estimated value function, $V(s)$, for the initial state of the MDP at the agent when the DOP is started. The function $V(s)$ is the expected cost of solving the DOP at the agent if it starts in state, s, and executes an optimal state transition policy thereafter. However, as state transitions can be to remote agents (the DOP can be delegated to a neighbour), we define $DOPCost(n_i)$ as the estimated cost of agent n_i solving the DOP. Also, CRL favours model-based learning, such as Q-Learning, over model-free learning, as a model can use the observed, statistical information about state transition probabilities, $P(s'|s,a)$, about which actions resulted in state s in the past to propagate changes in $V(s)$ to adjacent states without the need for actually executing actions which result in state s. In distributed systems where executing actions to acquire real-world experience is expensive, the model-based approach has a distinct advantage over model-free methods. CRL's model-based learning approach uses an action-value function, $Q(s,a)$, instead of $V(s)$, to describe the estimated cost of solving the DOP 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 the *Bellman Equations* (Sutton & Barto, 1998):

$$V^*(s) = \max_a Q(s,a). \tag{1}$$

We modified Bellman's definition of the action-value function to take account for the fact that RL is now multi-agent, $Q_i(s,a)$ is the action-value function at agent n_i and we also added a connection cost, $D_i(s'|s,a)$, to model the additional network cost of making a transition to a state on a neighbouring agent. The connection cost for a 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.

3.2 Coordinating the solution to discrete optimization problems

In heterogeneous systems, agents typically possess different capabilities for solving a given DOP. In RL the only abstractions available to an agent are states, actions, rewards and the state transition model. To model the differing capabilities of agents, CRL allows newly discovered agents to negotiate the establishment of *connected states* with their neighbours. Connected states represent the contractual agreement between neighbouring agents to support the delegation of DOPs from one to the other (Figure 1).

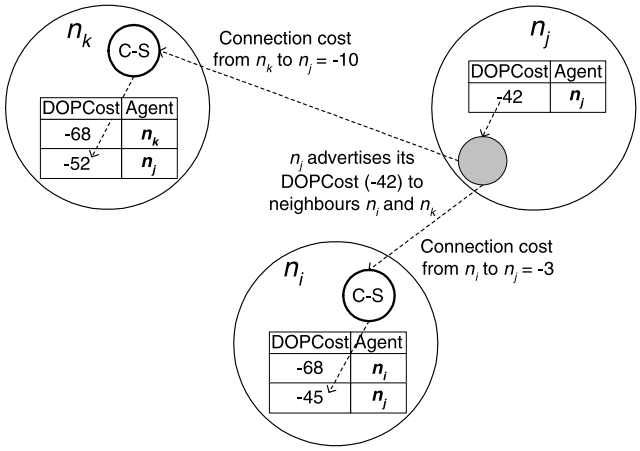

Figure 2 Advertisement in CRL

CRL also adapts the RL model by specifying three types of action defined that can be executed by a CRL agent to solve a DOP (Figure 1):

1. *DOP actions* try to solve the DOP locally at the agent;
2. *delegation actions* are coordination actions that delegate the solution of the DOP to a neighbouring agent;
3. *discovery actions* are coordination actions that an agent can execute in any state to attempt to find new neighbours.

Agents treat all three types of action equally, and attempt to solve a DOP by choosing the action with the minimal estimated cost, given the agent's current state. Discovery actions are necessary on system bootstrap when an agent does not have any neighbours, but can also be selected during normal operation with the aim of discovering new, lower-cost agents that can be added to the system.

3.3 Advertisement

When an agent executes an action it receives an instantaneous reward from its local environment that may cause an update to its DOP cost. In Figure 2, we can see how agents maintain a *cache* containing estimated costs of the agent solving the DOP locally, and an estimated cost for each neighbour. At agent n_k, the local cost for solving the DOP, $DOPCost(n_k)$, is -68, whereas the estimated cost of its neighbour n_j solving it is -52 (of which -10 is the connection cost to n_j). Agent n_k will, therefore, have a higher probability of delegating a DOP to n_j than attempting to solve it locally. The diagram also illustrates how agents *advertise* their DOP costs to one another, causing updates to the DOP costs stored in neighbours' caches. Implementation strategies for advertisement of DOP costs include periodic or conditional notification using events, or using return values from delegation actions.

In order to model network dynamism, and in particular agent and network connection failures, advertisements should be sent regularly to neighbours to indicate the agent's availability and DOP cost. In the absence of advertisements, agents *decay* their cached DOP costs for neighbours over time. The decay model allows agents to remove non-contactable agents from their set of neighbours when a cost threshold is reached. Decay enables agents to adapt their policies to a dynamic set of neighbours, as neighbours that were lower cost in the past are gradually forgotten over time in the absence of advertisements from them; for example, because they left the system. The rate of decay of costs in the cache is configurable, with higher rates more appropriate for more dynamic networks.

3.4 Distributed model-based reinforcement learning

As mentioned previously, CRL extends the RL model by including the estimated cost of using a network connection to the neighbouring agent. For this reason, we use a *distributed model-based reinforcement learning algorithm* that includes both the estimated cost for the neighbouring agent to solve the DOP and the connection cost. In the distributed model-based RL algorithm, the reward model consists of two parts. First, a *MDP termination cost* (reward), $R_i(s,a) \in \mathbb{R}$, provides an agent n_i with evaluative feedback on the cost of an action, i.e. executing a DOP, delegation or discovery action will result in the agent's local environment returning an immediate cost for that action. Second, if the action executed is a delegation action, a connection cost model, $D_i(s'|s,a) \in \mathbb{R}$, provides the estimated network cost of delegating the DOP from the local agent to a neighbouring agent. The distributed model-based reinforcement learning algorithm for delegation actions is

$$Q_i(s,a) = R_i(s,a) + \sum_{s' \in S_i} P_i(s'|s,a).(D_i(s'|s,a) + V(s')) \tag{2}$$

where s is the current state, $\mathcal{A}$ the action to be executed from the set of possible actions, $s' \in S_i$ is the next state from the set of possible next states at agent n_i, $V(s')$ is the DOP cost to the remote agent corresponding to the connected state s' and $P_i(s'|s,a)$ is the probability of the next state being s' after action $\mathcal{A}$ was executed in state s. If the action $\mathcal{A}$ is not a delegation action, $D_i(s'|s,a)$ from Equation (2) evaluates to zero. The distributed model-based reinforcement learning algorithm for DOP actions and discovery actions is:

$$Q_i(s,a) = R_i(s,a) + \sum_{s' \in S_i} P_i(s'|s,a).V(s'). \tag{3}$$

In general, costs are negative values and the estimated best action to take can be calculated as the action that minimizes $Q_i(s,a)$.

4 SAMPLE: *Ad hoc* routing using collaborative reinforcement learning

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 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 (Curran & Dowling, 2005) that possesses system properties, such as the adaptation of network traffic patterns around areas of congestion and wireless interference and the exploitation 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 the state transition model in RL and routing agents share their route cost 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 3.

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 (Curran & Dowling, 2005). In our experimental setup and simulations there are 33 fixed nodes, 50 mobile nodes and three server nodes, which 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 4 shows the variation in system throughput 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 three 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

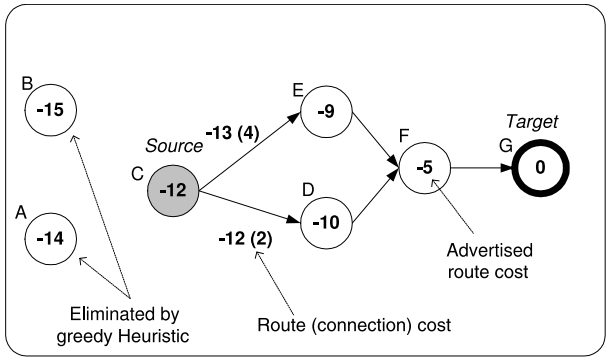

Figure 3 Routing decision in SAMPLE

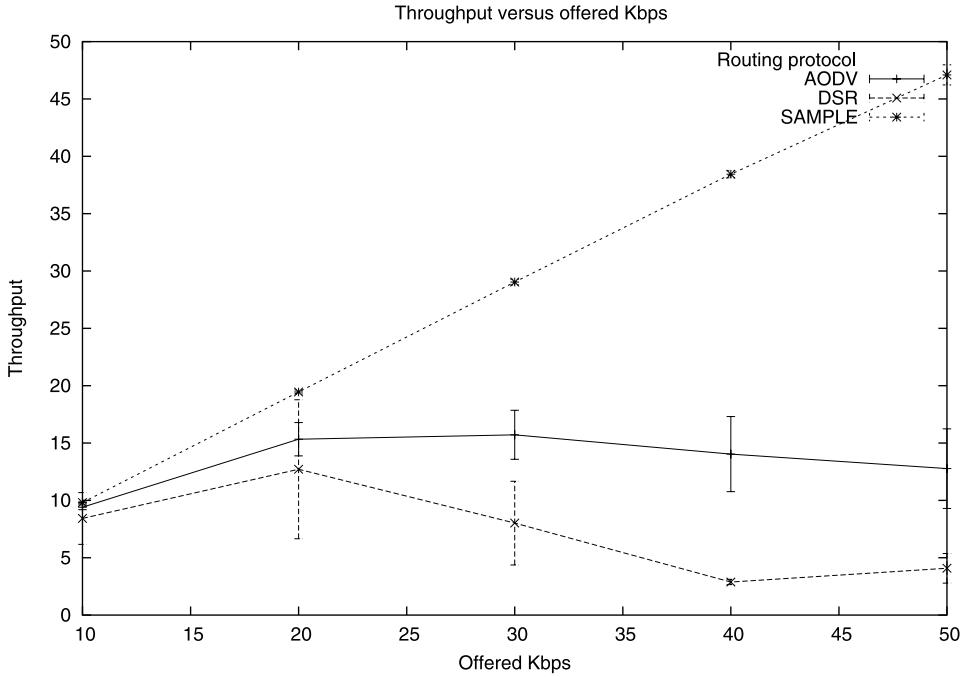

Figure 4 Routing throughput with increasing load (64 byte packets). Packets learn to exploit stable routes, improving system performance

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.

In a second experiment, we simulated wireless interference by introducing random packet loss into the simulation, i.e. increasing the packet error rate. In Figure 5, we can see that 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. AODV and DSR assume the link has failed and perform poorly when transmission error rates increase. In SAMPLE, however, routing agents share their experience about route costs, collectively improving the rate of learning and convergence on more optimal system routing behaviour. SAMPLE displays the system property of routing traffic around areas of wireless interference as well as network congestion. This property optimizes 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.

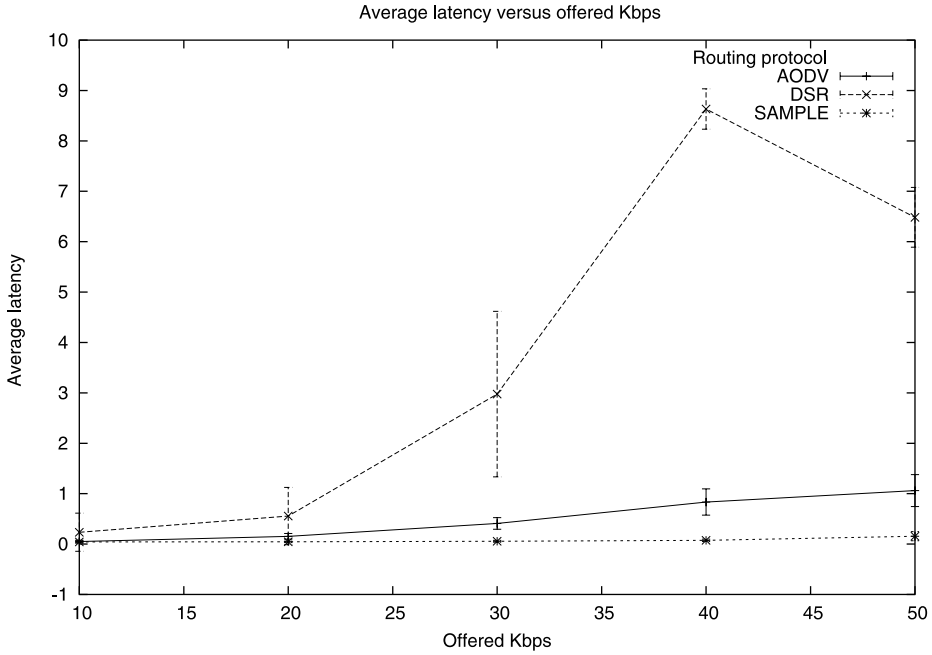

Figure 5 Average packet latency with increasing packet error rate. In SAMPLE, packets adapt to increased error rates by exploring the network for routes to the destination, increasing packet latency

4.1 Autonomic properties of SAMPLE

From a system perspective routing can be considered a continuous system optimization problem, but from the perspective of an individual agent routing is treated as a DOP where the goal of the agent is to either deliver the packet or forward the packet to the lowest cost neighbour. Note, however, that routing is a special type of DOP where there is only one agent in the system, the destination node, that is able to solve it.

The aggregate effect of a population of collaborating routing agents in SAMPLE produces system-level autonomic properties. The autonomic properties that can be observed 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 optimization problems, i.e. routing decisions, that operate only on local statistical models. As can be seen from the experimental results in the previous section, the autonomic properties of the routing protocol improve system routing performance, particularly in congested and lossy wireless networks.

4.2 Discussion and future work

In contrast to the self-optimizing routing protocol described in this paper, it is less clear how other autonomic properties such as self-configuration and self-protection can fit into the CRL model. 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 a cost model. Another application for CRL based on self-optimization that we are investigating, is maximizing the global flow of traffic in an Urban Traffic Control (UTC) system. Traffic control can be considered as a single system-wide problem that can be decomposed into a collection of discrete optimization problems, one at each traffic light junction in the UTC system. It is hoped that CRL will enable optimization of traffic flows without the need for global knowledge of traffic data.

5 Conclusions

This paper described CRL, a coordination model for solving system optimization 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 in decentralized systems, by agents collaboratively solving discrete optimization problems. We have shown in SAMPLE that autonomic system behaviour can be introduced into decentralized systems using CRL.

Acknowledgement

This work was carried out as part of the Digital Business Ecosystem project, an EU-funded FP6 IST Integrated Project, IST-2002-507953.

References

Babaoglu, O, Canright, G, Deutsch, A, Di Caro, G, Ducatelle, F, Gambardella, L, Ganguly, N, Jelasity, M, Montemanni, R and Montresor, A, 2005, Design patterns from biology for distributed computing. In *Proceedings of the European Conference on Complex Systems, Paris, France, November 2005*.

Curran, E and Dowling, J, 2005, SAMPLE: Statistical network link modelling in an on-demand probabilistic routing protocol for ad hoc networks. In *Proceedings of the 1st Conference on Wireless On-Demand Networking (WONS)*. Los Alamitos, CA: IEEE Computer Society.

Dorigo, M and Di Caro, G, 1999, The ant colony optimization meta-heuristic. *New Ideas in Optimization*. New York: McGraw-Hill, pp. 11–32.

Dowling, J, 2004, The decentralized coordination of self-adaptive components for autonomic distributed systems. PhD thesis, Department of Computer Science, Trinity College, Dublin.

Dowling, J, Curran, E, Cunningham, R and Cahill, V, 2005, Using feedback in collaborative reinforcement learning to adapt and optimize decentralized distributed systems. *IEEE Transactions on Systems, Man and Cybernetics (Part A)* (Special Issue on Engineering Self-Organized Distributed Systems), **35**(3), 360–372.

Kennedy, J and Eberhart, R, 2001, *Swarm Intelligence*. San Francisco, CA: Morgan Kaufmann.

Sacha, J and Dowling, J, 2005, A self-organising topology for master-slave replication in P2P environments. In *Proceedings of the 3rd International Workshop on Databases, Information Systems and Peer-to-Peer Computing, Trondheim, Norway, August 2005*, pp. 51–63.

Sutton, R and Barto, A, 1998, *Reinforcement Learning*. Cambridge, MA: MIT Press.