Decentralised Channel Allocation and Information Sharing for Teams of Cooperative Agents

Sebastian Stein* ss2@ecs.soton.ac.uk
Simon A. Williamson† swilliamson@smu.edu.sg
Nicholas R. Jennings* nrj@ecs.soton.ac.uk

*University of Southampton, SO17 1BJ, Southampton, UK
†School of Information Systems, Singapore Management University, Singapore

ABSTRACT

In a wide range of emerging applications, from disaster management to intelligent sensor networks, teams of software agents can be deployed to effectively solve complex distributed problems. To achieve this, agents typically need to communicate locally sensed information to each other. However, in many settings, there are heavy constraints on the communication infrastructure, making it infeasible for every agent to broadcast all relevant information to everyone else. To address this challenge, we investigate how agents can make good local decisions about what information to send to a set of communication channels with limited bandwidths such that the overall system utility is maximised. Specifically, to solve this problem efficiently in large-scale systems with hundreds or thousands of agents, we develop a novel decentralised algorithm. This combines multi-agent learning techniques with fast decision-theoretic reasoning mechanisms that predict the impact a single agent has on the entire system. We show empirically that our algorithm consistently achieves 85% of a hypothetical centralised optimal strategy with full information, and that it significantly outperforms a number of baseline benchmarks (by up to 600%).

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed AI—*multi-agent systems*

General Terms

Algorithms

Keywords

teamwork, multi-agent learning, communication

1. INTRODUCTION

It is envisaged that teams of heterogeneous software agents will be increasingly used to tackle complex real-world problems. These include intelligent sensor networks, autonomous vehicles that explore hostile environments and portable devices that provide emergency responders with situational awareness during a disaster situation. In all these multi-agent systems, and many others besides, coordination is typically achieved through communication between the agents, who share their beliefs about the state of the problem so that together they can find a better, coordinated response.

However, communication infrastructures in real-world problems often have considerable constraints. As dedicated high-bandwidth networks are costly to develop and deploy, or may simply be unavailable in an emergency situation, agents often need to rely on very limited existing communication means. These may include using mobile ad hoc networks [3], power line communications [13] or cognitive radio [1], where agents compete to utilise the spare capacity of other communication networks via spectrum sharing and channel allocation [11]. Now, using any of these constrained media introduces a new coordination challenge: how to best utilise the available capacity to communicate information effectively across agent teams.

To date, research on such systems has largely concentrated on finding a fair division of bandwidth between competing, self-interested agents in a decentralised setting. For example, there are protocols such as FDMA (Frequency Division Multiple Access) [1] and approaches that model the problem as a multi-armed bandit [12, 2] or as a congestion game [10]. However, all these solutions assume identical information needs and that each agent is only interested in maximising its own bandwidth.

Hence, existing work neglects the fact that a constrained communication system may be used by cooperative teams of agents to solve a joint problem whose global utility is not maximised by giving each of them fair access. To exemplify this challenge, we ground our work on a disaster management scenario where teams of ambulance, fire brigade and police agents must respond to an earthquake in an urban area (such as seen in the RoboCupRescue[1] competition [8]). Here, the overall joint problem is to contain the disaster and minimise damage to civilians and property. However, the individual capabilities of agents are very specific — ambulance agents rescue civilians, police agents clear roads and fire brigades control fires. Therefore, while agents may discover any type of information (e.g., an ambulance may discover a new fire), their individual information needs are highly heterogeneous (e.g., fire brigades are mostly interested in detailed information about the location and severity of fires), and to solve the overall problem effectively, relevant information needs to reach the right agents.

Now, in this setting, agents can communicate such information using a limited set of channels, which represent different parts of the radio spectrum or even entirely different communication media. However, communication on these channels is severely constrained, both in the number of channels an individual agent can access simultaneously (due to technological or cognitive constraints) and in the available bandwidth on each channel. Moreover, these constraints typically preclude the use of explicit or centralised co-

[1]Another application is distributed sensor optimisation, where different types of sensors for tracking environmental phenomena are coordinated, whilst trying to share an underlying restricted communication infrastructure. Also, to reduce deployment costs, ubiquitous computing often utilises existing communication media, which are shared between applications, e.g., in a smart house.

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

ordination, due to time constraints and because coordination messages would themselves congest the channels. Examples of such coordination approaches include decentralised task allocation models such as Max-Sum [7], reward shaping [15] or auctions [9], which consider how to optimally allocate tasks and distribute disparate beliefs. However, these models often make heavily restrictive assumptions about the communication infrastructure and network topology, which do not hold in the settings we consider (such as assuming that neighbours in a network can communicate with each other at zero cost).

Another option is to solve the communication problem optimally, by casting it as a decentralised POMDP [14]. Here, both centralised and decentralised algorithms can be used, which analyse the future impact of sending a specific message on a given channel [4]. However, this approach is computationally intractable [5] and not appropriate for a scalable solution in systems with hundreds of agents.

To address this setting, we propose a new decentralised approach that converges to a good overall channel allocation, which, unlike previous work, considers the asymmetric information needs of agents. Specifically, it generally allocates agents that are interested in certain types of information to specific channels (such that, e.g., all fire brigades use a specific channel to share information about fires). However, in doing this, our approach considers subtle synergies between different agents, potentially placing several heterogeneous types of agents on the same channel if they share common information needs. It also takes into account bandwidth availability and team sizes to select appropriate channels, and it deals flexibly with low channel availability by sometimes allowing even agents with no common information needs to share the same channels. Crucially, this behaviour is based on principled calculations that predict the overall impact of a channel allocation on the global system utility, and it requires no a priori coordination or channel assignment. Furthermore, agents in our approach continuously monitor and learn the value of information locally to decide when to break from a given allocation, either temporarily (to pass on information to other types of agents) or permanently (to find a better overall allocation).

In addressing this problem, we make several key contributions. First, we present a novel decentralised channel allocation problem and show that solving it is inherently hard. Then, we develop our new decentralised algorithm that uses local learning and decision rules to solve it. Finally, we show empirically that this converges to a good solution that is typically within 85% of a hypothetical centralised optimal approach and outperforms a number of standard benchmarks (achieving a 6-fold improvement in some cases).

The remainder of this paper is structured as follows. In Section 2, we formalise the channel allocation problem we solve in this paper and then discuss how this problem could be solved in a centralised manner in Section 3. This discussion then informs our decentralised learning solution, which we outline in Section 4. In Section 5, we evaluate our approach experimentally and conclude in Section 6.

2. CHANNEL ALLOCATION PROBLEM

We begin by formally outlining the channel allocation problem (CAP) we address in this paper. Specifically, we consider a system that consists of heterogeneous *agents* that need to share *facts* about the world with each other. These facts constitute key items of information about the state of the world that help the agents in solving their joint problem. Generally, these facts could include sensor readings, locations of other agents and new tasks that have been discovered. Normally, we assume these facts are known with complete certainty (but they could also represent probabilistic information that improves the agents' decisions). Critically, not all facts are equally important and some may only be of interest to a subset of agents.

More formally, $\mathcal{A} = \{a_1, a_2, \ldots, a_A\}$ denotes the set of agents, each of which has a type given by $m : \mathcal{A} \rightarrow \mathcal{S}$, where $\mathcal{S} = \{s_1, s_2, \ldots, s_S\}$ is the set of all types (we will sometimes refer to the agents of a specific type as a *team*). Furthermore, $\mathcal{F} = \{f_1, f_2, \ldots\}$ is the set of all possible facts, with each fact having a deadline $d : \mathcal{F} \rightarrow \mathbb{N}_0^+$, which refers to a specific time step (we assume discrete time steps), after which the fact is no longer relevant to any agent. For example, such a deadline could represent when a building on fire burns out or when an injured civilian can no longer be saved.

Now, in our model, agents derive an explicit reward for knowing about facts. This describes the relative importance of those facts and captures the intrinsic value that knowledge of them adds to the problem-solving ability of an individual agent. We assume these rewards are known a priori, representing the domain knowledge of the system designer, although in practice, they could be probabilistic estimates rather than deterministic values. In more detail, $r : (\mathcal{F} \times \mathcal{S}) \rightarrow \mathbb{R}^+$ describes the reward per time step that an agent of a particular type generates for knowing a specific fact (this function is available to all agents, e.g., a fire brigade is aware of the value of a civilian position to an ambulance). Thus, the total reward an agent generates per time step is the sum of the rewards (for its specific type) of all facts that it knows about and that have not exceeded their deadlines yet. Similarly, the overall reward generated in the system is the sum of all the agents' individual rewards, and this is the key objective we seek to maximise in our work.

We assume that new facts are discovered gradually over time by individual agents at the beginning of each time step (according to some random process), and they immediately start to generate rewards. However, to maximise rewards, agents can communicate the facts they know about on a set of highly constrained communication channels, $\mathcal{C} = \{c_1, c_2, \ldots, c_C\}$. Importantly, due to cognitive or technological constraints, agents can only use a limited number of channels each time step, as given by $l : \mathcal{S} \rightarrow \mathbb{N}_0^+$. When an agent decides to use a channel, it can post a single fact on that channel and listen to the facts that other agents have posted on that channel. Each channel has a finite bandwidth, $c : \mathcal{C} \rightarrow \mathbb{N}_0^+$, the number of facts that the channel can carry in a single time step. Any surplus facts are discarded uniformly at random.

Given this, the choice for the agents is which channels to subscribe to and which facts to send on these channels. This is done in two steps — first, all agents simultaneously choose the subset of channels they wish to subscribe to (up to their channel limit l and, critically, without knowledge of the others' choices), they then discover who else subscribed to the chosen channels, and finally all agents choose simultaneously what facts, if any, to post on those channels. At the end of the time step, agents are informed of all facts that were successfully posted on their subscribed channels and these now become shared knowledge among all subscribers (starting to generate rewards from the next time step).

Now, making these two decisions — first, deciding which channels to subscribe to, and, second, which facts to post on the channels — comprise the key channel allocation problem we consider in the paper. Its difficulty lies in the fact that agents cannot coordinate a priori on their choice of channels (and may therefore subscribe to channels that will contain no interesting facts) and because they do not know what facts others may post to the channel (and may therefore post low-value facts that displace other more valuable ones). Our overall aim here is to solve this problem in a decentralised manner with agents that use only local knowledge. However, to gain a better insight into what a good solution looks

like, we will first discuss a (hypothetical) centralised solution in the next section.

3. CENTRALISED SOLUTION

In this section, we present two centralised solutions for the problem of maximising the overall reward during channel allocation — one is optimal, but tractable only in small problems and the other is locally optimal and scales to larger problems. Although centralised solutions are infeasible in realistic settings, defining these provides us with useful benchmarks to compare our approaches against. Specifically, they constitute upper bounds for the performance of decentralised approaches, as they assume full information and control over all agents. Furthermore, examining the optimal centralised solution will guide our decentralised approach.

Now, as solving the CAP to maximise long-term rewards is intractable due to the huge search space (as in a decentralised POMDP), we here (and in our decentralised approaches) concentrate on myopic solutions. These consider only the actions available in the current time step and do not plan ahead. In small environments, where a long-term optimal can be found, we observed that the myopic performs close to this, as there is often little benefit in planning ahead and it is usually optimal to send facts that immediately generate large rewards.

However, we first show the complexity of the optimal solution.

3.1 Complexity Result

We will show that even the myopic version of the centralised CAP is NP-complete.

DEFINITION 1 (MYOPIC CENTRALISED CAP (MCCAP)). *Given the model in Section 2 and the agents' current beliefs (here, we let B_i denote agent i's belief, i.e., the facts that it is aware of), are the agents able to generate a total reward of at least x, for a given $x \in \mathbb{R}$, assuming that no more facts are generated or communicated in future time steps?*

First, recall the well-known NP-complete Hitting Set problem:

DEFINITION 2 (HITTING SET). *Given a collection S' of subsets of some universe U and a constant $k \in \mathbb{N}$, is there a subset $X \subseteq U$, such that X intersects every element of S' and $|X| \leq k$?*

THEOREM 1. *MCCAP is NP-complete.*

PROOF. We need to show that MCCAP is both in NP and also NP-hard. The first is straight-forward — given a solution of which channels each agent should subscribe to and what facts to send, we can calculate the reward generated by all agents in subsequent time steps in linear time and verify this is at least x.

To show that MCCAP is NP-hard we show that any instance of HITTING SET can be reduced in polynomial time to an instance of MCCAP. To do this, define $\mathcal{F} = U$, such that every fact corresponds to an element in U and set its deadline to 1, $d(f_j) = 1$ (assuming the current time is 0). Now, for each element $I \in S'$, create an agent a_I with an empty belief ($B_I = \emptyset$) and a unique type for that agent, s_I, i.e., $m(a_I) = s_I$. Limit the agent to subscribing only to one channel, $l(s_I) = 1$. Then, for each element $i \in I$, set the reward for a_I knowing the corresponding fact to 1, while all other rewards for that agent are 0, i.e., $r(f_i, s_I) = 1$ if $i \in I$, otherwise $r(f_i, s_I) = 0$. Finally, define an agent a_0 with type s_0 that knows all facts (with belief $B_0 = \mathcal{F}$), but set all its rewards to 0 ($r(f_i, s_0) = 0$ for all i). Set its subscription limit to k, i.e., $l(s_0) = k$. Finally, there are k channels, $\mathcal{C} = \{c_1, c_2, \ldots, c_k\}$, each with a bandwidth of 1. We now set the value threshold to $x = |S'|$.

Given this transformation, which can be performed in polynomial time in the size of the original instance, the solution of the original HITTING SET instance is *yes*, if and only if the solution to the new MCCAP instance is also *yes*. This is because each of the agents corresponding to the elements in S' generates a reward of 1 if and only if a fact corresponding to an element of its associated set $I \in S'$ is placed on one of the k channels (no more than 1 can be generated by each agent due to their subscription limits and the channel bandwidth constraints). Since at most k facts can be placed on the channels, all of these agents generate a reward (i.e., the overall value generated is $|S'|$), if and only if there is a subset of $X \subseteq U$ with $|X| \leq k$, such that it intersects all elements in S'. □

Given this, it is trivial to show that the non-myopic version of CAP is NP-hard, as the same transformation can be applied. This indicates that a centralised myopic solution is generally intractable, especially for larger problems. For this reason, we next present two algorithms: an optimal one that can be used in smaller settings and a suboptimal one that scales to larger problems. This allows us to see how close our decentralised algorithms are to optimal in small problems, and also have a good idea in larger problems.

3.2 Centralised Myopic Optimal

In the centralised myopic optimal solution (*OPTIMAL*), at every timestep x, a centralised authority gathers information about the facts held by each agent and solves the MCCAP instance optimally before distributing the allocation and facts to send to each agent. We achieve this by formulating MCCAP as an integer linear program and solving it using ILOG CPLEX.

Whilst this algorithm provides the optimal myopic solution, it is far from scalable. Specifically, even when looking for a myopic solution, the search space is doubly exponential in the number of agents A and the number of channels that can be subscribed to l. Next, we address the scalability of this solution.

3.3 Centralised Myopic Local Search

In the centralised myopic local search solution (*CMLS*), the same centralised authority gathers all information and distributes an allocation as for *OPTIMAL*. However, rather than considering the entire search space as in *OPTIMAL*, we start with a random solution and then carry out local improvements.

More specifically, given an initial allocation where agents randomly send facts to channels, we apply the single best deviation that an agent could perform (by switching or removing a fact to send, switching a subscribed channel or both). Then, we iteratively apply such deviations until a local optimum is reached and no more deviations yield a higher utility. The resulting allocation is then selected as the overall solution.

Clearly, this algorithm is suboptimal since it only allows a finite step size in the improvement phase. If this was relaxed, then combinations of new partial allocations could be considered in order to find a global optimum. However, the reduction in the state space used here does result in a substantially more scalable algorithm which is still guaranteed to find a local optimum for the myopic case. That said, the algorithms presented in this section are centralised and also compute solutions over all agents, facts and channels. However, it is useful to consider what an optimal solution to the channel allocation problem looks like, in order to inform our decentralised approach, as we do next.

3.4 Analysis

By examining actual channel allocation decisions made by the centralised solution, we can make several general observations about

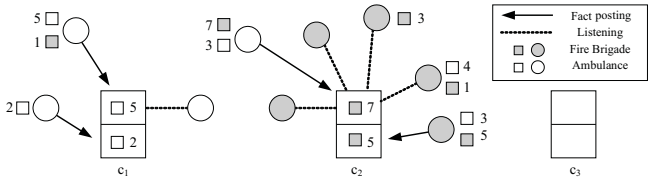

Figure 1: Example optimal channel allocation.

desirable behaviour in our problem domain. First, we note that generally it is beneficial for agents of the same type to use a dedicated channel for facts that they are interested in. That way, the overall reward generated by a posted fact is maximised as all interested agents receive it immediately, and there is typically no incentive for such a group to split up and use several channels. Conversely, it is usually not beneficial for agents of different types to share the same channel for their respective facts (unless the posted facts are relevant to both types), as this reduces the effective bandwidth of each team. Thus, our decentralised algorithm will dynamically designate channels to certain types and post only relevant facts to these.

Second, however, it is clearly suboptimal to restrict agents to posting and listening only to their designated channels. As each agent may find out about facts that are relevant to agents of other types, they need to occasionally communicate these by posting them to other channels (e.g., when an ambulance discovers a fire). However, this also means that the sending agent generates no additional reward by listening to more relevant facts on its own channel. For this reason, our algorithm will solve a local decision problem that explicitly balances the reward that the agent expects to generate by staying on its own channel with that generated by communicating a fact for a different agent type.

This general decision-making behaviour is reflected by the actions of the hypothetical centralised optimal strategy. To illustrate this, Figure 1 shows an optimal decision in an example scenario from RoboCupRescue. Note that this is the best allocation the agents could choose if they could perfectly coordinate their actions and had full knowledge of each others' available facts. Clearly, this is unrealistic, but we use it to illustrate what a good allocation looks like. In this example,[2] there are two types of agents, fire brigades and ambulances (shown as shaded and white circles, respectively). These can only subscribe to one channel each, but they have information about a variety of newly discovered facts that they can communicate to the others — these comprise information about both injured civilians (white squares) and fires (shaded squares). Ambulances are interested in injured civilians, while fire brigades derive rewards from finding out about fires (all rewards are noted in the figure). There are three channels with a bandwidth of two each.

In the optimal channel allocation shown here, we first note that channels are used exclusively for a particular type of fact (c_1 is used for facts about civilians and c_2 is used for fire), and most of the agents only listen to the channels for their respective type. The only exception to this is the ambulance in the centre (with one fire fact worth 7 and a civilian fact worth 3), which posts information about a particularly critical fire to the channel of the fire brigades. Although this means that this ambulance does not gain any reward itself from hearing about civilians, the value of its information to the fire brigades is high enough to warrant this decision. We note also that the third channel, c_3, is completely unused, because moving any of the agents and posting further facts there does not increase the overall rewards generated.

We will use these observations in the construction of a decentralised algorithm which avoids the complexity issues of the centralised approaches, but still performs well in comparison.

[2]An equivalent example from sensor networks sees two types of sensor: a ground based static sensor for detecting troops and UAVs for tracking faster-moving vehicles. These may detect targets for each other but would not take any direct action, instead communicating them to the respective agent. In ubiquitous computing, energy management devices need to communicate information about power usage with each other; however, the security system may need to track a threat and switch on high power devices over that same channel.

Algorithm 1 Basic Decentralised Learning Algorithm

1: **while** *true* **do**
2: Observe new facts and update fact beliefs.
3: Choose channels and facts to post using local beliefs.
4: Subscribe to channels and post facts.
5: Update channel beliefs.

4. DECENTRALISED SOLUTION

In this section, we develop a decentralised learning algorithm to solve this problem in realistic settings. With this, agents use only local information and previous observations to decide which fact to send on which channel. More specifically, we assume agents know only the facts they have discovered themselves and those they have heard on channels, but they do not know what facts are known by others. They also know the channels and their characteristics, and they know what other agents are present in the system and their types (although our algorithm can easily be extended to discover this at run-time).

As we argued in Section 1, this problem can be solved optimally by casting it as a decentralised POMDP. However, that methodology is intractable for even small agent teams. Further to this, that solution still requires a complex coordination mechanism (or communication) to execute the decentralised policies and full knowledge of the fact generating function in order to obtain the policies in the first place. Consequently, we look towards reinforcement learning of the underlying problem, together with a simple coordination mechanism, as an efficient means to finding a decentralised solution. Now, this section will first describe the overall approach we take in our solution, followed by its individual components and finally the overall algorithm is presented.

4.1 High-Level Approach

Our problem requires that the agents find a common agreement on how to divide the channels into their respective types, and then, agents need to decide when they should send facts (or listen) to their own channel and when they should communicate facts to other types. We address these two challenges separately, focusing first on the channel division and then on the fact communication problem.

Before considering the details, Algorithm 1 shows the high-level approach used in our decentralised solution. To make good decisions, our algorithm maintains beliefs about the system, including *fact* beliefs about the characteristics of the fact generation process in the system, and *channel* beliefs, which include historical frequencies over what types of agents have been observed on the various channels and some information about the value of facts that have been posted to the channels in the past. These beliefs are updated regularly at every time step based on new facts that are observed (line 2) as well as the agents and facts that are seen on channels (line 5).

The key decision about what channels to subscribe to and what facts to post is made in line 3. This is clearly critical, but it is important here to note that this can depend only on the agent's local beliefs, without knowing the beliefs of other agents or their future

decisions. However, these local beliefs (in particular the channel beliefs) are shaped by the past actions of other agents in the system. As such, our algorithm constantly *learns* and *adapts* to the actions of others, allowing it to respond better in the future. This is very similar to fictitious play [6], which has been used successfully in more competitive settings, such as congestions games. However, although our setting is cooperative in nature, fictitious play has several desirable properties — it is known to converge to Nash equilibria, and it represents a simple, tractable learning technique (rather than resorting to complex optimal approaches, such as POMDPs). This makes it suitable for our problem.

As part of making the decision about what channels to subscribe to and what facts to post, an agent needs to first know what teams are assigned to what channels. We will consider this question in Section 4.2. Second, given that it has this channel allocation, it then needs to decide what facts to post (if any) and on what channels. We treat this problem in Section 4.3. Finally, we present an overall algorithm in Section 4.4.

4.2 Channel Division

Dividing the channels effectively between the agent types is a central part of our algorithm. The aim of this is to find a function $division : \mathcal{S} \rightarrow \mathbb{P}(\mathcal{C})$, which maps each agent type to the set of channels used by that type (note these can overlap). All agents need to agree on this and the mapping needs to also consider the bandwidth of channels (it may be more beneficial to share a single channel with a large bandwidth between several types than have one type use a very small channel).

To quickly find a consensus for the *division* function, the agents treat their own type and other types differently. Specifically, they assign other types simply to the channels that have the highest participation by those agents, based on previous observations made when subscribing to channels, up to the channel limit of that type. For its own channel assignment, each type first designates a leader, who decides on the best channels for its type.[3]

The leader's decision is based on probabilistic knowledge about arrivals of new facts (its fact beliefs), which it uses to calculate the expected utility of choosing a particular channel. In more detail, we denote by $\overline{u}(\mathcal{S}', s_i)$, with $\mathcal{S}' \subseteq \mathcal{S}$ and $s_i \in \mathcal{S}$, the total reward an agent of type s_i expects to gain from a randomly chosen fact, given that the fact has a non-zero reward for at least one of the members of $\mathcal{S}'$. With this, the expected utility generated by a particular set of agent types $\mathcal{S}'$ using channel c_i is estimated as:

$$\overline{r}(\mathcal{S}', c_i) = c(c_i) \cdot \sum_{s_j \in \mathcal{S}'} \overline{u}(\mathcal{S}', s_j) \cdot \left(\text{size}(s_j) - \frac{\text{size}(s_j)}{\text{total}(\mathcal{S}')} \right), \quad (1)$$

where $\text{size}(s_j)$ is the total number of agents of type s_j and $\text{total}(S') = \sum_{s_k \in \mathcal{S}'} \text{size}(s_k)$. As such, it is the expected reward all agents on channel c_i expect to generate through communication per time step, given that the channel is always fully utilised for sending new facts, that agents choose to send facts randomly from among those that are relevant to at least one other agent on the channel and that all agents in the team subscribe to the channel. These are simplifying assumptions, but they serve to allow an agent to quickly compare the relative merit of choosing a particular channel over another. In doing so, the calculation considers the relative sizes of a team on a channel (as all agents but the sender benefit from a posted fact), the bandwidth of the channel, as well as taking into account the relative frequencies of facts, e.g., when facts for a particular type are more frequent than those for another type (this is intrinsically part of $\overline{u}(\mathcal{S}', s_j)$).

Given this, the leader then uses Equation 1 to evaluate the overall impact of choosing a particular channel for its type (keeping the allocations of other types constant, as given by the *division* function), and greedily chooses the channels with the highest respective utilities, up to the channel limit for its type. Its other team members, in turn, also adopt these chosen channels (if necessary by searching for the channels the leader subscribes to, as will be explained later). Assigning a single leader agent in this way allows the team to converge quickly to a set of channels, which can be easily recomputed at any time, e.g., to take into account dynamically changing reward distributions, team sizes or available channels.

4.3 Fact Communication

So far, we have described how agents reach consensus about a consistent channel division between the agent types. However, the agents only derive value from actually sending facts to channels. As we argued in Section 3.4, agents will generally send facts only on their own channels, unless they have a particularly valuable fact for a different type. To determine which facts to send to which channels, we use a decision-theoretic approach and estimate the expected total reward that an agent a_l would contribute to the system by sending a given fact f_i at time t to channel c_j:

$$\mathbb{E}(r_{\text{send}}(a_l, f_i, t, c_j)) = \sum_{s_k \in \text{onChannel}(c_j)} \text{size}_{-l}(s_k) \cdot (r(f_i, s_k) \cdot (d(f_i) - t) - (1 - \text{total}_{-l}(c_j)^{-1}) \cdot \overline{s}(s_k)), \quad (2)$$

where $\text{onChannel}(c_j)$ is the set of agent types that are mapped to channel c_j by the *division* function, $\text{size}_{-l}(s_k)$ is the number of agents of type s_k excluding a_l, $\text{total}_{-l}(c_j)$ is the total number of agents of the types given by $\text{onChannel}(c_j)$ excluding a_l and $\overline{s}(s_k)$ is the reward that a single bandwidth unit on a channel for the given type s_k is expected to generate (again, based on previous observations, and, to avoid bias, excluding the facts the agent previously posted itself). Thus, this equation is positive if the fact improves on what is otherwise expected to be posted on the bandwidth unit it would occupy.

As this assumes that posted facts are new to all agents on the channel, we allow agents to only post facts they themselves have discovered and that have not been sent successfully to this channel (or other channels occupied by the same types) before — this greatly simplifies the decision problem, as agents do not need to keep track of the belief states of other agents.

Apart from $\mathbb{E}(r_{\text{send}}(f_i, t, c_j))$, agent a_l expects to benefit from listening when posting to a particular channel c_j (if this is one of its own channels). We denote this expected benefit as $\mathbb{E}(r_{\text{listen}}(a_l, c_j))$ and note that it is 0 when the channel is not one of its own (i.e., $m(a_l) \notin \text{onChannel}(c_j)$), otherwise it is $\overline{s}(m(a_l)) \cdot (l(c_j) - 1)$.

Thus, we can obtain an overall valuation for an agent a_l of sending a fact f_i at time t to channel c_j:

$$\mathbb{E}(r_{\text{overall}}(a_l, f_i, t, c_j) = \mathbb{E}(r_{\text{send}}(a_l, f_i, t, c_j)) + \mathbb{E}(r_{\text{listen}}(a_l, c_j)) \quad (3)$$

Using Equation 3, the agent can now greedily choose the best facts to post, or stay silent and listen to its own channels when this is more beneficial (the utility of this is $\overline{s}(m(a_l)) \cdot l(c_j)$ — slightly higher than $\mathbb{E}(r_{\text{listen}}(a_l, c_j))$ because the agent potentially hears one additional fact).

[3] This can be achieved without explicit communication, using, e.g., unique identifiers, such as IP or MAC addresses of the agents encountered so far. Throughout this section, we will assume that an agent has a priori knowledge of all other team members and their identifiers, but even when this is not available, a simple adaptive leader election protocol based only on team members observed so far could be designed. We leave this, and the re-assignment of missing leaders, to future work.

4.4 Overall Algorithm

Our approaches for choosing a consistent channel allocation and deciding what facts to post on what channels constitute the core of our decentralised algorithm. In addition to these decision-making procedures, however, it is important for the agents to sometimes choose actions that allow them to better update their beliefs, rather than simply maximise their expected rewards through posting facts. This is because our algorithm relies on some statistical knowledge that is learnt at run-time, but when this is inaccurate, an agent may lack the incentive to further subscribe to the affected channels and update its knowledge. This exploration/exploitation tradeoff is common in learning problems, and we adopt an approach that is often employed there: ε-greedy. More specifically, with a small probability[4] $\varepsilon = 0.01$, agents *randomly* pick a subset of channels to subscribe to, rather than considering all possible channels. This ensures both that the *division* function is updated when other teams move channels, and, similarly, that $\bar{s}$ is learnt.

Given this, Algorithm 2 summarises the overall decision-making mechanism each agent follows. In brief, each agent starts a time step by observing its environment and gathering new facts about the world (line 5). These are used to update the agent's local $\bar{u}$ function (line 6). Next, the agent calls a procedure depending on whether it is a leader or not (line 9).

As a leader, it first re-evaluates the current channel division with a small probability, $\varphi = 0.05$. Here, the FINDBESTDIVISION procedure (line 16) finds the best channels to use for the agent's own type, given the current *division* for all other types (as described previously), and returns this only if it is at least $\delta = 5\%$ better than the current choice. Note that φ and δ are included here to prevent the agent from switching channel divisions too quickly. If no exploration takes place, and the leader has not subscribed to its own channels for at least $maxAbsence = 5$ time steps, it is forced to only consider its own type's channels (line 18), which allows other agents to easily detect when the leader has chosen to change channels (hence, a longer absence indicates that the leader has changed channels). If this is not the case, the leader chooses random channels with probability ε, where RANDOMCHANNELS returns a set of random channels of size equal to the agent's subscription limit (line 20).

As a follower, if the leader has not been observed for more than *maxAbsence* time steps, the agent starts searching, where SEARCHCHANNELS returns channels that have not been visited recently (line 23). Otherwise, the follower also chooses random channels with probability ε (line 25).

Next, the agent chooses the best facts to post to the eligible set of *channels* (line 10). This is done using the procedure described in the previous section. The agent then follows this, first subscribing to channels and then posting facts (lines 11 and 12). Finally, it uses information observed on its subscribed channels (i.e., participating agents and posted facts) to update a number of statistics (line 13).

Concluding this section, as we noted earlier, our algorithm can be seen as a form of fictitious play, i.e., each agent effectively plays the best response to the other agents' actions (as learnt by the *division* and $\bar{s}$ functions). This allows our strategy to adapt to a dynamic environment. For example, when only a few, low-value facts are sent to a particular channel (as indicated by a low average reward per bandwidth, $\bar{s}$), agents decrease their threshold for sending facts, but when congestion is high, only very valuable facts are sent.

[4]We stress our algorithm does not depend on the exact choice of this parameter and others mentioned in this section. However, we list the parameters used in our implementation for completeness.

Algorithm 2 Decentralised CAP Algorithm (DecCAP)

```
1:  a_i ← myself                                              ▷ Agent executing this
2:  leaderAbsent ← ∞                                          ▷ When was the leader last seen?
3:  ownFacts ← ∅                                              ▷ Facts discovered by this agent
4:  while true do                                             ▷ For each time step
5:      newFacts ← OBSERVEENVIRONMENT()                       ▷ Gather new facts
6:      ū ← UPDATEFACTSTATS(newFacts)                         ▷ Update fact statistics
7:      ownFacts ← ownFacts ∪ newFacts
8:      channels ← C                                          ▷ Channels to consider
9:      LEADERLOGIC() / FOLLOWERLOGIC()                       ▷ Depends on whether leader
10:     decision ← CHOOSEBESTFACTS(channels, ownFacts)        ▷ Facts to post
11:     SUBSCRIBE(decision)                                   ▷ Subscribe to chosen channels
12:     heardFacts ← POSTFACTS(decision)                      ▷ Post facts
13:     division, leaderAbsent, s̄ ← UPDATECHANNELSTATS(heardFacts)
14: procedure LEADERLOGIC( )                                  ▷ Leader's decision-making logic
15:     if Random(0,1) ≤ φ then                               ▷ Explore new channel division?
16:         division ← FINDBESTDIVISION(division, δ)
17:     else if leaderAbsent ≥ maxAbsence then                ▷ Stick to own channels?
18:         channels ← division(m(a_i))
19:     else if Random(0,1) ≤ ε then                          ▷ Random exploration?
20:         channels ← RANDOMCHANNELS()
21: procedure FOLLOWERLOGIC( )                                ▷ Follower's decision-making logic
22:     if leaderAbsent > maxAbsence then                     ▷ Search for leader?
23:         channels ← SEARCHCHANNELS()
24:     else if Random(0,1) ≤ ε then                          ▷ Random exploration?
25:         channels ← RANDOMCHANNELS()
```

5. EMPIRICAL RESULTS

In this section, we comprehensively evaluate the performance of our algorithm, *DecCAP*, against the interesting space of problems captured by our model. To measure its performance, we compare it to a number of benchmarks:

- *OPTIMAL/CMLS* is the centralised optimal[5] algorithm from Section 3. As such, it is clearly not realistic in practice, but rather serves as an *upper bound*.
- *RANDOM* subscribes to random channels placing a single random fact from its current beliefs on each channel.
- *BEST-FACT* subscribes to random channels and then places the fact that promises to generate the highest reward on each channel. This is based on the agent's local beliefs about what facts are already known by others.
- *EPSILON-GREEDY(ε)* generally subscribes to the channel that has generated the highest overall average reward for that agent (based on past observations), but with probability ε, it picks a random channel instead for exploration. When subscribed, it behaves as the *BEST-FACT* strategy. As such, it represents a common solution approach to work that models the channel allocation problem as a multi-armed bandit [2].

In the following, we restrict our analysis to a problem setting from the recent RoboCupRescue competition (since this was created to be a taxing problem with all of the challenges discussed earlier) and then explore some interesting parameters within this domain. Specifically, we first consider the standard setting from RoboCupRescue with three types of agents: ambulances, police and fire brigades. These are interested in three types of information: new civilians, road blocks and fires. We assume facts are discovered randomly, such that each agent discovers one new fact on average every four time steps (using a Poisson distribution), each fact is of a random type, has a reward drawn uniformly at random from $[0,1]$, and a deadline drawn uniformly at random from

[5]Due to the complexity of this, we use myopic optimality here, and, for more than 10 agents, we replace the *OPTIMAL* by the greedy *CMLS*, which achieves the *same* performance as *OPTIMAL* on the smaller settings.

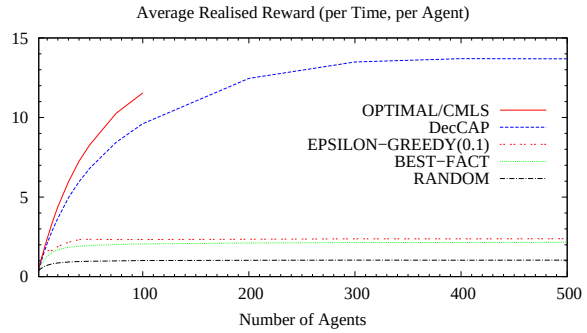

Figure 2: Performance as the number of agents is increased.

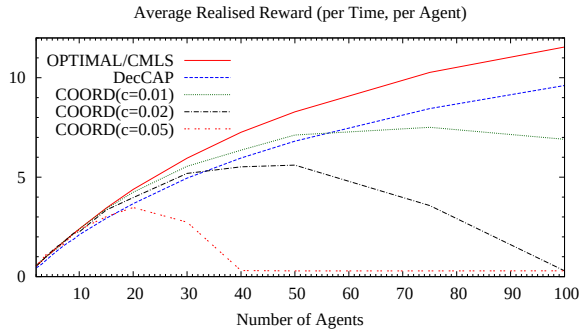

Figure 3: Performance with explicit coordination.

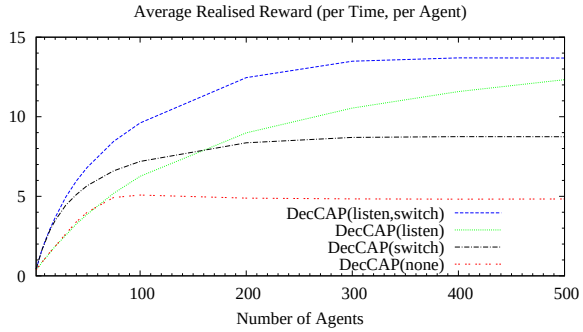

Figure 4: Performance of *DecCAP* variants.

$\{2,3,\ldots,10\}$ (to represent buildings that burn out or civilians that die). We also consider a highly constrained communication infrastructure with five channels that can contain only two facts each.

Given this setting, we are first interested in establishing the relative performance of our approach to the benchmarks and to evaluate whether it scales to large systems.

5.1 Basic Benchmarks

To first establish how *DecCAP* performs in increasingly large settings, Figure 2 shows the results[6] with increasing numbers of agents. It is clear here that *DecCAP* significantly outperforms the two baseline benchmarks, *RANDOM* and *BEST-FACT*, improving on them by up to 600%, as it is able to select a good channel allocation and communicate the best facts to the right agents. The learning approach *EPSILON-GREEDY* is also outperformed by *DecCAP* (we only plot $\varepsilon = 0.1$ here as one of the best performing parameter choices). This is because the former quickly converges to a local optimum, in which all agents communicate on the same channel.

Finally, we observe that *DecCAP* also consistently achieves 85% or more of *OPTIMAL* and *CMLS*. This is a significant result, given that those assume full information and complete control over all agents' actions. Finally, we note that we could only run *CMLS* up to 100 agents, beyond which it became exceedingly slow, while *DecCAP* still made fast decisions in less than 0.5 ms per agent and time step with 1000 agents, using a Java implementation on an Intel 2.2GHz laptop (not shown on the graph for readability).

5.2 Explicit Coordination

While we argue that the *OPTIMAL/CMLS* centralised approach is unrealistic in most settings, it could be achieved in practice through the use of explicit coordination between agents. In order to do this, agents need to exchange coordination messages, which places an additional burden on the communication infrastructure and thereby reduces the effective bandwidth that can be used to exchange facts.

To capture this class of coordination approaches, exemplified by the Max-Sum algorithm [7] or auctions for resource allocation [9], we define a new benchmark, *COORD(c)*, which behaves as the centralised approach, but incurs a small communication cost of c per agent in the system. Here, c can be seen as the size of a single coordination message that each agent needs to send in order to achieve a fully coordinated response. More specifically, the cost c is an expected loss of bandwidth on every channel at each time step per agent, such that $c = 0.01$ in a system of 50 agents implies that one fact less can be posted on each channel every other time step.

[6]The reward is the average achieved per time step during steps 2000 – 2050, to give our learning algorithm time to converge in settings with hundreds of agents. For statistical significance (at the $t < 0.01$ level), we sample each point 500 times.

The results for representative costs 0.05, 0.02, 0.01 and 0 (the latter being equivalent to *OPTIMAL/CMLS*) are shown in Figure 3. These demonstrate that approaches using explicit communication perform well in smaller settings, but as soon as the number of agents grows, the communication costs for coordination become non-negligible, and they begin to be outperformed by *DecCAP*.

5.3 Strategy Components

To highlight the benefits of the various components of *DecCAP*, we here briefly evaluate a number of variants of *DecCAP*. As described in Section 4.3, *DecCAP* sometimes forces agents to *listen* to their own channels, i.e., stay silent when this appears more beneficial, while other times agents actively *switch* to channels used by other teams to post particularly valuable facts. We here examine the benefits of these behaviours, using *DecCAP(listen,switch)* to denote a strategy that implements both behaviours, while *DecCAP(listen)* denotes one that implements only the listening, and so on.

The results are shown in Figure 4. This clearly highlights that both the listening and the switching behaviours are key to achieving a high overall performance. Interestingly, the benefit of switching is more pronounced in settings with fewer agents, while listening becomes more important in settings with more agents. Intuitively, this is because fewer facts are discovered in the smaller settings, and so agents benefit from disseminating information to other teams, to fully utilise the available bandwidth. On the other hand, when there are many agents and channels are typically congested, agents gain more from simply listening to the high-value facts posted to their own channels.

5.4 Convergence

Since our strategy relies on learning channel allocations and fact distributions, it takes some time to converge to a good solution. To evaluate how long this takes in practice, Figure 5 shows the performance of *DecCAP* over time, for a small problem with 30 agents

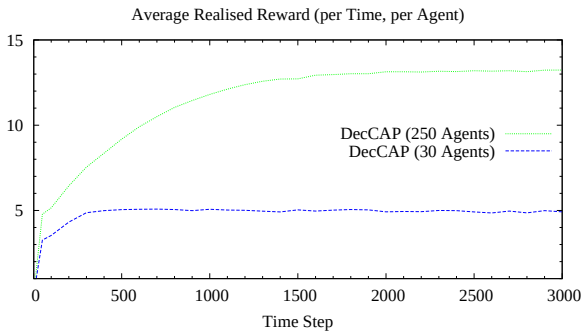

Figure 5: Convergence of *DecCAP* learning.

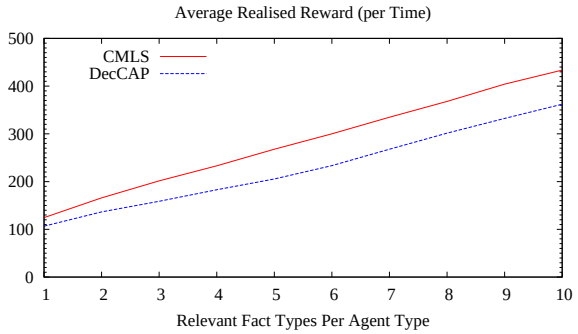

Figure 6: Performance as heterogeneity increases.

and a larger problem with 250 agents. This shows that the strategy converges in a reasonable amount of time. While the maximum in the larger setting is reached after around 2000 time steps, it already achieves 50% of the maximum after 200–300 time steps. In the smaller setting, the maximum is reached more quickly, after around 300 time steps.

5.5 Heterogeneity and Team Synergies

Finally, we consider a more heterogeneous setting where some facts are of interest to multiple types of agents (these synergies also exist in some RoboCupRescue strategies, e.g., when an ambulance uses information about fires to identify particularly critical civilians). We also increase the heterogeneity of the setting under consideration. Thus, we now assume there are five agent types, and each is interested in n out of 10 different types of facts, where we vary n from 1 to 10. As n increases, so do the synergies between teams (more are likely to be interested in the same facts). We also randomly vary the bandwidth of channels, ranging from 0 to 5, we consider 30 agents, and randomly vary the size of agent teams. We choose these parameters to test a more heterogeneous environment where the best performance is not necessarily achieved by allocating each team to a single channel.

The results are given in Figure 6. First, this shows an overall increase in rewards, as each generated fact is increasingly likely to benefit several agent types. Here, *DecCAP* achieves a performance that is within 80–85% of *CMLS*. There is a small drop in reward (relative to *CMLS*) around $n = 5$. This is because *CMLS* can benefit here from re-assigning agents instantaneously between time steps, depending on the overlap of currently known facts. Clearly, such a strategy is not feasible without a central coordinator with full information about all agents (as assumed for *CMLS*). Despite this, *DecCAP* achieves 80% of the centralised near-optimal *CMLS*, indicating that it chooses a suitable channel allocation that performs well in the long run. In more detail, examining the decisions made by *DecCAP* shows that it initially (for $n = 1$) separates different agent types into different channels, but, as fact synergies increase, they increasingly converge to shared channels. This confirms our algorithm flexibly adapts to the problem parameters and communication constraints.

6. CONCLUSIONS

In this paper, we presented an algorithm for channel allocation and information sharing in cooperative agent teams with a highly constrained communication medium. This setting requires the agents not only to reason about who to communicate with and about what, but also how to allocate a restricted communication resource. We present a tractable and fully decentralised learning approach which uses reinforcement learning ideas to learn a channel allocation and then principled decision-theoretic approaches to evaluate the utility of sending pieces of information to others, or even to break from the previously adopted channel allocation, which cannot be done using existing techniques, e.g., in cognitive radio. We compared our approach to benchmarks and showed that it converges quickly to a solution that typically achieves 85% of a centralised optimal strategy.

With this established, we intend to explore the relationship with existing congestion game models including investigating if a finite improvement policy exists, which would allow simple local techniques to converge to Nash equilibria here. This is important since it would allow us to bound our solution quality (which is only empirically demonstrated at present).

Acknowledgement This work was funded by the ALADDIN and ORCHID projects (`www.aladdinproject.org` and `www.orchid.ac.uk`).

7. REFERENCES

[1] I. F. Akyildiz, W.-Y. Lee, M. C. Vuran, and S. Mohanty. Next generation/dynamic spectrum access/cognitive radio wireless networks: A survey. *Computer Networks*, 50(13):2127–2159, 2006.

[2] A. Alaya-Feki, E. Moulines, and A. LeCornec. Dynamic spectrum access with non-stationary multi-armed bandit. In *SPAWC 2008*, pages 416–420, 2008.

[3] S. Basagni, M. Conti, S. Giordano, and I. Stojmenovic. *Mobile Ad Hoc Networking*. Wiley-Blackwell, 2004.

[4] R. Becker, A. Carlin, V. Lesser, and S. Zilberstein. Analyzing Myopic Approaches for Multi-Agent Communication. *Computational Intelligence*, 25(1):31–50, February 2009.

[5] D. S. Bernstein, S. Zilberstein, and N. Immerman. The complexity of decentralized control of markov decision processes. In *UAI 2000*, pages 32–37, Stanford, USA, 2000.

[6] G. W. Brown. Iterative solution of games by fictitious play. In T. C. Koopmans, editor, *Activity Analysis of Production and Allocation*, pages 374–376. New York, 1951.

[7] A. Farinelli, A. Rogers, A. Petcu, and N. R. Jennings. Decentralised coordination of low-power embedded devices using the max-sum algorithm. In *Proc. AAMAS-08*, pages 639–646, May 2008.

[8] K. Hiroaki. Robocup rescue: A grand challenge for multi-agent systems. In *Proc. ICMAS 2000*, pages 5–12, Boston, MA, USA, 2000.

[9] S. Koenig, P. Keskinocak, and C. Tovey. Progress on agent coordination with cooperative auctions. In *Proc. AAAI-10*, pages 1713–1717, 2010.

[10] M. Liu, S. H. A. Ahmad, and Y. Wu. Congestion games with resource reuse and applications in spectrum sharing. In *Proc. GameNets'09*, pages 171–179, 2009.

[11] U. Mir, L. Merghem-Boulahia, and D. Gaiti. A cooperative multiagent based spectrum sharing. *Proc. AICT 2010*, pages 124–130, 2010.

[12] A. Motamedi and A. Baha. Optimal channel selection for spectrum-agile low-power wireless packet switched networks in unlicensed band. *EURASIP Journal on Wireless Communications and Networking*, 2008.

[13] N. Pavlidou, A.J. Han Vinck, J. Yazdani, and B. Honary. Power line communications: state of the art and future trends. *IEEE Commun. Mag.*, 41(4):34–40, 2003.

[14] L. Peshkin, K. Kim, N. Meuleau, and L. Kaelbling. Learning to cooperate via policy search. In *Proc. UAI 2000*, pages 307–314, San Francisco, USA, 2000.

[15] S. A. Williamson, E. H. Gerding, and N. R. Jennings. Reward shaping for valuing communications during multi-agent coordination. In *Proc. AAMAS–09*, pages 641–648, Budapest, Hungary, 2009.