Decentralised Dynamic Task Allocation Using Overlapping Potential Games

ARCHIE C. CHAPMAN[1], ROSA ANNA MICILLO[2], RAMACHANDRA KOTA[1] & NICHOLAS R. JENNINGS[1]

[1]*School of Electronics and Computer Science, University of Southampton, SO17 1BJ, UK*
[2]*Department of Information Engineering, Second University of Naples, Aversa (CE), Italy*
Email: {acc,rck05r,nrj}@ecs.soton.ac.uk, rosaanna.micillo@unina2.it

This paper reports on a novel decentralised technique for planning agent schedules in dynamic task allocation problems. Specifically, we use a stochastic game formulation of these problems in which tasks have varying hard deadlines and processing requirements. We then introduce a new technique for approximating this game using a series of static *potential games*, before detailing a decentralised method for solving the approximating games that uses the distributed stochastic algorithm. Finally, we discuss an implementation of our approach to a task allocation problem in the RoboCup Rescue disaster management simulator. The results show that our technique performs comparably to a centralised task scheduler (within 6% on average), and also, unlike its centralised counterpart, it is robust to restrictions on the agents' communication and observation ranges.

Received 29 August 2009; revised 20 November 2009

1. INTRODUCTION

The design and control of large, distributed systems is a major challenge in computer science research. It is important precisely because the application domains of such systems are so broad and ubiquitous: they arise in disaster management and evacuation scenarios (e.g. [1, 2]), wide–area surveillance and distributed sensor network management (e.g. [3, 4]), industrial task allocation and scheduling problems (e.g. [5, 6]), and in the control and management of congestion within air, road, rail, and information networks (e.g. [7]), among many other places. Now, scenarios in these domains are inevitably characterised by distributed information and costly and/or noisy communication, which limits the applicability of traditional centralised control approaches, and by noisy observations and inherent dynamism, which necessitate flexible, agile and robust decision–making. Furthermore, in certain settings, the timeliness of decisions is of paramount importance.

One broadly investigated approach to tackling these general challenges is known as *multi–agent systems* (MAS) — a paradigm that views such systems as a collection of autonomous, interacting, decision–making entities called *agents* [8]. In this framework, each agent usually controls a small subset of the variables in the system, and the collective actions of the agents are then engineered to produce desirable system–wide behaviour. In this setting, by control we mean the problem of finding and implementing configurations of the system's variables in order to achieve certain global goals. Specifically, using a MAS, control of the system's variables is handed to several agents, that reason about the variables under their control using distributed (sometimes local, sometimes private) knowledge, and by so doing, spread the system's computational burden and remove the brittleness and risk associated with a single point of failure or a computational bottleneck [9]. Within the MAS paradigm, agents are usually cast as one of two types — they are either *cooperative* or they are *self–interested* (or *noncooperative*). In the former case, agents explicitly share the same goals, which are typically the system–wide goals, and are designed to coordinate their actions to achieve these ends. However, achieving these goals typically requires that the agents share a common view of the world in order to compute the values of different variable configurations, making them prone to similar risks as centralised control mechanisms. In contrast, in the noncooperative case, agents possess their own internal motivations and act on their own local information, which removes these risks. Now, although these agents are self–interested, they are not necessarily adversarial; they simply have *private utility functions* that guide their reasoning and decision–making processes. To date, self–interested agents are typically the focus of open systems, such as auctions and markets (e.g. [10, 11]), trust and reputation systems (e.g. [12, 13]), or work–flow provisioning (e.g. [14]). In particular, agents in these setting are used to automate the decision making of stakeholders that have divergent interests, and these interests are outside the control of the system designer.

However, self–interested agents can also be successfully applied to the design of control mechanisms for large distributed problems with a single stakeholder or global

objective. This is because in many such design problems, the private motivations of the agents can be constructed in such a way that they are aligned with the global system objectives (as in [15, 16, 17]). In particular, in this work we use such self–interested agents to construct a MAS–based control mechanism for the dynamic task allocation and scheduling problem at hand, because we believe that self–interested agents can provide the degree of robustness and flexibility needed in large distributed applications. Specifically, agents are designed to act on their local knowledge of the state of the world and the actions of their neighbours such that improvements to their own payoffs only increase the global solution quality. As a consequence, by following their own interests, they coordinate their actions to achieve a high quality system–wide outcome. The resulting control mechanism is robust and flexible in the face of changes in aspects of the underlying control problem or the resources available to implement the control mechanism precisely because the agents only require local information to act: at no point must all the information regarding the world's state or the agents' actions be available to a single decision–making entity. As such, the time, communication and computational costs of recomputing a solution to the control problem at hand can be substantially reduced, thus meeting the requirements of a control mechanism for a large distributed system: namely, flexibility and robustness to component changes, limited use of communication, and timeliness generation of solutions.

Given this background, the particular scheduling problems that we address in this work consist of a set of agents that negotiate to assign themselves to, and execute, a dynamic stream of tasks, without the aid of a centralised manager. Each task has a hard deadline and a particular processing requirement (e.g. in a disaster response scenario, a casualty must be taken to hospital by 10:30am and this will take 30 minutes). In particular, the task set is dynamic because it is gradually revealed over time. This, in turn, means we require a practical mechanism that can respond to such changes in a timely fashion. Now, an agent can attend any task — there are no subtasks or specialisations — but it can act on only one task at a time and it performs this task at a fixed rate of processing. As a consequence, some of the tasks cannot be completed by an individual agent before their deadlines, and so must be tackled by a team working together.[1] For example, two fire crews working together to extinguish a burning building have a greater chance of success than one, or act to reduce the time it takes to extinguish the fire. However, this induces a problem of scarcity among competing tasks for limited agents. As the full set of tasks is not known at the outset, an agent has to continually negotiate with other agents over the sequence of tasks to execute, so that all currently known tasks are considered in generating a solution. This negotiation is carried out in such a way that the agents achieve their joint goal of completing as many as possible and in the least amount of time, and may include agents de–committing from tasks in order to maximise the total number when new tasks are revealed. Furthermore, the execution of some tasks precludes the execution of others, and decisions made at a given time may affect the structure of the problem facing the agents in the future. For example, the decision to rescue one trapped casualty may mean that a second cannot be saved, but may not prevent the rescue of a third later, or the decision to extinguish a particular fire may remove the risk of nearby buildings catching alight. Consequently, agents must consider the future effects of their current actions during their negotiations.

To jointly satisfy all three of the requirements of agility and robustness, timeliness, and high–quality solutions, we pursue a game theory–based approach, in which planning is achieved via negotiation between agents. Using a negotiation protocol allows us to spread the computational burden of solving the problem across the agents in the system, which adds to the control mechanism's robustness and the timeliness of its solutions. This approach is derived by integrating two principled approaches to the separate problems of, first, reducing the original dynamic problem to a tractable version via an appropriate approximation (in order to improve the timeliness of solutions and the communication requirements of the control mechanism), and second, incentivising the agents to behave in a manner consistent with the global objectives of the system, without requiring full global knowledge of the actions of all other agents or the complete set of tasks.

In more detail, we tackle the problem of designing a control mechanism for distributed dynamic task allocation by first defining a global utility function, which is constructed as if we were defining it for a centralised Markov decision problem (MDP). This is the global target function we wish to maximise, and its approximation is discussed later. Next, we address the requirement of robustness in our control mechanism. We satisfy this requirement by giving control over the variables of the problem — the resources used to complete the tasks — to a set of agents, each of which makes its own independent choices. As a consequence, we use game theory to analyse their joint behaviour, because it is the natural way to model the interaction of such autonomous agents. Specifically, if the variables of a MDP are controlled by a set of autonomous agents, then we are in the realms of *stochastic games* [19]. In this type of game, self–interested agents play a multi–stage game, in which the stage game varies probabilistically at each time–step as a function of the state and actions taken in the previous time–step. Here, each stage game corresponds to a particular set of tasks to be completed at that time and the status of the agents in the system (their locations, commitments, etc). Each agent has its own private utility function which it aims to maximise. However, as discussed above, we can define these such that, for any unilateral switch in strategy, an agent's change in payoff is equal to the change in the global utility. Consequently, the global maximum is a pure strategy Nash equilibrium (i.e. it is a stable solution to the game). In this way, selfish agents can be used to solve an inherently cooperative problem, because

[1] Note that additional agents only help complete a task — they do not hinder each other. This uncomplicated task model means we do not need to adopt a sophisticated planning language to express the agents' interactions (cf. [18], for example), because the value of all actions can be easily expressed in terms of utilities.

their self–interest drives them towards solutions with higher global utility.

Although our problem can be cast elegantly in such a model, it is not practical to solve it at real–world scales, because agents have to consider the utilities of the future tasks present in all possible future states during the negotiations. In particular, any multi–stage game with stochastic state transitions is NEXP-complete [20], due to the factorial growth in the number of state–action–transition combinations. Thus, we must address the requirement of the timeliness of the solutions generated by the control mechanism. To do this, we show that the stochastic game can be approximated using a sequence of finite length multi–stage games of complete information. In this context, we approximate the global utility function with a receding horizon version of the same function. This approximation incurs two penalties: (i) for truncating the length of the decision window and therefore ignoring payoffs from future states, and (ii) for ignoring the possible future changes in the state of the world, other than those brought about by the agents' own actions. Our use of this approximation is predicated on the assumption that changes in the world do not significantly affect the long–run payoffs to the agents; that is, all states are *quiescent*. This assumption makes sense in our setting, because the effect of scarcity of agents to complete tasks means that the introduction of additional tasks into the system only affects the agents' payoffs at the margin, and does not alter the utility of those tasks that the agents may have already begun to process (or chosen not to process). Furthermore, we derive the agents' utilities from the approximate global utility function such that the agents play a *potential game* [21] at each time–step, with the strategy space of the game for each subsequent time–step *overlapping* with the games before (and after) it. We do this by rewarding agents with their marginal contributions to task utilities. Generating a potential game is very useful because, first, the maximum approximate global utility is a Nash equilibrium, and second, it implies that each game can be solved by a distributed local search algorithm. In particular, we use the Distributed Stochastic Algorithm [5] to solve each approximating game (we could equally well use alternative methods, such as Distributed Simulated Annealing or Fictitious Play, as shown in [22]). Thus, we call our technique the *overlapping potential game approximation* (OPGA).

Beyond this, we extend the analysis of our approach to situations where the communication and observation range of the agents is restricted, so that we can test OPGA's robustness to these complications. In such settings, the agents cannot see the entire state of the world or do not know the strategies of all other agents, and therefore must make their decision on the basis of incomplete information. This type of limitation is common in many real–world scenarios, and particularly those that possess a spatial dimension. For example, in a disaster response setting, the central message distribution point may be out of action, or damage to physical infrastructure may remove the ability to use wide–area broadcast communication. In these situations, the agents' utility functions can still be derived in such a way so that they are aligned with the global objectives of the system.

In order to test the efficacy of OPGA, we empirically evaluate it on the ambulance–to–civilian allocation problem in RoboCup Rescue (RCR), which is an example of a task allocation problem with hard deadlines and varying processing requirements. By doing so, we show that OPGA performs comparably to a centralised task scheduler when communication is not limited, and that it outperforms the centralised approach as the agents' communication and observation ranges are restricted. Interestingly, we also find that OPGA sometimes performs better with moderate restrictions on its communication and observation range than it does when it has complete information. We conjecture that this is due to a reduction in the space of possible solutions, resulting in a quicker rate of convergence, and consequently fewer occasions of a configuration that is not a Nash equilibrium being generated as a solution (which itself is down to the fact that due to the finite number of iterations employed there is always a small probability that the generated solution is not a Nash equilibrium).

Given this context, this work extends the state of the art in the following ways:

1. We introduce a new technique for approximating stochastic games using a series of overlapping potential games.
2. We develop a novel distributed solution technique for the approximating games, based on the distributed stochastic algorithm.
3. We show that our technique is robust to restrictions on the range over which agents can communicate and observe (note that these are restrictions that typically cause centralised approaches to fail).

The first contribution is aimed at balancing the need for high quality solutions with the requirement that the control mechanism itself is computationally feasible, while, at the same time, the second contribution produces a robust and flexible mechanism that operates with limited communication overhead. The third contribution, then, demonstrates that the control mechanism we derive successfully satisfies these aims. Elements of this paper have appeared before in [23].

The paper is organised as follows: In the next section we review other approaches to distributed dynamic scheduling, and argue why they do not meet our requirements. Section 3 then introduces the game–theoretic background to our model. In Section 4 we formulate the problem as a stochastic game, and describe our approximation of the global utility function. Building on this, we show how to derive agents' utilities so that the resulting game is a potential game, and describe a local search algorithm that can be used to solve it. Finally, we discuss the effects of restricting the range over which agents can communicate. Then, in Section 5, we evaluate OPGA in the ambulance–civilian allocation problem in RCR. This demonstrates the benefit of using a decentralised method of control in settings where communication and observation are limited. Section 6 concludes.

2. RELATED WORK

In this section we review a number of control methods that could be applied to dynamic allocation and scheduling problems. To date, the main approaches to dynamic task allocation include: (i) domain–specific heuristics, (ii) modelling scheduling as a constraint program and solving this with either a centralised or decentralised algorithm, and (iii) auction allocation mechanisms and more general market–based approaches. Each of these will now be dealt with in turn. A comprehensive review of related problems can be found in [24].

First, there is a long history of using heuristics to solve scheduling problems. In particular, [25] addresses the family of *earliest deadline first* heuristic algorithms for scheduling in real–time scenarios comprising tasks with hard deadlines that compete for the same resources. The problem that we tackle here falls into this general class of problems, and, furthermore, the greedy algorithms we use as experimental benchmarks in Section 5 are also based on such heuristics. However, such algorithms rely on the centralisation of information and decision–making, and so are not appropriate for our application domains.

Second, a number of optimal algorithms for multi–agent scheduling problems have been proposed that work by reducing the problem to a constraint program. Examples of constraints include *resource relations*, which are shared by tasks that compete for the same resource, and *precedence relations*, which are shared by subtasks that need to be completed in a certain order. From these relations, a constraint program is constructed. This can be solved centrally (as in [26]) or using a decentralised constraint programming algorithm (such as DPOP [27]). Again, we rule out using centralised constraint solvers. Moreover, the distributed exact algorithms suffer from exponential growth of some aspect of the solution process (e.g. the size of the messages passed in DPOP is exponential in the depth of the communication network it is run on), so cannot easily be applied at the scales or in the timeframes we require.

Third, auctions and other market mechanisms are beginning to be used to add robustness to task allocation by giving agents the autonomy to construct their own bids, based on their own private or partial knowledge [28, 29]. However, such auctions often involve a significant communication overhead, which can impact on the timeliness of their solutions, and, to some degree, an auctioneer represents a single point of failure (just like a central decision maker). Other market–based allocation methods, such as bargaining and exchange markets, are similar to our work, as the local search algorithm we employ to solve each potential game effectively specifies a negotiation protocol. However, our method differs from this literature, because we are able to directly specify agents' utility functions.[2]

From this landscape, the work most similar to ours is [17], in which an autonomous vehicle–target assignment problem is addressed using a potential game formulation. In their model, vehicles (agents) operate individually to optimise a global utility. The global utility optimisation is obtained via an appropriate design of the vehicles' utilities and their negotiation protocol. While we use a similar approach, there are two fundamental differences. First, in their work, vehicles are assigned to a single target, whereas in our scenario each agent is required to perform a sequence of tasks, each of which has a hard deadline. This means that our agents are required to reason over the order in which they attend to tasks, not just which tasks to attend. Second, their approach assumes that all tasks are known at the start, while we assume that they are continually discovered at run–time.

Finally, our approach to approximating the stochastic game is motivated by a somewhat similar technique for producing approximate solutions to partially–observable stochastic games using a series of smaller Bayesian games [30]. In that work, a tractable Bayesian game is constructed at each time step from the most likely current states and state–transitions given an agent's beliefs. This Bayesian game is then solved to obtain a one–step policy that approximates the globally optimal solution of the original partially–observable stochastic game. In contrast, we construct a tractable multi–stage game of complete information at each time step, and because this is a potential game, it can be easily solved using a decentralised algorithm. The solution to this game is then used as a multiple–step policy to approximate the globally optimal solution.

More broadly, this work is also linked to other work carried out within the ALADDIN Project.[3] Alongside this and other work that has been applied in the RoboCup Rescue simulator,[4] ALADDIN Project researchers have also applied MAS–based techniques to the closely–related problem of industrial task allocation [6], evacuation scenarios [2] and distributed sensor management [34, 3]. Research has also been carried out on fundamental problems that arise in MAS design, including research into algorithms for distributed constraint optimisation problems (such as the Distributed Stochastic Algorithm used in this paper) [35], models of congestion in common–resource usage games [7] and coalition formation problems [33, 36].

3. PRELIMINARIES

This section introduces the foundations of our model, beginning with noncooperative games, extending these ideas to stochastic games, and finally considering the class of potential games.

3.1. Noncooperative Games

A noncooperative game, $\Gamma = \langle N, \{S_i, u_i\}_{i \in N} \rangle$, consists of a set of *agents*, $N = \{1, \ldots, n\}$, and for each agent $i \in N$, a

[2] In contrast, market–based task allocation methods are designed to incentivise agents with arbitrary utility functions to act in a way that maximises a social welfare function. Nonetheless, the connections between our work and mechanism design, in particular the *Groves mechanism*, are discussed in Section 4.3.

[3] www.aladdinproject.org

[4] See www.aladdinproject.org/technologies for applications in RoboCup Rescue of auctions for efficient resource allocation [31], consistent multiple hypothesis estimation in the face of faulty/untrustworthy sensors or incomplete probabilistic models [32], and optimal anytime coalition formation algorithms [33].

set of *strategies* S_i, and a *utility function* $u_i : S \rightarrow \mathbb{R}$. A joint strategy profile $s \in S$ is referred to as an *outcome* of the game, where $S = \cup_{i=1}^{N} S_i$ is the set of all possible outcomes. An agent's utility function ranks its preferences over outcomes in terms of the payoff it receives for an outcome, such that $u_i(s) > u_i(s')$ if and only if the agent prefers outcome s to outcome s'. We will often use the notation $s = \{s_i, s_{-i}\}$, where s_{-i} is the complimentary set of s_i.

In noncooperative games, an agent's goal is to maximise its own payoff, conditional on the choices of its opponents. Stable points in such a system are characterised by the set of *Nash equilibria*. A joint strategy, s^*, such that no individual agent has an incentive to change to a different strategy, is a Nash equilibrium, i.e.:

$$u_i(s_i^*, s_{-i}^*) - u_i(s_i, s_{-i}^*) \geq 0 \quad \forall\, s_i,\ \forall\, i. \tag{1}$$

In the next subsection, we discuss an extension of this simple static game model to stochastic, multi–stage settings, in which the agent's payoff functions vary as a function the state of the world.

3.2. Stochastic Games

Stochastic games are an extension of standard noncooperative games, for repeated interactions in which the game played by the agents at each time–step, t, varies probabilistically as a function of the state and the choice of strategies in the previous round, or simply as some environmental process evolves [19]. Formally, a stochastic game is a tuple, $\Gamma = \langle N, \Omega, \{\{S_i, u_i\}_{i \in N}\}_{\omega \in \Omega}, q \rangle$, comprising a set of *agents* $N = \{1, \ldots, n\}$, a set of *state variables* $\omega \in \Omega$, a set of *stage games* $\gamma(\omega)$ indexed by elements of Ω, with each having a *strategy space* S^ω and a set of *utility functions* $u_i^\omega(s)$, defined as in the standard noncooperative model above, and a *state transition function* $q(\omega^{t+1} | \omega^t, s^t)$. The state transition function gives the probability that the next period's state is ω^{t+1}, given the current state ω^t and the strategy chosen by the agents at time t, s^t. Although state transitions are stochastic, agents are assumed to know with certainty the current state. Intuitively, payoffs in the current stage game depend only on the state and the agents' current strategies, while the probability distribution on the following state is completely determined by the current state and strategy selection. A strategy in a stochastic game comprises a strategy for each of the stage games $s_i = \{s_i^\omega\}_{\omega \in \Omega}$; that is, there is a strategy component for every possible state of the world. Strategies in finite time step stochastic games are typically evaluated by their expected total reward:[5] $\mathbf{E}[u_i(s_i, s_{-i})] = \sum_{t=0}^{T} u_i^{\omega^t}(s_i, s_{-i})$.

3.3. Potential Games

Potential games are a subclass of noncooperative games. They are characterised as those games that admit a potential function, which is a real-valued function on the joint strategy space whose gradient is the gradient of the constituents' private utility functions [21]. The class of finite potential games have long been used to model congestion problems on networks [37], and, recently, they have been used to analyse distributed methods of solving target assignment problems [17, 38] and job scheduling [5].

Formally, a function $P : S \rightarrow \mathbb{R}$ is a *potential* for Γ if:

$$P(s_i, s_{-i}) - P(s_i', s_{-i}) = u_i(s_i, s_{-i}) - u_i(s_i', s_{-i})\ \forall\, s_i,\ s_i' \in S_i\ \forall\, i \in N.$$

Γ is called a *potential game* if it admits a potential. Intuitively, a potential is a function of action profiles such that the difference in its value induced by a unilateral deviation equals the change in the deviating agent's payoff.

The usefulness of potential games lies in the fact that the existence of a potential means that the game possesses two particularly desirable properties. The first is that every finite potential game possesses at least one pure strategy equilibrium [21]. Now, pure strategy Nash equilibria are particularly desirable in decentralised agent-based systems, as they imply a stable, unique outcome. Mixed strategy equilibria, on the other hand, imply a probability distribution over pure strategy profiles, which is not appropriate for the task allocation problems at hand. The second desirable property possessed by potential games is that they have the *finite improvement property*, meaning that any sequence of unilaterally improving moves converges to a Nash equilibrium in finite time. This property is important as it is used to guarantee the convergence of many simple adaptive processes to Nash equilibria in potential games (including the Distributed Stochastic Algorithm, as discussed in Section 4.3).

4. THE TASK ASSIGNMENT MODEL

We begin this section by defining our task allocation problem as a stochastic game. We then describe our approach to this problems, called the *overlapping potential game algorithm*, beginning in 4.2 where we describe our finite–horizon approximation of the global utility function. Then in 4.3 we show how agent utility functions are derived so that the agents play a potential game. Section 4.4 then discusses the distributed stochastic algorithm, which we use to solve the approximating potential games. Finally, in 4.5 we discuss the effects on our approach of limiting the distance over which agents can observe or communicate.

4.1. Stochastic Game Formulation

The full task allocation model is a finite stochastic game of complete information: the current state is known, future states are uncertain, agents have a finite set of strategies and play for a finite number of time steps. Similar to the general stochastic game description in Section 3.2, our model comprises:

- A set of *states* $\omega \in \Omega$, each of which defines a set of *tasks* $X = \{x_1, x_2, \ldots x_j, \ldots\}$, with each task possessing a *deadline*, $t_{x_j}^d$, a number of required *processing units*, y_{x_j}, and a *task utility function*, $u_{x_j} : S \rightarrow \mathbb{R}$,
- A set of *agents* $N = \{1, 2, \ldots, i, \ldots, n\}$, each with a *strategy space* S_i, with elements s_i, composed of a sequence of tasks to attend to, one for each time step

[5] This is in contrast to infinite time step stochastic games, which typically use the discounted expected total reward.

(i.e. $s_i = \{x^1, x^2, ..., x^t, ...\}$, and an *agent utility function* $u_i(s_i, s_{-i}) : S \rightarrow \mathbb{R}$,
- A state transition function $q(\omega^{t+1}|\omega^t)$, and
- A *global utility function* $u_G(s) : S \rightarrow \mathbb{R}$.

The problem we face is twofold: we must design (i) the agents' utility functions and (ii) a distributed negotiation protocol such that the system produces high quality solutions. The other elements of the above model — the transition function and the task and global utility functions — come directly from the problem specification, and are detailed now.

To begin, the transition function describes how new tasks are generated and introduced into the system. Then, the task utility function represents the payoff for completing a task, and in this case it is:

$$u_{x_j}(s) = \begin{cases} \beta^{t^c_{x_j}(s)} & \text{if } t^c_{x_j}(s) \leq t^d_{x_j}, \\ 0 & \text{otherwise,} \end{cases} \quad (2)$$

where $t^c_{x_j}(s)$ is the completion time for the task, given the agents' strategies s, $t^d_{x_j}$ is the hard deadline for successfully completing a task, and $0 < \beta \leq 1$ is a discount factor that incorporates any benefit of completing the task earlier. The task utility function possesses two properties that are important in our scenario. First, the conditional statement models the hard deadline, so if fewer than the minimum number of agents that are required to complete the task before $t^d_{x_j}$ attend to it, the utility is zero, even if some agents do attend to the task. This is important because, in our scenarios, tasks that are incomplete at the deadline are equivalent to unattended tasks. Second, increasing the number of agents beyond the number necessary to complete the task by its deadline improves the completion time, which raises the task payoff. This captures the benefit of completing tasks earlier.[6]

Given this, the global utility function ranks the overall allocation of tasks to agents, and is an aggregation of task utilities:

$$u_G(s) = \sum_{x_j \in X} u_{x_j}(s). \quad (3)$$

This preserves the desirable properties of the task utility function.

Now that we have defined the task and global utility functions for our problem, if we were working directly with the stochastic game model, we would define the agents' utility functions. However, note that an agent's strategy space in this model is the set of all permutations of assignments to tasks for each period; a strategy prescribes an action for each time step for every contingent state of the world. Thus, an agent's strategy is a set of vectors of actions, one vector for each state of the world, with an agent's utility function defined over this set and taking into account the transition probabilities between stage games. Given the huge number of possible states and action vectors of extremely large sizes (factorial on the number of tasks), evaluating and negotiating a set of joint strategies for this problem is clearly a very computationally expensive process, that would likely take a great deal of time. Furthermore, given that we intend to deploy our agents in a system where they will be required to make decisions in a short time frame, constructing such a strategy for the full set of possible outcomes is practically impossible, due to the huge number of possible future states and action combinations that need to be evaluated. For these reasons, we derive the agents' utility functions from a tractable approximation of the global utility function for the stochastic game.

4.2. An Approximation of the Global Utility Function

Rather than attempting to solve the stochastic game above, we approximate it using a series of static potential games of complete information, and in so doing, we directly address our requirement of tractability. Specifically, in this section we approximate the global utility function using a technique similar to a look–ahead policy or receding–horizon controllers commonly used in MDPs (see, [39] Chapter 5, for example). We can use this type of approximation because, in our application domains, we expect all states to be quiescent; that is, changes in the state of the world do not significantly affect the long–run payoffs to the agents, or there are no 'doomsday' outcomes.[7]

In more detail, the global utility is approximated as follows. At each time step, a game is constructed with each agent's strategy defined for a fixed decision window of w future time steps. In each of these games, an agent's strategy is a vector of tasks to attend to during the interval $[t, t+w]$, $s_i = \{x^t, x^{t+1}, \ldots, x^{t+w}\}$. In this way, at each time step, the stochastic game is approximated by a static game of complete information defined over the next w time steps. Then, the task utility functions in each approximating game are defined as in Equation 2, with the addition that, for tasks not completed by $t+w$, payoffs are calculated as if all of the agents' final strategy components s_i^{t+w} are repeated until the task in question is completed or its deadline passes. If we did not assume the continuation of these tasks, the utility of all incomplete tasks at time $t+w$ would be zero, potentially leading to an artificial bias against tasks with large processing requirements and/or long completion times. The global utility of this model is given by:

$$u_G^{t,w}(s) = \sum_{x_j \in X} u_{x_j}(s), \quad (4)$$

and we note that it is of the same form as that for the stochastic game model given in Equation 3, except that the constituent task utilities are defined over the restricted interval $[t, t+w]$.

This approximation of the global utility function introduces two types of errors. The first source of error is the

[6] The value of β in Equation 2 represents a trade–off between the number of tasks completed and the timeliness of those completed tasks. As we aim to maximise the number of tasks completed, we chose a value close to 1, however, if timeliness was our main concern, we would choose a lower value.

[7] If any state is non–quiescent, then our use of a look–ahead style approximation will suffer from the *horizon problem* [39] meaning that it will not be able to avoid entering states that lead, unavoidably, to bad outcomes beyond the length of the decision window used.

restriction of strategies to the decision window $[t, t+w]$. The result of this is that the value of future states beyond the next w time steps are not evaluated with the current choice of strategy. Now, although we only maximise $u_G(s)$ over the next w time steps, any adverse effects on the full $u_G(s)$ is expected to be small, because the game puts greater importance on tasks with closer deadlines. Similarly, if we used the full stochastic game model, tasks with earlier deadlines would be processed earlier. The second type of error is caused by not incorporating information about the exogenous evolution of the world (in our model, the arrival of new tasks) into the choice of state. However, as argued earlier, in the domains we are considering, the state of the world moves slow enough for us ignore this effect without introducing any significant errors (admittedly, this is a domain specific trait).

Now, because we are working on a problem for which a sequentially optimal solution is intractable, we are faced with a trade–off between the two sources of approximation error. The first type is reduced as the restriction on the size of w is relaxed. On the other hand, the second type is mitigated by using a shorter length window, because the difference in the predicted and actual state reached in the future is reduced. Consequently, our choice of window length reflects the need to balance the effect of these two sources of approximation error. Thus far, the value of w has to be determined experimentally as it depends on the domain (elaborated in Section 5.2).

4.3. Deriving the Agents' Utility Functions

Given the above approximation of the global utility function for our problem, the agents' payoffs are designed so that any increase in an agent's utility corresponds to an increase in $u_G^{t,w}(s)$. This enables us to produce high quality solutions using a control mechanism comprised of self–interested agents. However, in order to satisfy the robustness requirement, these utilities cannot simply be set equal to the global objective function, because that would mean each agent needs to have complete information of the strategies of others in the system to evaluate their own strategy. Instead, because the global utility is the sum of task utilities, an agent's marginal contribution to the global utility can be specified in terms of the sum of its contributions to individual tasks — that is, the difference between the task utility when the agent contributes to the task and when it does not. This form of utility function is similar to the Groves mechanism, in which agents in a team are paid an amount equal to their marginal contribution to the team utility [15].[8] However, in our setting we can do away with the explicit utility transfers that occur in mechanism design because the system designer is able to specify each agent's utility function directly. Note that this type of utility structure is also similar to the *wonderful life utility* introduced by [16]. Specifically, by using the marginal utility, a system designer can equate the agent's utility to the effect of its actions on the global utility. To this end, the marginal contribution of agent i to a task x_j is given by:

$$mu_i^{x_j}(s_i, s_{-i}) = u_{x_j}(s_i, s_{-i}) - u_{x_j}(s_0, s_{-i}), \tag{5}$$

where s_0 is the *null strategy*, in which the agent does not contribute to completing any task.

The relationship between the task utility function and an agent's marginal contribution to the task utility is shown in the example in Figure 1. This shows $u_{x_j}(s)$ and $mu_i^{x_j}(s)$ for a task requiring 4 units of processing and with a deadline 2, in which an agent processes at a rate of 1 unit per time step. A minimum of 2 agents are required to complete the task — a constraint captured by the increase in the task and agents' utilities as the number of agents increases from 1 to 2. If more than this number of agents attend, the task utility continues to increase as the completion time decreases, however, the marginal contribution of each additional agent beyond this point decreases.

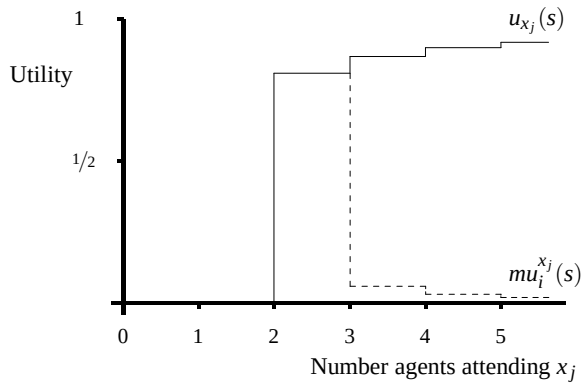

FIGURE 1. An example of task utility and agent marginal utility, with $y_x = 4$, $t_x^d = 2$ and $\beta = 0.9$, and an agent processing rate of 1 per time step.

An agent's marginal utility values are used to construct its payoff for each strategy, which is the sum of its marginal contributions to all the tasks it attends to in the next w time steps:

$$u_i(s_i, s_{-i}) = \sum_{x_j \in s_i} mu_i^{x_j}(s_i, s_{-i}). \tag{6}$$

Note that the first summation could be taken over all tasks in X with the same result, as $mu_i^{x_j}(s_i, s_{-i})$ is zero for all tasks to which i does not contribute. This point is important, as it implies that a change in strategy that increases an agent's utility always corresponds to an increase in the global utility restricted to the decision window $[t, t+w]$. Consider the difference in i's utility for switching from s_i to s_i'. The following shows that the change in an agent's own utility

[8] In order to make the connections clear, observe that if we were trying to incentivise agents, who possess private preferences, to act in a certain manner, then a mechanism design procedure, such as the Groves mechanism, would be an appropriate choice of control mechanism.

is equal to the change in the global utility:

$$
\begin{aligned}
&u_i(s_i, s_{-i}) - u_i(s_i', s_{-i}) \\
&\quad = \sum_{x_j \in s_i} \big(u_{x_j}(s_i, s_{-i}) - u_{x_j}(s_0, s_{-i})\big) \\
&\qquad\qquad - \sum_{x_j \in s_i'} \big(u_{x_j}(s_i', s_{-i}) - u_{x_j}(s_0, s_{-i})\big) \\
&\quad = \sum_{x_j \in X} \big(u_{x_j}(s_i, s_{-i}) - u_{x_j}(s_0, s_{-i}) \\
&\qquad\qquad - u_{x_j}(s_i', s_{-i}) + u_{x_j}(s_0, s_{-i})\big) \\
&\quad = u_G^{t,w}(s_i, s_{-i}) - u_G^{t,w}(s_i', s_{-i}).
\end{aligned}
$$

Thus, a game played between agents with utility functions as given above is a potential game, with a potential function given by the approximation of the global utility function over the decision window. There are two consequences to this result. First, the globally optimal allocation of tasks to agents in the window resides in the set of Nash equilibria. To see this, assume that the optimal point is not a Nash equilibrium. Then there must be some agent that can alter its state to improve its utility, which in turn will improve the global utility, which contradicts the assumption that the optimal point is not a Nash equilibrium. Despite that, in most cases some sub–optimal Nash equilibria also exist at local maxima of the global objective function. Second, the game has the finite improvement property (see Section 3.3), implying the convergence of the distributed stochastic algorithm, as we discuss in the next section.

4.4. The Distributed Stochastic Algorithm

The Distributed Stochastic Algorithm (DSA) is a local iterative approximate best response algorithm [5]. We use it here because in previous work, which compared the performance of such algorithms in the related class of distributed constraint optimisation problems, we identified it as an algorithm that can quickly produce good quality solutions with a low communication overhead [22]. Though we suggest DSA, many other decentralised negotiation protocols can also work well with our model (e.g. spatial adaptive play or fictitious play, as catalogued in [35]). However, we note that using a complete algorithm in place of DSA, such as DPOP [27], is not appropriate, because of their high computation time and communication requirements. This is particularly so in our setting since the constraint graphs for the task allocation problems addressed here are fully connected; that is, every agent is connected to every task. As such, the computational burden on complete algorithms would be prohibitively large, as the complexity of these algorithms increases exponentially with the degree of cyclicity of the constraint graph (as discussed in Section 2).

In more detail, DSA is a synchronous algorithm, in that agents act in step, however, at each time step, an agent has some probability p of activation, known as the degree of parallel executions [5]. At each time step, an agent computes a strategy that increases its payoff the most — its best response. Then, with probability p, it switches to this best–response strategy, and with $(1-p)$ it keeps its current strategy. Finally, if no change improves the payoff, the agent does not change its strategy. Importantly, DSA converges to a Nash equilibrium in potential games. Briefly, this is because no agent will leave a Nash equilibrium after the first time it is played, and for values of $0 < p < 1$, the probability that the agents play a Nash equilibrium goes to 1 as time progresses, as a consequence of the finite improvement property. In contrast to the complete algorithms, the complexity of computing the best responses used in DSA is low. Specifically, the best–response function takes as an input a strategy for every agent except i, computes values for each of its own strategies given this context, and maximises over this vector, resulting in a worst–case complexity of the order $O(n|s_i|)$, where n is the number of agents $|s_i|$ is the size of i's strategy space.

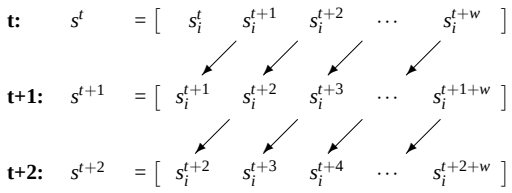

FIGURE 2. Recycling solutions: Typically, the $t+1$ to $t+w$ strategy components from game t are used as initial conditions for DSA in game $t+1$, and so on.

In application domains with short decision horizons, like RoboCup Rescue, a good initial set of conditions can significantly improve the convergence time of DSA. For this reason, in our model, solutions to one approximating game are used as initial conditions, or partial solutions, for DSA in the following game. In more detail, usually the initial best response played by an agent using DSA is computed assuming that its neighbours are playing randomly. Instead, in our problem, because the $t+1$ to $t+w$ strategy components of consecutive games overlap, we can reuse these components as the starting conditions for DSA in each subsequent game (as shown in Figure 2). In particular, each agent assumes that its neighbours play a strategy comprising their $t+1$ to $t+w$ components from the previous game with a random final strategy component. Reusing the solutions to previous games as initial conditions in this manner is particularly useful for situations where the number of negotiation steps is limited by communication restrictions (such as in RoboCup Rescue). A related issue is that newly arriving tasks with pressing deadlines have the potential to induce significant re–coordination by the agents, rendering the partial solutions negotiated in previous game's strategies irrelevant. In this way, these new tasks have the potential to disrupt the convergence of the algorithm. However, in practice, by using any reasonable value for p (i.e. $0 \ll p \ll 1$) in conjunction with a long window, the agents are seen to deal with such disruptions in a graceful manner. That is, as long as p imparts significant inertia on the existing strategy, it will prevent the algorithm from being

significantly disrupted.

Finally, we note that because the probability of DSA converging goes to 1 in the time limit, defining a termination condition for the algorithm is problematic. However, in practice, DSA converges reliably and quickly [5, 22], and as such, the algorithm is usually left to run for a predefined number of iterations. In our setting, which is characterised by severe communication restrictions, we limit this number to *one* iteration for each action time step. This may seem an unreasonably severe restriction. However, due to the recycling of previous solutions described above, the action taken at t is iterated over w times during the operation of the OPGA algorithm. Thus, for large enough w convergence is expected with very high probability.

4.5. Dealing with Limited Range Communication

So far in this section, we have shown how to implement a general distributed technique for solving a dynamic task allocation problem. However, as motivated earlier, we also wish to develop a technique that is robust in the face of restrictions on the distance over which agents can communicate. In particular, the technique we develop is a natural extension to the utility function described in Section 4.3, and is appropriate for any task allocation problem with a spatial dimension. In more detail, we model situations where an agent can communicate over a limited distance, r, and is only aware of some of the tasks. As such, the major changes in the method are that: (i) the set of strategy components available to an agent is restricted to only those tasks it is aware of, $X_i \subseteq X$,[9] and (ii) the agents' utility computations are carried out using only the strategies of those agents that are currently within its communication range, $j \in N_i \subseteq N$. This gives us the following agent utility function:

$$u_i(s_i, s_{N_i}) = \sum_{x_j \in s_i} mu_i^{x_j}(s_i, s_{N_i}) = \sum_{x_j \in s_i} \left(u_{x_j}(s_i, s_{N_i}) - u_{x_j}(s_0, s_{N_i}) \right), \tag{7}$$

where s_i is restricted to the set of tasks of which i is aware, X_i. That is, an agent's utility function is identical to the setting without communication restrictions, and only the set of tasks which it know of is restricted. Now, using this form for an agent's utility function means that the approximate global utility function (Equation 4) need not be a potential function for the game. However, if all agents are aware of those agents attending to their tasks, then Equation 4 acts as a potential function. This is always the case when the agents are at (or sufficiently near) the location of their tasks, since once the agents are within each other's communication ranges, they exchange all of their information about the state of the world, and, therefore, their task utility computations are identical (as they are in the unlimited communication range setting). Moreover, under these conditions, DSA is also guaranteed to converge. On the other hand, because the components of an agent's strategy are restricted to those tasks it is aware of, parts of the global utility function are, in effect, inaccessible to the agents. Nonetheless, the accessible local maxima of the approximate global utility are Nash equilibria of the game.

[9] The way that agents learn about tasks is typically specific to the domain, and how this occurs in RoboCup Rescue is discussed in Section 5.

5. APPLICATION TO ROBOCUP RESCUE

In this section, we describe an application of our overlapping potential game algorithm (OPGA) to RoboCup Rescue (RCR). RCR is a simulation of a disaster response scenario in a large city (see `http://www.robocuprescue.org` for more details). RCR is a well–known domain used for benchmarking solutions related to multi–agent based coordination, and an exemplar dynamic task allocation and scheduling problem, which shares many salient characteristics with other such problems. It is a complex setting in which teams of agents have to allocate and perform tasks using incomplete information in a non–deterministic environment, in real–time. Thus, it provides an ideal platform for evaluating the efficacy of our control mechanism.

In more detail, RCR is composed of a map of a large city, containing buildings, roads, injured casualties, and three types of emergency service agents: ambulances, fire brigade and police. Figure 3 gives an example of the maps we used in our RCR experiments, the different agents marked as dots. In more detail, ambulance agents work by extracting trapped casualties from collapsed buildings, fire brigade agents have the task of extinguishing burning buildings, and police agents unblock roads and search for trapped casualties and burning buildings. However, in this work, in order to clearly identify the effects of using OPGA, we consider a limited version of RCR containing no fires or blocked roads, so the only problem is to coordinate the ambulance agents to extract injured civilians. The global objective, then, is to coordinate the actions of emergency service agents such that they save as many casualties as possible, in as short a time as possible. The performance of a coordination strategy is measured by the sum of casualties 'health points', which decay over time while they are trapped in collapsed or burning building.

We now map the ambulance–casualty allocation problem in RCR to our generic OPGA framework, and then discuss the experiments we use to evaluate our approach.

5.1. Ambulance–Civilian Allocation

In order to apply the model developed in Section 4, we assume that each agent, i, corresponds to an ambulance and each task x_j represents an injured civilian that needs rescue.[10] Each civilian has a hard deadline, $t_{x_j}^d$, by which time it must reach a refuge if it is to survive, and a processing requirement, y_{x_j}, corresponding to the time it would take the crew of a single ambulance to be in a position to remove it from the scene. In RCR, the number of injured civilians is typically much greater than the number of ambulances, and

[10] The ambulance to casualty allocation mechanism derived in this Section represents a first step in deriving a complete set of algorithms integrating all of the allocation problems in RCR. This and other extensions are discussed in Section 6.

Foligno

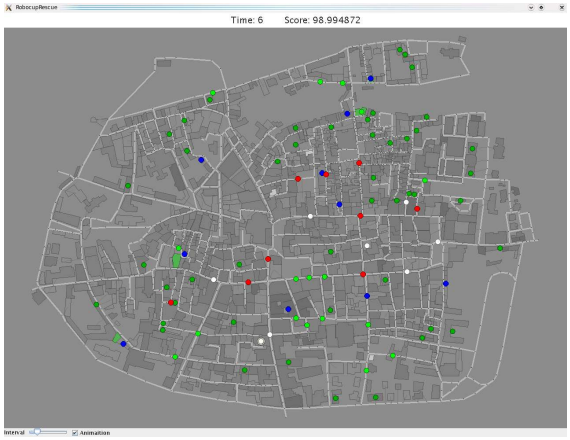

FIGURE 3. Example of a map used in our RoboCup Rescue experiments.

not all the injured civilians are known to the ambulances at the start. Rather, they are discovered over time. This means that an ambulance must negotiate a sequence of civilians to attend to with other ambulances, with the rescue of some civilians requiring more than one ambulance because of a high y_{x_j} and/or an imminent $t^d_{x_j}$.

Given this, the task completion time, $t^c_{x_j}(s)$ in Equation 2, is the time it takes a team of ambulances, given by the joint strategy profile s, to rescue the civilian. This incorporates both the civilian's processing requirement (the time needed by the team before taking it to the refuge) as well as estimates of the time it takes for the team members to travel to the civilian. The global utility $u_G(s)$ increases with the number of casualties rescued, and for each casualty, increases with lower completion times. Regarding an ambulance's marginal utility (Equation 5), because β is close to 1, the contribution of an ambulance that is critical to the rescue of civilian x_j before $t^d_{x_j}$ is greater than the benefit of speeding up the rescue of a civilian that is already assured of being saved. This effect is demonstrated in Figure 1. Following this, an agent's utility (Equation 6) is then the sum of its contribution to all tasks in the window $[t, t+w]$, and consequently, the approximate global utility function acts as a potential for the entire game. Thus, the salient features of this problem are captured in our model.

Nonetheless, two small variations to the standard DSA are necessary to successfully implement our model in RCR. First, one component of an ambulance's role is to transport rescued civilians to a refuge, which takes time and can upset the agent's strategy. Because of the difficulties of capturing this requirement in an agent's strategy space, we allow the following innovation to DSA; whenever an ambulance has returned a civilian to a refuge, it always computes a completely new best response strategy. This is done because its existing strategy will be completely outdated (i.e. the tasks it had in its current strategy it will not have attended to). Second, because agents are not computing their best strategy at every time step, if others have changed their strategy, it is possible that an agent's value for a task completion time differs significantly from the true value. This may require the agent to shift forward many elements of its strategy vector (many more than illustrated in Figure 2). Now, if too many elements are removed, the resulting recycling of partial past solutions as initial conditions may be counter–productive. For this reason, if, in recycling past components, an agent's strategy vector is advanced up by less than $w/3$ components, then with probability p the agent computes a completely new strategy, and with probability $(1-p)$ it generates a new strategy for the remaining components only, as in the usual operation of DSA (i.e. the same value of p is used as discussed in Section 4.4). However, if more than than $w/3$ components are removed, maintaining the current strategy will rarely be useful to the system, so the agent always computes an entirely new strategy.[11]

5.2. Experimental Design

We now discuss the design of the RCR experiments we used to evaluate our OPGA algorithm. Specifically, we ran OPGA on three standard RCR maps — Kobe, Virtual City and Foligno (the map shown in Figure 3), each of which contain 80 civilians and 6 ambulances. Foligno is a larger and more unstructured map than Kobe or Virtual City, making it relatively hard to detect civilians there. The information about the casualties is gathered by fire–brigades and police–patrols that explore the map and pass it on to the ambulances, thus simulating a dynamic continuous stream of tasks. In the limited range scenario, an exploring agent (i.e. a police or fire–brigade) can pass on information to an ambulance only when it is within the communication range, thus representing the limited observation range of the ambulances. Each simulation is 300 time steps in duration. We evaluate two parameter setting of our method with $p = 0.5$ and 0.9 — OPGA(0.5) and OPGA(0.9) — and a decision window of $w = 30$ steps for both. As our preliminary experiments showed that the results were not very sensitive to different p values between 0.5 and 0.9, we limited our current results to the two endpoints of this range. The value for w was chosen from preliminary experiments, as it depends on the nature of the domain. To achieve statistical significance (results are shown with 95% confidence intervals), each experiment was run 30 times. The performance in an experiment is reflected by the score obtained at the end of the simulation. This score is a standard provided by the RCR framework and is the sum of the health points of the civilians in the map. The health of an unrescued civilian decreases with time until it reaches 0 (which, in fact constitutes the deadline), while that of a rescued civilian improves with time. We also measure the number of civilians saved over time, because it gives an insight into how the rate of rescue is affected by the rate of discovery of casualties.

We ran two batches of experiments. In the first, the agents' communication range was not restricted. We use these results to directly compare the performance of OPGA

[11] Experimental evidence has shown two–thirds to be a reasonable value for this threshold.

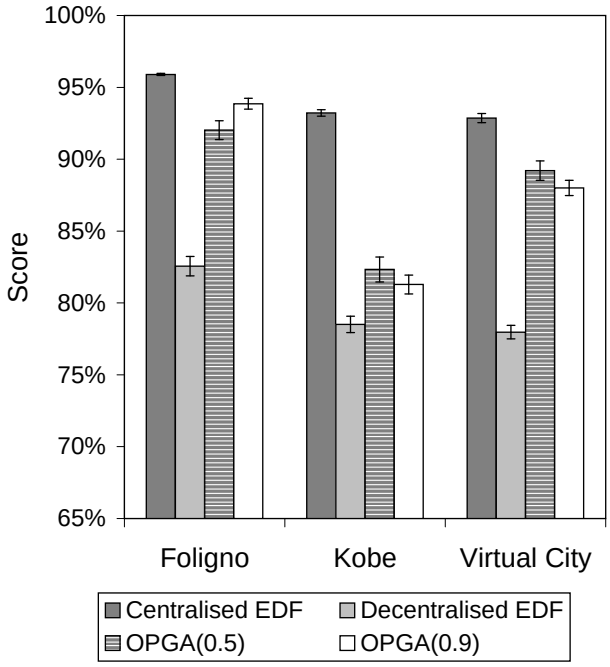

FIGURE 4. Comparing the methods across three maps.

to a centralised (myopic, greedy) *earliest deadline first* (EDF) scheduler as an upper bound and a decentralised EDF heuristic as a comparable lower benchmark. The former lists civilians in order of their deadline, centrally allocates free ambulances to the civilian with the earliest deadline up to the point where it is assured of being completed, and then allocates ambulances to the next civilian on its list, and so forth. Now, because the allocation is performed centrally, no mis–coordination is possible — neither fewer, nor more agents than are required will ever be allocated to a civilian. As such, the centralised EDF scheduler is expected to out perform OPGA. Under the decentralised EDF heuristic, each ambulance simply attends to the task with the shortest deadline. This approach will typically lead to an over–allocation of ambulances to civilians with short deadlines, and it will occasionally allocate ambulances to a civilian even when their efforts to save it are bound to be futile.

In the second batch of experiments, we test the performance of OPGA with restrictions on the range of the agents' communication and observations, as discussed in Section 4.5. These restrictions are 20% 15%, 10%, and 5% of the maximum distance between two points on the map in question (so the area covered by an agent's range decreases quadratically with these values). In this batch, we compare OPGA(0.9) to the centralised EDF scheduler only. We do this to test our hypothesis that OPGA performs better than the centralised EDF scheduler in scenarios where the communication and observation range is restricted.

5.3. Results

To begin with, we discuss the results of the first batch of experiments. To this end, Figure 4 shows the mean performance of OPGA(0.9) and OPGA(0.5) compared to the centralised and decentralised EDF methods in the three maps. Although the difference in score between OPGA and the centralised EDF heuristic is statistically significant, OPGA produces solutions that are within 6% of the centralised approach and, additionally, OPGA performs significantly better than decentralised EDF. When taken together, these results show that our approach, based on overlapping potential games, is a good approximation of the optimal solution to the stochastic game model of the ambulance–to–civilian problem in RCR.

In more detail, both versions of OPGA perform better in the Foligno and Virtual City scenario than in the Kobe scenario. Furthermore, a $2^k r$ factorial design test on the results evaluating the effects of the value of p and the map on the score indicates that 95% of the variation of the score is explained by variation in the map, and less than 1% by variations in p. The cause of the variation in scores between maps is due to the rate at which new trapped civilians are introduced. In particular, civilians are discovered at a quicker rate in the Kobe map than in Foligno or Virtual City. This is illustrated clearly in Figure 5. Here, a slower rate of discovery allows OPGA to find good quality solutions more regularly than in maps where, at times, the rate of civilians' discovery is faster. Thus, OPGA performs better in Foligno and Virtual City than in Kobe. Furthermore, this matches with the assumption we make that the state of the world moves slowly enough for us ignore the effect of the possible changes to the state of the world (in particular, the list of civilians), without inducing significant errors. When this assumption is less warranted, as in the Kobe scenario, the algorithm performs relatively worse.

Observations of the behaviour of agents when using the three different approaches yield the following deeper insights into the difference in their performance. (We encourage the reader to visit `www.aladdinproject.org/technologies.html` to view a video demonstrating OPGA and contrasting it to the centralised and decentralised EDF heuristics.) Beginning with the decentralised EDF heuristic, this algorithm's behaviour is characterised by 'clumping', meaning the agents all attend to the same task, and move to a new task together once the the current task has been processed. This often leads to severe over–allocation of resources to tasks with close deadlines, and the negative effects on the quality of the solutions generated are two–fold. First, the over–allocation comes at the expense of other tasks completion, and, consequently, these other tasks' deadlines pass before they receive sufficient processing. Second, tasks with imminent deadlines and high processing requirements are started even if they cannot be completed in time, representing a gross waste of resources.

Next, consider the centralised EDF heuristic. In the present full information setting, agents share the same world view (because communication ranges are not restricted). Thus, centralised EDF succeeds in avoiding most of the pitfalls of the decentralised EDF approach. Namely, first, because it is centralised it can allocate only the minimum number of agents to complete tasks, and so frees up

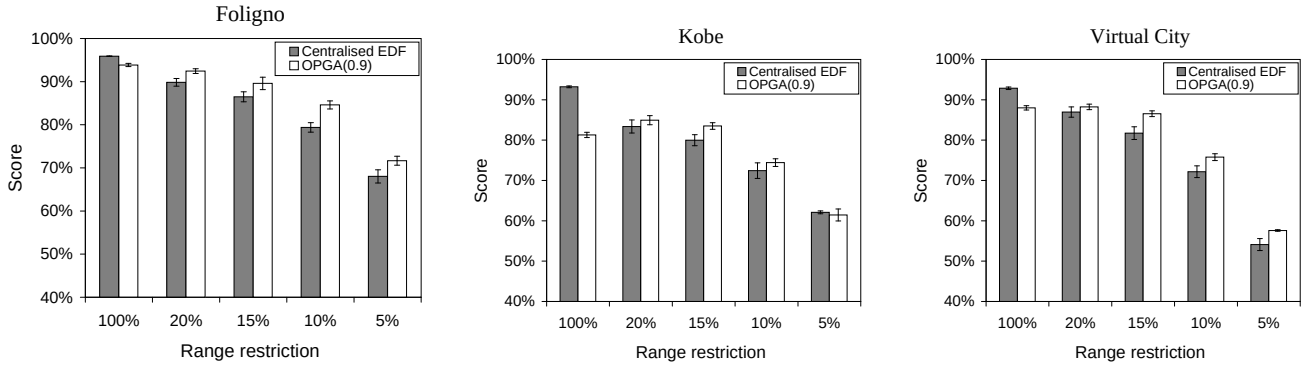

FIGURE 6. Comparing the methods as the communication and observation range is restricted.

resources to be allocated to other tasks with impending deadlines. Second, if a task cannot be completed before its deadline, it is not attempted by the centralised EDF approach. In the presence of full communication, then, this heuristic results in behaviour characterised by the orderly partition of agents among tasks, and little of the clumping seen under the decentralised EDF algorithm (except when warranted by the tasks at hand).

Finally, OPGA's behaviour, as well as the quality of the solutions it generates, falls between the two EDF heuristics. Although for the main part, the agents' behaviour under OPGA follows the orderly division seen in the centralised EDF heuristic, some clumping behaviour emerges as does frequent reassignments or 'thrashing' (also noted by [5] in a jobs scheduling domain). Thrashing is more prevalent with higher values of p (i.e. $p = 0.9$ saw greater levels of thrashing than $p = 0.5$), because the agents adopt new best responses more frequently and, therefore, are more likely to adopt conflicting strategies at the same time. Similarly, some over–allocation to tasks also occurs, and this is more common with lower values of p, due to the agents' not adjusting their strategies frequently enough. These behaviours make OPGA's performance poorer than that of centralised EDF, and are an artefact of the DSA algorithm used to solve each of the static potential games. However, the agents do partition themselves between the tasks in a generally sensible fashion, so avoid the most costly errors that decentralised EDF commits.

Now we turn to the second batch of experiments, in which the observation and communication ranges of the agents are restricted. Note that, because the decentralised EDF heuristic does not communicate, its performance does not vary at all over the second batch of experiments, so it is not considered. From Figure 6, we observe that the performance of the centralised EDF heuristic degrades at a quicker rate than OPGA, both in terms of its mean performance and the variability in its performance (as seen in larger error bars at each restriction level). The reasons for this are as follows: recall that the centralised EDF heuristic operates by the agents all following an identical decision procedure, which acts like an oracle. The good performance of this heuristic is, therefore, dependent on the agents holding identical information about the state of the world. When the local information held by the agents differs, the overall coherence of the plans generated by the centralised EDF heuristic degrades, resulting in a rapid downturn in performance.

In contrast OPGA performs better than the centralised approach whenever the agents' communication range is restricted (with the exception of Kobe when restricted to 5%, which is probably due to stochastic variation, and not statistically significant). This is because the performance of OPGA does not depend on the degree of coherence between the agents' world view to the same degree as centralised EDF. OPGA is flexible enough to cope with different information because it operates by exchanging strategies: that is, best responses are always only computed on the basis of known strategies of other agents in the system, not from a common world view.

However, when the range is severely restricted (such as when it is restricted to 5% of the map), the information flowing to the agents is minimal, hence the performance of any method will tend to be poor. This is precisely the effect of restricting the communication and observation range we expected to see, and justifies our arguments for using a principled decentralised algorithm in restricted range environments.

Furthermore, for moderate restrictions (15–20%), the performance of OPGA actually improves in both Kobe and Virtual City. This is a surprising result. It occurs because, under restricted information, the quiescence assumption motivating our choice of approximation is better supported than in the full information case. That is, the state of the world, as perceived by each agent, is more stable when their communication and observation ranges are moderately restricted. The subsequent degradation in performance of OPGA is due to a simple lack of information flowing to the agents. This effect is not reproduced in Foligno because OPGA does well, even in the full communication case, as the rate of discovery of casualties is slow.

Again from observations of the behaviour of the agents we gain further insights into the reasons for the performance of the different algorithms. First, consider the centralised EDF heuristic. As the restrictions on the communication range begin to bite, agents using this heuristic begin

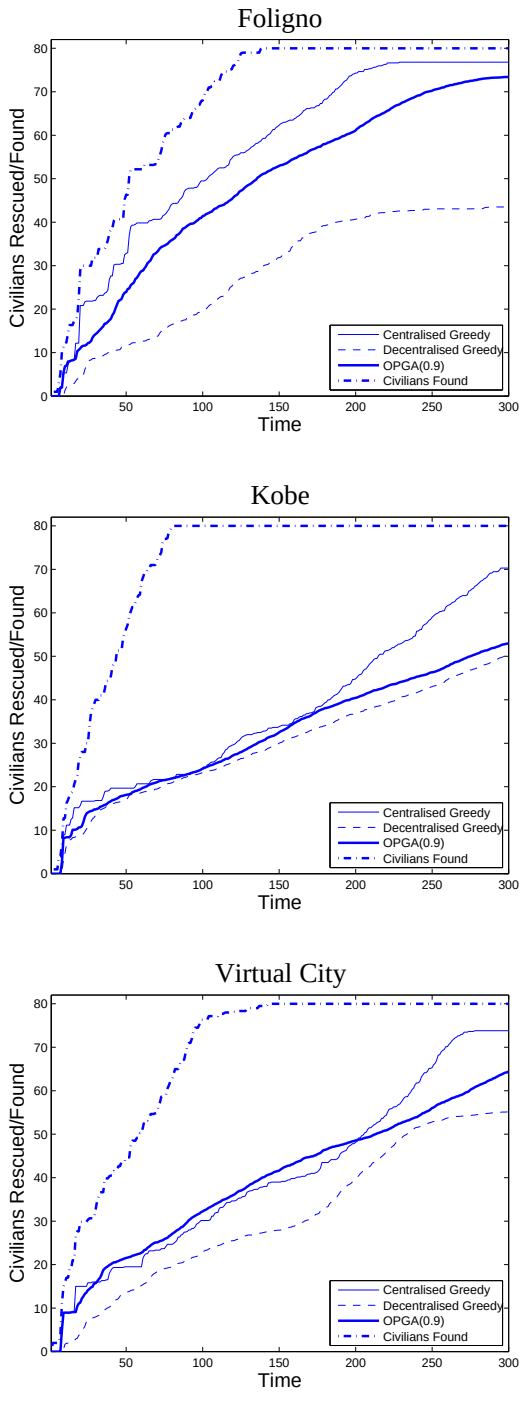

FIGURE 5. Exemplar time series comparing the methods on the Kobe, VC and Foligno maps, alongside the number of casualties found.

to act more like they are using the decentralised EDF heuristic. The orderly partition of agents between tasks seen with full communication range disappears, and is replaced by clumping behaviour and some thrashing. These detrimental behaviours begin to emerge because information about the state of the world is exchanged only once the agents move into each others' communication range. However, worse than these two behaviours, for more limited communication ranges, gross mis–allocation of resources occurs. Specifically, individual agents act as if all others share their world–view and allocate themselves to tasks requiring more than just their own resources to complete (plus the resources of any other similarly 'deluded' agents). This would appear to be the main cause of the dramatic decline in the performance of the centralised EDF heuristic in scenarios with communication range restrictions.

Second, OPGA's behaviours and performance remains relatively stable, with an increase in thrashing only seen for severe communication range restrictions. Like the centralised EDF heuristic, this is a result of the agents sharing information only once they have moved into range of each others' communications. However, because OPGA computes new best response strategies based on communicated strategies alone, it avoids the other costly mis–allocations that adversely affect the behaviour of centralised EDF, thus resulting in better performance when communication ranges are restricted.

6. CONCLUSIONS

In this paper, we derived a distributed game–theoretic control mechanism for decentralised planning to address dynamic task allocation problems. In more detail, there are two main aspects to the problem addressed. First, each agent has to perform a sequence of tasks over time and often tasks may require more than one agent for their successful completion. Second, the set of tasks is dynamic as new tasks are discovered over time. This leads to a stochastic game formulation.

However, stochastic games are generally intractable. Consequently, an optimal algorithm would not have satisfied the additional requirements of a control mechanism for the domain, namely robustness and computational tractability (with its consequences for communication use and the timeliness of the solutions generated) as well as optimality. In order to satisfy these requirements, we proposed a technique for approximating a stochastic game using a sequence of overlapping potential games, which are derived from a finite horizon approximation of the global objective function. Importantly, the agents' utilities are derived in such a way that they do not require complete information about the state of the world or of the actions of other agents in the system, and as such, the technique is robust to communication restrictions. In order to generate a solution, agents negotiate with each other to decide which tasks to act on in the next few time steps, and, in particular, we suggested the use of the distributed stochastic algorithm as a negotiation technique, because it has been shown to converge quickly and operates using a low communication overhead. Empirical results showed the efficacy of our approach in stochastic environments with limited information. Specifically, in order to test the performance and robustness of our control mechanism, we implemented it in the RoboCup Rescue disaster response simulator. We found that it performs comparably to a centralised task scheduler when communication is not

limited, and that it outperforms the centralised approach as the agents' communication and observation ranges are restricted.

Possible extensions to this work include generalising our model to capture other aspects of complex disaster scenarios, such as allowing agents to have differing costs for performing the same task or representing deadlines by a distribution over times, all of which the agents have to consider when making their decisions. Furthermore, we believe it is possible to incorporate heterogeneous tasks and agents with differing resources at their disposal, to extend the scope of the overlapping potential game technique. An example application of this would be a control mechanism that integrates the coordination of all three agent types in the RoboCup Rescue disaster response simulator— ambulances, police and fire brigade.

ACKNOWLEDGEMENT

This work was supported by the ALADDIN (Autonomous Learning Agents for Decentralised Data and Information Systems) project and is jointly funded by a BAE Systems and EPSRC (Engineering and Physical Sciences Research Council) strategic partnership (EP/C548051/1). The authors thank Sebastian Stein and Simon Williamson for their useful comments.

REFERENCES

[1] Kitano, H., Todokoro, S., Noda, I., Matsubara, H., and Takahashi, T. (1999) Robocup rescue: Search and rescue in large-scale disaster as a domain for autonomous agents research. *IEEE International Conference on System, Man, and Cybernetics (SMC '99)*, Tokyo, Japan, 12–15 October, pp. 739–743. IEEE.

[2] Filippoupolitis, A. and Gelenbe, E. (2009) A distributed decision support system for building evacuation. *Proceedings of the 2nd IEEE International Conference on Human System Interaction*, Catania, Italy, 21–23 May, pp. 323–330. IEEE.

[3] Stranders, R., Farinelli, A., Rogers, A., and Jennings, N. R. (2009) Decentralised coordination of mobile sensors using the max–sum algorithm. *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI–09)*, Pasadena, CA, USA, 11–17 July, pp. 601–608. Elsevier, Amsterdam, Netherlands.

[4] Heikkinen, T. (2006) A potential game approach to distributed power control and scheduling. *Computer Networks*, **50**, 2295–2311.

[5] Zhang, W. and Xing, Z. (2002) Distributed breakout vs. distributed stochastic: A comparative evaluation on scan scheduling. *Proceedings of the AAMAS–02 workshop on Distributed Constraint Reasoning*, Bologna, Italy, 16 July, pp. 192–201.

[6] Stranjak, A., Dutta, P. S., Ebden, M., Rogers, A., and Vytelingum, P. (2008) A multi–agent simulation system for prediction and scheduling of aero engine overhaul. In Padgham, L., Parkes, D., Müller, J., and Parsons, S. (eds.), *Proceedings of the 7th International Conference on Autonomous Agents and Multiagent Systems (AAMAS–08)*, Estoril, Portugal, 12–16 May, pp. 81–88. IFAAMAS.

[7] Byde, A., Polukarov, M., and Jennings, N. R. (2009) Games with congestion-averse utilities. *Proceedings of the 2nd International Symposium on Algorithmic Game Theory*, Cyprus, 18–20 October, pp. 220–232.

[8] Jennings, N. R. (2001) An agent-based approach for building complex software systems. *Communications of the ACM*, **44**, 35–41.

[9] Jennings, N. R. and Bussmann, S. (2003) Agent-based control systems. *IEEE Control Systems Magazine*, **23**, 62–74.

[10] Greenwald, A., Kirby, R. M., Reiter, J., and Boyan, J. (2001) Bid determination in simultaneous auctions: A case study. *Proceedings of the 3rd ACM Conference on Electronic Commerce (EC–01)*, Tampa, Florida, 14–17 October, pp. 115–124. ACM, NY, USA.

[11] Gerding, E. H., Dash, R. K., Yuen, D. C. K., and Jennings, N. R. (2007) Bidding optimally in concurrent second-price auctions of perfectly substitutable goods. *Proceedings of the 6th International Conference on Autonomous Agents and Multiagent Systems (AAMAS–07)*, Honolulu, Hawai'i, USA, 14–18 May, pp. 267–274. IFAAMAS.

[12] Dasgupta, P. (1998) Trust as a commodity. In Gambetta, D. (ed.), *Trust: Making and Breaking Cooperative Relations*, pp. 49–72. Blackwell, MA, USA.

[13] Teacy, W. T. L., Chalkiadakis, G., Rogers, A., and Jennings, N. R. (2008) Sequential decision making with untrustworthy service providers. *Proceedings of the 7th International Conference on Autonomous Agents and Multiagent Systems (AAMAS–08)*, 12–16 May, pp. 755–762. IFAAMAS.

[14] Stein, S., Jennings, N. R., and Payne, T. (2008) Flexible service provisioning with advance agreements. *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS–08)*, Estoril, Portugal, 12–16 May, pp. 249–256. IFAAMAS.

[15] Groves, T. (1973) Incentives in teams. *Econometrica*, **41**, 617–631.

[16] Wolpert, D. H. and Tumor, K. (1999) An overview of collective intelligence. In Bradshaw, J. M. (ed.), *Handbook of Agent Technology*. AAAI Press/MIT Press, MA, USA.

[17] Arslan, G., Marden, J. R., and Shamma, J. S. (2007) Autonomous vehicle-target assignment: A game theoretical formulation. *ASME Journal of Dynamic Systems, Measurement and Control*, **129**, 584–596.

[18] Hindriks, K., van der Hoek, W., and van Riemsdijk, B. 10–15 May Agent programming with temporally extended goals. In Decker, K., Sichman, J., Sierra, C., and Castelfranchi, C. (eds.), *Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems (AAMAS–09)*, Budapest, Hungary, 10–15 May, pp. 137–144. IFMAAMAS.

[19] Littman, M. L. (1994) Markov games as a framework for multi-agent reinforcement learning. *Proceedings of the 11th International Conference on Machine Learning (ICML–94)*, New Brunswick, NJ, USA, 10–13 July, pp. 157–163. ACM, NY, USA.

[20] Goldman, C. V. and Zilberstein, S. (2004) Decentralized control of cooperative systems: Categorization and complexity analysis. *Journal of Artificial Intelligence Research*, **22**, 143–174.

[21] Monderer, D. and Shapley, L. S. (1996) Potential games. *Games and Economic Behavior*, **14**, 124–143.

[22] Chapman, A. C., Rogers, A., and Jennings, N. R. (2008) Benchmarking hybrid algorithms for distributed constraint optimisation games. *Proceedings of the 1st International Workshop on Optimisation in Multi-Agent Systems (OptMas–08)*, Estoril, Portugal, 10 May, pp. 1–11.

[23] Chapman, A. C., Micillo, R. A., Kota, R., and Jennings, N. R. (2009) Decentralised dynamic task allocation: A practical game–theoretic approach. *Proceedings of the 8th Conference on Autonomous Agents and Multiagent Systems (AAMAS–09)*, Budapest, Hungary, 10–15 May, pp. 915–922. IFAAMAS.

[24] Chapman, A. C. (2009) *Control of Large Distributed Systems Using Games with Pure Strategy Nash Equilibria*. PhD Thesis, School of Electronics and Computer Science, University of Southampton. http://eprints.ecs.soton.ac.uk/18234.

[25] Stankovic, J. A., Spuri, M., Ramamritham, K., and Buttazzo, G. C. (1998) *Deadline Scheduling for Real-time Systems*. Springer, Berlin.

[26] van Hoeve, W.-J., Gomes, C. P., Lombardi, M., and Selman, B. (2007) Optimal multi-agent scheduling with constraint programming. *Proceedings of the 19th Conference on Innovative Applications of Artificial Intelligence (IAAI–07)*, Vancouver, British Columbia, Canada, 22–26 July, pp. 1813–1818. AAAI Press / The MIT Press, MA, USA.

[27] Ottens, B. and Faltings, B. (2008) Coordinating agent plans through distributed constraint optimization. *ICAPS–08 Multiagent Planning Workshop*, Sydney, Australia, 14 September.

[28] Wellman, M. P., Walsh, W. E., Wurman, P. R., and MacKie-Mason, J. K. (2001) Auction protocols for decentralized scheduling. *Games and Economic Behavior*, **35**, 271–303.

[29] Koenig, S., Tovey, C., Zheng, X., and Sungur, I. (2007) Sequential bundle-bid single-sale auction algorithms for decentralized control. *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI–07)*, Hyderabad, India, 6–12 January, pp. 1359–1365. Elsevier, Amsterdam, Netherlands.

[30] Emery-Montemerlo, R., Gordon, G., Schneider, J., and Thrun, S. (2004) Approximate solutions for partially observable stochastic games with common payoffs. *Proceedings of the 3rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS–04)*, NY, USA, 19–23 July, pp. 136–143. IFAAMAS.

[31] Vetsikas, I. A. and Jennings, N. R. (2009). Bidding strategies for realistic multi–unit sealed–bid auctions. *Journal of Autonomous Agents and Multi-Agent Systems*. In press.

[32] Osborne, M. A., Rogers, A., Ramchurn, S., Roberts, S. J., and Jennings, N. R. (2008) Towards real-time information processing of sensor network data using computationally efficient multi–output Gaussian processes. *International Conference on Information Processing in Sensor Networks (IPSN–08)*, 22–24 April, pp. 109–120.

[33] Rahwan, T., Ramchurn, S. D., Giovannucci, A., and Jennings, N. R. (2009) An anytime algorithm for optimal coalition structure generation. *Journal of Artificial Intelligence Research*, **34**, 521–567.

[34] Ebden, M. ., Briers, M., and Roberts, S. (2008) Decentralized predictive sensor allocation. *Proceedings of the 47th IEEE Conference on Decision and Control (CDC–08)*, Cancún, Mexico, 9–11 December, pp. 1702–1707. IEEE.

[35] Chapman, A. C., Rogers, A., Jennings, N. R., and Leslie, D. S. (2009). A unifying framework for iterative approximate best response algorithms for distributed constraint optimisation problems. *The Knowledge Engineering Review*. In press.

[36] Chalkiadakis, G., Elkind, E., Polukarov, M., and Jennings, N. R. (2009) The price of democracy in coalition formation. *The 8th International Conference on Autonomous Agents and Multiagent Systems (AAMAS–09)*, Budapest, Hungary, 10–15 May, pp. 401–408. IFAAMAS.

[37] Rosenthal, R. W. (1973) A class of games possessing pure-strategy Nash equilibria. *International Journal of Game Theory*, **2**, 65–67.

[38] Krainin, M., An, B., and Lesser, V. (2007) An application of automated negotiation to distributed task allocation. *IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT–07)*, Silicon Valley, California, USA, 2–5 November, pp. 138–145. IEEE.

[39] Russell, S. and Norvig, P. (2002) *Artificial Intelligence: A Modern Approach*, 2nd edition. Prentice Hall, NJ, USA.