Journal of Artificial Intelligence Research 24 (2005) 407–463 Submitted 02/05; published 10/05

Cooperative Information Sharing to Improve Distributed Learning in Multi-Agent Systems

Partha S. Dutta PSA01R@ECS.SOTON.AC.UK
Nicholas R. Jennings NRJ@ECS.SOTON.AC.UK
Luc Moreau L.MOREAU@ECS.SOTON.AC.UK
School of Electronics and Computer Science,
University of Southampton,
Highfield, Southampton SO 17 1BJ, UK

Abstract

Effective coordination of agents' actions in partially-observable domains is a major challenge of multi-agent systems research. To address this, many researchers have developed techniques that allow the agents to make decisions based on *estimates* of the states and actions of other agents that are typically learnt using some form of machine learning algorithm. Nevertheless, many of these approaches fail to provide an actual means by which the necessary information is made available so that the estimates can be learnt. To this end, we argue that cooperative communication of state information between agents is one such mechanism. However, in a dynamically changing environment, the accuracy and timeliness of this communicated information determine the fidelity of the learned estimates and the usefulness of the actions taken based on these. Given this, we propose a novel information-sharing protocol, *post-task-completion sharing*, for the distribution of state information. We then show, through a formal analysis, the improvement in the quality of estimates produced using our strategy over the widely used protocol of sharing information between nearest neighbours. Moreover, communication heuristics designed around our information-sharing principle are subjected to empirical evaluation along with other benchmark strategies (including Littman's Q-routing and Stone's TPOT-RL) in a simulated call-routing application. These studies, conducted across a range of environmental settings, show that, compared to the different benchmarks used, our strategy generates an improvement of up to 60% in the call connection rate; of more than 1000% in the ability to connect long-distance calls; and incurs as low as 0.25 of the message overhead.

1. Introduction

A central challenge in multi-agent systems (MAS) research is to design mechanisms for coordinating agents that have partial, possibly mutually inconsistent, and inaccurate views of the system so that they can generate consistent solutions to complex, distributed problems. In such settings, the problem solving steps of one agent can influence those of another where they act on a common overall problem or use a set of sharable resources. Thus, to coordinate successfully, the agents need to *cooperate* by assisting each other to make better choices about the actions they take.[1] This cooperation is made more difficult because the individual agents usually have restricted capability in performing expensive computational

1. A different philosophy studied by MAS researchers is that of using *competitive* agents to find solutions to distributed problems (Takahashi & Tanaka, 2003; Walsh & Wellman, 2003). Here, however, we specifically focus on cooperative agents and thus we are firmly in the realms of cooperative distributed

activities (due to limited memory, CPU cycles, communication bandwidth, and/or communication latency) and because the target environment is usually characterised by continuous and unpredictable changes which, in turn, necessitate continuous adaptation of the problem solving process by the agents.

To overcome these problems and to coordinate effectively, the agents need a mechanism to act adaptively such that a consistent overall solution is generated. In this context, multi-agent coordination is typically based on techniques of modelling the states of other agents and enabling an agent to take actions based on these models (Dutta, Moreau, & Jennings, 2003; Dutta & Sen, 2003) (see section 2 for more details). However, given that the agents can only directly observe a limited subset of the system, they need to be provided with some information about the unobservable states to generate such models. To achieve this, we believe the agents should share some of their knowledge about their own locally observed states (at a suitable level of abstraction). This knowledge can then be used by the receiving agents to take more informed actions for better coordination. Note that in this approach, the agents cooperate by voluntarily distributing information in the system to facilitate the problem solving process. Nevertheless, practical resource bounds such as limited bandwidth and latency prohibit the use of exhaustive communication so that all agents could be made aware of the status of all other agents at all times. Thus the communication must be selective and aim to communicate the minimal amount of information that is necessary for effective coordination.

Given such constraints, it is not practical (nor possible in most cases) to generate models of all agents in the system that are accurate and up-to-date. Nevertheless, the agents can usually generate *estimates* of the un-observed states to take coordination decisions. Further, in dynamically changing systems, the agents should have a way of updating these estimates to *adapt* their problem solving decisions with the changing environment for generating quality solutions.

Now, in many applications, reinforcement learning (Sutton & Barto, 1998) (RL) has been successfully used (Ernst, Glavic, & Wehenkel, 2004; Mahadevan, Marchalleck, Das, & Gosavi, 1997; Tong, 2002) to generate such adaptive estimates in dynamic environments (see section 2 for a discussion of other alternatives). RL uses prior experience of performing tasks to develop a model of the environment. Specifically, a reinforcement learning agent receives a certain "reward" for taking an action in a given state, that acts as feedback to indicate the quality of performance against the context defined by the state-action pair. Using such rewards, RL is capable of incrementally generating robust estimates of the outcomes of different actions in different states. Such estimates provide the agent with a generic (independent of the problem) mathematical reasoning mechanism to take adaptive decisions in dynamic environments. In particular, the Q-learning (Watkins & Dayan, 1992) variant of RL is widely used because it allows estimates to be learnt without having prior knowledge about the system dynamics. Using Q-learning, with suitable training of the agent, where it is allowed to repeatedly take different actions in different states, the correct environment model can be learned from rewards. However, the assumptions underlying this result are that the agent is able to *observe all environmental states* and *receive all rewards accurately* for any action taken at any state. But, in practical MASs, this assumption is

problem solving systems (Allsopp, Beautement, Bradshaw, Durfee, Kirton, Knoblock, Suri, Tate, & Thompson, 2002; Lesser & Erman, 1988; Mailler, Lesser, & Horling, 2003).

impossible to realise because an agent can only observe a part of the complete system. This implies that it can only perceive the result of its actions within its local environment and that it may not be able to observe immediately and correctly the actions (or, their effects) taken by other agents.

To overcome the above-mentioned limitations, we have developed an effective and efficient communication protocol that a set of cooperative agents can use to learn the estimates of unobserved states in a dynamic environment. In particular, we present a novel principle of *post-task-completion* (PTC) *information sharing*. In this, agents take actions for solving a given task using their *current* estimates of the system states and then distribute their local state information to one another only *after* the task is completed. No communication is assumed during the period of task processing. Then, information is shared between those agents who cooperate to complete the task. Upon receiving this information, the agents, subsequently, update their previous estimates of the states of the other agents.

This protocol is completely generic since it is not developed based on any domain or problem-specific assumptions (see analysis of section 5). However, specific instances of PTC can be implemented for a given problem domain. For example, in section 4, we describe such an instance of PTC implemented in a call routing problem where the agents attempt to estimate available bandwidth on nodes.[2] The fact that PTC is not based on any domain-specific assumptions implies that it can be used in cooperative multi-agent problems other than call routing and to verify this, we are currently studying its applicability in a distributed fault detection application with promising initial results (Dutta, Jennings, & Moreau, 2005).

The PTC protocol is distinct from the relatively standard approach of updating estimates using the information from only nearest neighbours (hereafter, referred to as NN) while processing a task which forms the basis of a family of routing protocols (Hedrick, 1988; Tanenbaum, 2003). [3] To emphasise this fact, we choose to compare the quality of estimates learned using the PTC principle against that of NN. Specifically, the NN protocol allows information to be shared between direct neighbours only, whereas our protocol allows information to be shared between a cooperative group of agents. Furthermore, our protocol ensures more timely communication which, therefore, leads to more up-to-date estimates than NN (section 5 establishes these formally).

There has also been other research (Shen, Lesser, & Carver, 2003; Xiang, 1996; Xuan, Lesser, & Zilberstein, 2001) that has studied how sharing information between agents can aid cooperative problem solving. Typically, these approaches treat communication as a distinct part of an agent's overall decision-making problem and show how its incorporation aids in solving the latter (section 2 has more details). But, there has been little in the way of a systematic study of developing a communication protocol that is both practically applicable (in terms of it being based on realistic assumptions) and effective (in terms of improving performance) in real-life MAS. Our work, on the other hand, investigates a specific communication protocol that has both the above desirable characteristics.

2. In this application, the "state" of an agent is the bandwidth availability of its node. More details follow in section 4.
3. In our example application domain, a node i's "nearest" neighbours are those nodes that are within i's transmission range, those with whom i can directly communicate.

Against this background, in this paper, we evaluate PTC using two main approaches. First, by a mathematical analysis we demonstrate that the estimates generated by our approach are indeed more up-to-date than NN. Second, the effectiveness of our protocol is measured using a simulated distributed resource allocation problem.[4] In particular, we use a simulated wireless telephone network where the agents have to allocate bandwidth to connect calls. Communication heuristics, based upon the PTC principle, are devised to be used by the agents in this domain (see section 6 for a discussion). The performance of these are compared against two well-known algorithms used for network routing: Boyan and Littman's Q-routing (Boyan & Littman, 1993) and Stone and Veloso's Team Partitioned Opaque Transition Reinforcement Learning (TPOT-RL) (Stone & Veloso, 1999). The former is chosen because it is one of the most widely used benchmarks in learning-based network routing applications (Caro & Dorigo, 1998a; Peshkin & Savova, 2002; Stone, 2000). The latter has attracted attention more recently by being shown to be useful in a variety of domains (Stone, 2000). Hence, we believe these algorithms are reasonable benchmarks for empirical verification of PTC. Therefore, using both the formal analysis and empirical comparisons is sufficient to be able to firmly establish the merits of PTC.[5]

Empirical studies have been conducted over a wide range of environmental settings by selecting different network topologies, network loads, and dynamically changing load patterns. These studies indicate very promising results for PTC. In particular, we observe substantial improvements in the rate of successfully connected calls (up to 60%) and the ability to route calls to distant destinations with high network load (more than 1000%); both achieved by incurring a much lower communication cost in terms of information message rate (as low as 0.25) by our PTC protocol compared to the benchmark strategies in the experimental settings used; all results being statistically significant at the 95% confidence level.

The following summarises our contributions towards advancing the state of the art:

- We argue for the use of information sharing based on realistic assumptions to improve cooperative distributed problem solving.

- We propose a communication protocol independent of problem-specific features — post-task-completion sharing — for generating good estimates that learning agents can use for better cooperation.

4. We choose resource allocation because it is a generic task domain widely used in practical MAS (Chaibdraa, 1995; Cockburn & Jennings, 1996; Jennings, Norman, & Faratin, 1998). Therefore, we believe it is a reasonable choice to test our information-sharing strategy. A preliminary empirical investigation of our information-sharing mechanism in this domain can be found in the work of Dutta, Dasmahapatra, Gunn, Jennings, and Moreau (2004).
5. We also attempted to compare our algorithm with a global broadcast mechanism in which all agents issue a broadcast of their local state information whenever their states change. This mechanism was designed to verify whether system performance improves by transmitting all state-change information to all agents. Nevertheless, it is unsuitable to be used in a practical application due to its exorbitant message overhead. This was verified when we deployed it as a stand-alone application on a single machine with a dual 2.2 GHz AMD Opteron processor and 2GB memory in which it ran out of memory on the smallest topology and with the lightest load used in our experiments. However, a distributed implementation of the broadcast algorithm could be a matter of future study.

- We establish, using formal analysis, the advantage of the PTC protocol in generating more accurate estimates by ensuring a more timely distribution of information than the NN information-sharing protocol.

- We demonstrate the effectiveness of the PTC protocol using empirical studies in a representative multi-agent resource-allocation problem under a wide variety of environmental settings and against a range of other strategies.

The remainder of this paper is organised as follows. Section 2 discusses the general principles of multi-agent coordination and focuses on the specific approach of using machine learning techniques in this context. Also, the role of communication in these is analysed. In section 3, the characteristics of the cooperative multi-agent system on which we exemplify our research are outlined. It also contains a brief description of the network application that we simulate to empirically evaluate our communication principle. Section 4 presents a qualitative argument of the importance of using cooperative communication to improve learning. Also, how the PTC principle is implemented in the context of our example application is highlighted. Moreover, brief descriptions of the implementations of the benchmark algorithms (Q-routing and TPOT-RL) in this application are presented. Section 5 presents a formal analysis of the advantage of our strategy over the nearest-neighbour protocol in generating more accurate estimates. Subsequently, a detailed description of the simulation is provided in section 6. Section 7 describes the performance measures against which the various strategies are compared. It also analyses the results from our empirical studies. Finally, section 8 presents concluding remarks and identifies avenues of future work.

2. Related Work

In this section, we first review the major theoretical and empirical works on cooperative MAS that are developed around the theme of generating reliable estimates of unobserved states from limited interactions and adapting decisions in response to dynamic environments (section 2.1). Then, section 2.2 discusses cooperative MAS applications based on RL. As identified in section 1, RL makes adaptive decision-making possible without explicit domain knowledge or pre-defined rules of coordination. So it is used as the basic decision-making framework of our agents. This review specifically focuses on the use of communication in these applications and analyses the practical feasibility of the methods proposed. The shortcomings of these approaches are identified and the contributions of our research towards alleviating them is highlighted. Finally, section 2.3 discusses relevant literature in the area of network bandwidth estimation which is similar to our application domain and identifies how our learning-based approach differs from these.

2.1 Cooperative Multi-Agent Systems for Resource Allocation

In the following, we discuss the major research contributions in the area of cooperative MAS designed for resource allocation problems.

2.1.1 Functionally Accurate Cooperation

The functionally accurate, cooperative (FA/C) approach advocates the exchange of partial, tentative solutions of local problems among agents to generate consistent partial solutions (a distributed speech recogniser application, based on this concept, is developed by Lesser and Erman 1988). In turn, this helps to generate predictive information about future partial solutions that furthers the build-up of a consistent global solution. However, the cost to obtain complete and up-to-date information to build a completely consistent solution can be prohibitively large because of communication delays. Hence, in such situations, it is more practical (cost effective) to achieve a global solution that may have a tolerable degree of inconsistency via the timely exchange of partial, tentative solutions. Thus FA/C is based on the use of communication to generate consistent estimates of the global problem solving scenario, as identified as a key requirement in section 1. Nevertheless, FA/C only discusses "what" is to be done, viz., agents should cooperate with partial solutions to reach an acceptable solution quality. It does not provide a recipe of "how" it could be achieved. Our work, on the other hand, advocates a specific communication strategy for the agents to improve the learning of state estimates which, in turn, would improve the cooperative problem solving. Moreover, we provide both a formal analysis and empirical results to justify the benefits of our strategy, something which the FA/C approach does not do.

2.1.2 Organisational Structuring

Some researchers have incorporated organisational structures — patterns of information and control relationships that exist between the agents and the distribution of problem-solving abilities among them — into the agent models (Carley & Gasser, 1999). The idea is that such structures give an agent a broad, high-level knowledge about how the system solves problems, the roles that agents play, its own position in the network, and how they are connected. Imposing these structures, therefore, essentially, resolves the requirement (see section 1) of maintaining high-fidelity estimates of the portions of a system that the agents cannot directly monitor. This is analogous to the way human organisations are formed to solve complex tasks that are beyond the capability of "rationally bounded" (March & Simon, 1958) individuals. In our work, the agents do not follow pre-defined structures of roles and relationships. Instead, they learn, via cooperative communication, the estimates of the agents' states so that the dependencies can be inferred and the optimal actions can be taken. Thus, our cooperation model is applicable across domains without requiring explicit organisation structures to be specified.

2.1.3 Sophisticated Local Control

The sophisticated local control methodology (for example, the *p*artial *g*lobal *p*lanning (PGP) approach Durfee & Lesser, 1991) advocates that the cooperating agents should reason about how to exchange information to resolve inconsistencies, whom to interact with to improve cooperation, what information exchange can achieve that objective, and the like.

Now, in PGP, the agents form contracts, plan their actions and interactions, negotiate over plans, use organisational information to guide their planning and problem-solving decisions, tolerate inconsistent views, and converge on acceptable network performances in dynamic environmental conditions. In this model, each agent maintains its own set

of PGP's — a set of local plans that represent the agent's view of the global problem solving situation. They are updated by the exchange of local, partial plans among agents and reflect the most recent network scenario in terms of achieving the global solution. Hence, this methodology addresses the requirement (see section 1) of state estimation using cooperative communication.

Elaborating on the PGP approach, the *TAEMS* (Task Analysis, Environment Modelling, and Simulation) framework (Decker, 1995a, 1995b) was developed to model the impact that the characteristics of a task environment can have on coordination. Using *TAEMS*, coordination is achieved by three broad areas of agent behaviour: how and when to communicate and construct non-local views of the current problem solving situation; how and when to exchange the partial results of problem solving; how and when to make/break commitments made to other agents about what results will be available and when. The generalised partial global planning (GPGP) consists of a group of coordination mechanisms based on the above broad behavioural types (Decker & Lesser, 1995). Depending on the characteristics of the task environment, agents select the appropriate coordination mechanism. Unlike PGP, however, GPGP distinguishes local scheduling of an agent from its coordination activities: the coordination mechanisms provide an agent non-local views of the problem and the local scheduler creates plans (including both local actions and non-local effects via such actions) to improve global system-wide utility by using information from the coordination mechanisms. Thus, GPGP is developed around the principle of selecting actions based on estimates of the non-local states, as identified in section 1.

However, both PGP and GPGP employ, albeit flexibly, a set of predetermined coordination mechanisms. Such preplanned coordination can prove to be inadequate against all sorts of contingencies that can occur in domains where agents maintain incomplete, incorrect views of the world state (which change non-deterministically) and may even fail without prior indication. In contrast, our approach of learning to map the agents' views of the world states to the selection of actions which would guarantee the improvement of the global system performance, requires no such pre-specified coordination rules.

2.1.4 Teamwork Based on Joint Intentions

Probably the most comprehensive cooperative MAS framework existing in current literature is STEAM (Tambe, 1997) (see also Jennings, 1995; Rich & Sidner, 1997). It is developed around the principles of the joint intentions theory (Cohen & Levesque, 1991) and joint commitments (Jennings, 1993). To coordinate, the agents maintain a "joint persistent goal" (JPG) that the team is jointly committed to for doing some team activity, while mutually believing that they are doing it. Agents in STEAM arrive at a JPG by exchanging speech acts: "request", that they use to announce their individual partial commitments about attaining the global goal, and "confirm", which establishes that an agent has the same partial commitment to the one who made the "request". Further, STEAM borrows principles from the "shared plans" model (Grosz & Kraus, 1996) to ensure team coherence so that all team members follow a common solution path.

Although STEAM provides a principled framework for reasoning about coordination in teamwork, achieving a joint belief in large systems of widely distributed agents is, in most cases, likely to be a performance bottleneck rather than an advantage because of the

excessive communication required to achieve it. This is especially true in environments where the agents have to re-plan the task execution process in response to environmental changes. Although STEAM treats this issue by using a replanning protocol, it requires the re-establishment of joint commitments among all the team members. A significant amount of communication overhead (message flow and delay) might be incurred before this is achieved which can degrade the quality of service significantly in some applications. Thus it is important to have a cooperation model that allows the agents to continue solving the overall task without requiring them to establish a system-wide commitment whenever replanning occurs. Our learning-based cooperation model has this advantage.

2.2 Learning-Based Cooperative Multi-Agent Systems

Now we review some key applications that use machine learning techniques for solving multi-agent cooperation problems.

2.2.1 Q-Routing in Dynamic Networks

RL is applied to the problem of cooperative distributed problem solving in a seminal piece of work by Boyan and Littman (Boyan & Littman, 1993) where they solve a network packet routing problem. In their paper, they model each communication node on an irregular 6×6 grid as a reinforcement learner who maintains estimates of the delays in routing packets to different destinations. To route a packet to a given destination, an agent requests each of its neighbouring agents for their delay estimates for that destination node. Upon receiving the delay estimates of its neighbours, the requesting agent chooses the neighbour with the minimum delay estimate to forward the packet. It then updates, using standard Q-learning, its prior delay estimate for that destination with the estimate that it received from this neighbour.

The authors of this paper demonstrate, using empirical studies, that their approach enables the agents to learn better policies (in terms of choosing a neighbouring agent for routing to a given destination) than a hand-coded shortest path algorithm. The differences are more pronounced when the network load increases indicating that the learning algorithm is able to adapt routing decisions (the paths along which packets are routed) under dynamic network conditions. In addition, the authors test their algorithm with changes in the network topology (by manually breaking the links between certain nodes) and in the pattern of call traffic (changing different regions in the network where calls can originate and terminate). They demonstrate that their Q-routing algorithm successfully adapts to these changes and performs better than the deterministic shortest path algorithm.

In their work, therefore, Boyan and Littman have used a simple communication protocol to allow the agents to cooperatively share their own knowledge about the packet routing delays. Nevertheless, the communication protocol they have used only allows an agent to inform its immediate neighbour about its own knowledge. This method would incur long latency for information to reach agents further away. As a result, considering states change continuously, the information can become outdated by the time an agent receives it. Such outdated information would then be of little use to generate reliable estimates of the non-local states. In this context, therefore, it is envisaged that by allowing state-change information to be shared between the group of cooperating agents only after task

completion, the agents can maintain more accurate estimates of their non-local states. This, in turn, can improve the overall performance of the cooperative MAS as achieved by Boyan and Littman (1993). Moreover, in their work, the agents update their Q-estimates with the *estimates* received from direct neighbours. Note that in so doing, one learner depends on the estimates learned by another. Thus, this approach (essentially, a TD(0)-type learning Sutton, 1988) has the potential pitfall that "bad" estimates are propagated due to the poor learning of one agent. PTC, on the other hand, advocates transmitting the *actual states* of individual agents to the others in a group. Thus, PTC is not envisaged to have the shortcomings of the approach of Boyan and Littman (1993). To verify this, we choose the Q-routing algorithm as one of our benchmarks for empirically evaluating the performance advantage of PTC.

2.2.2 Team Partitioned Opaque Transition Reinforcement Learning

The **T**eam **P**artitioned, **O**paque **T**ransition **R**einforcement **L**earning algorithm (TPOT-RL) (Stone, 2000) has the objective to make the learning task easier in a MAS by reducing the state space dimensionality. It does this by mapping the state onto a limited number of *action-dependent* features. Analogous methods of state aggregation have been used in other reinforcement learning algorithms (e.g., McCallum, 1996; Singh, Jaakkola, & Jordan, 1995) to reduce the size of the learning task. However, TPOT-RL differs from these approaches because it emphasises deriving a set of small yet informative features for effective learning. More specifically, these features are used to represent the short term effect of actions that an agent may take. Thus, the agents learn the utility of selecting actions with respect to their own feature space. This is especially useful when the agents cannot observe or immediately influence the actions taken by other agents (such as in many practical multi-agent settings, in particular, network routing). That TPOT-RL is an effective algorithm is demonstrated by its successful application across multiple domains (Stone & Veloso, 1999).

The authors have evaluated the TPOT-RL algorithm in a simulated network routing environment. The action-dependent features in this case are the load levels of a node's adjacent links. The agents transmit their delay estimates along with a packet while routing the latter. Furthermore, these estimates, collected at the corresponding destination nodes, are distributed to the nodes who participated in the routing after fixed time intervals. Thus, TPOT-RL in fact uses communication to distribute information among the agents. Their empirical studies demonstrate that TPOT-RL outperforms (performance measure is average packet delivery time) the shortest path and Q-routing protocols when learning is done under switching traffic conditions — the algorithm is trained under conditions where the selection of packet sources and destinations are changed to form two different traffic patterns. Nevertheless, the following limitations are envisaged in this work. First, identifying action-dependent features from local observations only can lead to loss of information. This is because not all non-local state changes may be reflected in an agent's immediate state space but such information may be required by an agent to select actions. Thus, in such circumstances, explicit knowledge of the non-local states is necessary. Second, as a consequence of the above-mentioned problem, the fidelity of the derived estimates would deteriorate. This, in turn, would decrease the overall performance of the system. Third, in the work of Stone (2000), since the agents update their estimates based on others' estimates

and not using the actual states, a similar shortcoming as identified in section 2.2.1 of learning bad estimates can occur. In contrast, our work attempts to alleviate these limitations by distributing the actual node states to keep the agents informed of the non-local states. Finally, in TPOT-RL, information is distributed at regular intervals (Stone, 2000), however, a formal way of specifying this interval is not prescribed. This is an arbitrary scheme which can result in large latencies in information reaching target nodes. Hence, estimates generated based on such information may not be up-to-date. PTC, on the other hand, distributes information immediately after task completion, thus, attempting to minimise the latency. Again, because of its claimed effectiveness and broad applicability, we choose TPOT-RL as the second benchmark for empirical evaluation against PTC.

2.2.3 Policy Gradient Search

Another approach of using RL for cooperative distributed problem solving is that of policy gradient search (Sutton, McAllester, Singh, & Mansour, 2000). A *policy*, in a RL context, is a mapping from a state on to an action. The policy is thus a function of a set of *parameters* which are variables defining the local state and, hence, influencing the action selection. A policy gradient search is a mechanism that tries to optimise the parameter values such that the average long-term reward of the learners is maximised. For example, in a network routing problem, these parameters can be the destination of the packet and the outgoing link a router (agent) selects for that destination (note these parameters are locally observable to an agent) while the reward is a measure of utility (or, sometimes the negative cost) that an action achieves given the parameter values. In the network routing domain, the reward can be the negative trip time for a packet to reach its destination node. In the policy gradient approach, it is assumed that each agent (the individual learners) receives the reward of all actions taken by all agents at every time step.[6] It is only this reward information that is globally known by the agents. Thus the policy gradient algorithm is model free — independent of domain models and knowledge about others' states and actions. Individual agents adjust their policy parameters in the direction of the gradient of the average reward that they compute using the global reward information — hence the term policy *gradient*. Therefore, communication of reward values is key to allow the learners to optimise the parameter values. However, the dependence on the global reward information to update the policy parameters can be a bottleneck in systems where the communication bandwidth is limited and there is a finite latency in messages to propagate (as in most practical systems). These constraints can lead to very slow responsiveness to environment changes in agents using the policy-gradient approach. Moreover, broadcasting rewards by all agents in highly dynamic environments (such as, networks experiencing heavy loads) can cause the network to completely saturate by the reward messages (as observed in our implementation of the global broadcast strategy, stated in section 1) resulting in a very inefficient system.

This method is used to build cooperative MAS by Williams (1992), Baxter and Bartlett (1999), and Peshkin and Savova (2002), among others. All of these works demonstrate that the policy-gradient search achieves reasonable performance (in terms of average routing delay) compared to other benchmark algorithms such as the shortest path algorithm. However,

6. More realistically, each agent may broadcast the reward it receives to all other agents. Hence, the agents may receive the reward signals from the entire system with some delay.

in the works of both Williams (1992) and Baxter and Bartlett (1999), an exceedingly large amount of time is required for the learners to converge. This is because the learners need the global reward information to update their policy parameters which results in a slow optimisation of the parameters. This puts a restriction on the applicability of this approach to build practical systems. A similar limitation is likely in the work of Peshkin and Savova (2002) although the authors do not provide these results. In addition, with the high communication overhead incurred, this approach is likely to be unsuitable for implementing practical MASs.

In contrast to the uninhibited communication required in the policy gradient search approach, in our work, communication is used as a controlled strategy to inform the agents about the portions of the global state that are relevant to their action selection. Thus, we believe our work offers a superior practical solution to the policy gradient approach.

2.2.4 Communication Decisions in Multi-Agent Coordination

Xuan, Lesser, and Zilberstein (2001) advocate that communication decisions are integral to an agent's decision to coordinate in a cooperative, distributed MAS. The authors consider that each agent solves a local Markov Decision Process (MDP) (Feinberg & Schwartz, 2001) that generates both a communication action and a state-changing action at every decision sub-stage. The agents are only given local observability (i.e., they cannot observe the states of other agents). However, they can observe the communication actions of other agents. The important reason for introducing a communication decision in an agent's local MDP, as argued in this paper, is that communication incurs cost. Hence, an agent should employ reasoning to decide when communication is required such that the overall utility earned from the agent's decisions is maximised. In this context, this work extends the theoretical analysis of multi-agent MDP of Boutilier (1999) where the agents are assumed to have global state information. Specifically, the authors propose two simple heuristics to generating communication decisions that aim to reduce the computational complexity of solving the full MDP to generate the optimal global policy.

As we are interested in studying the impact of communication on the performance of a cooperative MAS, the work of Xuan et al. (2001) is related to our research. However, while they analyse *whether* communication is necessary at any stage of an agent's decision, we consider communication to be inevitable. We analyse, both quantitatively and empirically, the impact of a specific information-sharing protocol, PTC, on the performance of a MAS. Our work differs from that of Xuan et al. (2001) in the following additional ways. Firstly, in their work, the agents are assumed to iterate through a sequence of communicate and act stages *synchronously*. We adopt a more generic approach of completely asynchronous behaviour. Secondly, they assume to be instantaneous communication; thus, information sent by an agent is received by another immediately. On the contrary, we consider a more realistic scenario where there is a finite delay associated with communication. Finally, the communication heuristics that they propose are based on each agent individually monitoring their own progress towards achieving a commonly agreed upon goal. In our work, we consider fundamentally distributed task processing where an agent can take a local action and the actions of multiple agents together complete a task (more on this in section 3). Hence, in our work, individual agents cannot monitor the progress of a task execution pro-

cess towards completion; they are only capable of taking their local actions using estimates of the unobserved states.

2.2.5 Other Machine Learning Algorithms

The motivation for PTC initially may appear to be similar to conventional supervised learning (SL) (Widrow & Hoff, 1960), where the actual outcome of a multi-stage prediction problem is fed back to the individual learners (predictors). However, they differ on the following critical issues:

- In SL, only the final outcome (such as whether a prediction was "correct") acts as the source of information for the learners to update their prediction algorithm. In contrast, PTC allows a learner to gain knowledge about how the states of other agents in the cooperative group change along with the outcome.
- In SL, an agent typically learns a mapping from its own actions onto the outcomes of a multi-stage prediction problem. PTC, however, allows an agent to consider the impact of the states of other agents on the final outcome of the task.
- Typically, SL is used for prediction in stationary environments. On the other hand, we evaluate the competence of PTC in maintaining high-fidelity estimates in fundamentally dynamic and uncertain environments.

Also, PTC is distinct from the approach of using eligibility traces in which a learner is provided with the knowledge of the entire sequence of state transitions after a complete training episode (i.e., after starting from the start state and reaching the goal state) (Mitchell, 1997). In the latter, an agent, upon reaching the goal state after executing a series of actions, updates in reverse order (i.e., starting from the goal state and moving to the starting state) its Q-estimates for each state transition. In a practical MAS, however, it is not possible for a single agent to observe all transitions occurring during a task processing episode as assumed in the approach using eligibility traces. Further, in a large and complex MAS, the computational load incurred by a single agent attempting to take decisions based on its knowledge of all state-transitions sequences of every task would be too prohibitive to be realised in practice. In this situation, therefore, our research contributes towards developing a practical and effective means of distributing state information to improve learning. In so doing, it removes the requirement of an agent having to maintain the entire chain of state transitions in its memory. Rather, it allows the agents to acquire an overall picture of the state-changes in the cooperative group that perform a task, which, in turn, allows them to take decisions for effective coordination.

2.3 Network Bandwidth Estimation

Network bandwidth availability directly impacts the performance of networked applications such as web services, peer-to-peer systems, and mobile networks. Therefore, effective estimation and prediction of bandwidth availability have attracted considerable attention in the networks community. Our application of PTC to do bandwidth estimation and routing, therefore, bears close resemblance to this line of research. In this paper, we use Q-learning (Watkins & Dayan, 1992) for bandwidth estimation because it generates robust

and flexible estimates from observations. Therefore, to generate the estimate, it needs to observe the bandwidth availability pattern. In network bandwidth estimation, such knowledge is harnessed by either *active* or *passive* measurements. The former is done by injecting perturbation traffic into the network and then assessing the states based on the statistical characteristics of this traffic (Jain & Dovrolis, 2002). However, in applications such as limited-bandwidth ad-hoc networks, such perturbation traffic wastefully consumes valuable bandwidth. A related approach adopted in the agent's community is that of using mobile agents or "ants" to harness traffic conditions across the networks and update node routing tables (Caro & Dorigo, 1998a, 1998b). In low-bandwidth networks, the introduction of such extraneous agents can still impact bandwidth usage. Besides, the security and privacy problems associated with the use of mobile agents have made their applicability in real-life systems debatable. Alternatively, passive measurements are done by offline analysis of actual traffic traces (Lai & Baker, 1999; Ribeiro, Coates, Riedi, Sarvotham, Hendricks, & Baraniuk, 2000). However, in our case, we require estimation to be online so that we can cope with the dynamic conditions during the operational period of the system. Therefore, PTC generates bandwidth estimates by using information disseminated by the cooperative agents *while* they route calls. Further, the passive measurement approaches typically estimate bandwidth by assuming a network model that can affect the estimation fidelity. For instance, in the work of Ribeiro et al. (2000), a network path is modelled as a single queue which disregards the effects of variable queueing delays along that path on the estimate. By choosing Q-learning, we aim to develop a statistical model of bandwidth availability by continuous monitoring without getting constrained by any predefined models.

3. Task Domain Characteristics

In most MAS, complex tasks are performed by groups of agents. Generally speaking, the problem solving activities of such agents may be executed in parallel to generate the final solution. But, in many cases, constraints over processing different, but related, parts of a task by different agents may require the task execution process to be partially sequentialised with appropriate scheduling between parallel executions (Maley, 1988; Parunak, Baker, & Clark, 2001). As a simple example, we can consider a car factory where the units manufacturing the different parts of a car can run in parallel (satisfying mutual compatibility) while the final assembly comes into play only after all the parts are correctly manufactured (Jennings & Bussmann, 2003). Alternatively, the nature of a task can enforce a strictly sequential processing. For example, in a distributed transportation system, delivery of goods between two points requires a set of cargo movements in sequence. In this paper, we exemplify the application of our PTC principle in a sequential domain. However, the choice of a sequential domain does not indicate a limitation of PTC; PTC is not based on any assumption of sequential task processing (section 4 has more details). Nevertheless, this is a reasonable choice since several key MAS applications such as sensor networks (Viswanathan & Varshney, 1997), supply chains (Denkena, Zwich, & Woelk, 2004), telecommunication bandwidth allocation (Minar, Kramer, & Maes, 1999), among others, feature, to different extents and at different levels of granularity, sequential task processing. Here, it should be clarified that although individual tasks are considered sequential, the entire multi-agent system would typically be performing multiple such sequential tasks simultaneously and asynchronously.

Further, the individual agent activities in our example application are considered equivalent to allocating resources. This is because in many practical MAS applications, such as the ones mentioned above, the agents essentially allocate resources such as network bandwidth, processor cycles, etc. to complete tasks.

Specifically, in this research, we consider a wireless telephone network (TN) as a representative application in which to implement communication heuristics based on PTC and evaluate these empirically. Note, however, that this choice does not limit the applicability of our results. This is because this application has characteristics that are common to many real-world large-scale distributed systems and hence our solutions are also more generally applicable. These include: multiple agents situated in different portions of the system (a network node is modelled as an agent), agents having incomplete views of the activities and states of others, and agents having to coordinate their local (routing) decisions in order to successfully achieve a global task (routing a call from source to destination). Moreover, there is a clear measure of success which is proportional to the bandwidth usage efficiency, or, equivalently, to the total number of successfully routed calls.

The TN uses circuit-switched communication where node bandwidth has to be allocated end to end (from a call source to destination) to establish calls. Each routing node is treated as an agent. A node has a certain amount of bandwidth that can be allocated to a certain maximum number of calls simultaneously. Each agent can monitor the load (the amount of bandwidth allocated to calls) on its node and can communicate only with the nodes within its transmission range — the set of "neighbour" nodes. Calls of finite duration can originate from and be destined to any node. Calls originate (so, terminate) continuously. Therefore, the load on the nodes continue to vary with time. The objective of any such agent is to allocate bandwidth and forward a call to one of its neighbours such that the probability of the call getting routed via the least congested (with the maximum available bandwidth) path is maximised. Routing a call at a given time along the least congested path at that time ensures an efficient use of bandwidth, hence, increases the number of successfully routed calls in the system. The forwarding is based on the agent's estimate of the congestion levels across the network, i.e., its estimate of the unobserved states. Hence, the task is completed by a sequence of such allocations and forwarding by multiple distributed agents. In this scenario, the agents continually process tasks (i.e., route calls from source to destination as new calls originate). We refer to the process of routing a call as a task processing *episode*. Section 5 uses this notion of episodic tasks to explain the difference between the estimates generated by PTC and the NN protocols. A detailed description of the simulation of various agent activities during such episodes is provided in section 6.

The next section expands on the PTC principle and outlines how information sharing based on this principle is designed in the TN domain. Furthermore, the implementations of the benchmark strategies, Q-routing and TPOT-RL are also explained.

4. Sharing Information to Improve Learning

To implement effective learning in dynamic environments by sharing local knowledge between agents, the communication strategy should satisfy the following criteria:

- **Time efficient distribution:** There is a latency associated with communication. Hence, the more timely the information that is communicated to an agent, the more likely the information is up-to-date.
- **Accuracy of information:** In continuously changing environments, it is impossible for all agents to remain synchronised with state changes at all times. Nevertheless, the more accurate the information received, the better.

Given these desiderata, here PTC is proposed as an effective strategy for distributing the local state information of agents.

Definition 1 *Post-task-completion information sharing refers to the distribution of local state information between a group of cooperative agents, by way of a mechanism that depends on the allowed agent interactions, only after the completion of tasks undertaken by the agents.*

The motivation for using this scheme is to let an agent that participated in completing a collaborative task have an indication of the state changes of the other agents in the group that resulted from processing that task. Such information is then useful for making more informed decisions while processing any subsequent task. In a dynamic system, the world states change while the agents process a given task. Therefore, by delaying the transmission of information until the task is completed, this protocol ensures that all those agents who participated in the task completion process are informed about these state changes and how that affects the outcome of the task. In so doing, we hypothesise that by using this mechanism the agents would be able to distribute information in a time-efficient manner and learn reasonably accurate estimates, thereby satisfying the requirements identified above. The analysis presented in section 5 establishes our hypothesis by comparing the timeliness and estimate qualities of PTC against those of the NN protocol.

It is important to note that PTC is a general principle and specific instances of PTC can be implemented in a given problem. Our implementation of PTC in the TN domain, discussed in section 4.2, is just such an instance. Further, although we exemplify the applicability of PTC in a sequential domain, the protocol is not designed around any such assumption. Note that in applications where task processing between agents occurs independent of one another and in a purely parallel fashion, the objective of maintaining estimates of other agents' states becomes redundant. Rather, in such systems, a central task allocator that allocates sub-tasks to the parallely executing agents would be a more suitable approach. However, if the processing of a sub-task by an agent requires estimates of the others (due to some dependencies between them), PTC can be used to distribute information between these processors. Moreover, if the sub-tasks in such systems, in turn, require sequential processing, our information-sharing protocol can be used in that context. Therefore, we can argue that communication strategies based on the PTC principle can be suitably designed to be applicable at various levels of granularity in domains other than the one chosen in this paper.

In the remainder of this section, we first outline the basics of Q-learning. Subsequently, we describe how PTC is implemented in a TN to improve the distributed learning of bandwidth availability. The Q-routing and TPOT-RL implementations are also described in the same light.