Efficient Inter-Team Task Allocation in RoboCup Rescue

Marc Pujol-Gonzalez
Jesus Cerquides
IIIA-CSIC
Campus de la UAB
Bellaterra, Spain
mpujol@iiia.cisc.es
cerquide@iiia.cisc.es

Alessandro Farinelli
Computer Science Department
University of Verona
Ca Vignal 2, Strada le Grazie 15
Verona, Italy
alessandro.farinelli@univr.it

Pedro Meseguer
Juan A. Rodriguez-Aguilar
IIIA-CSIC
Campus de la UAB
Bellaterra, Spain
pedro@iiia.cisc.es
jar@iiia.cisc.es

ABSTRACT

The coordination of cooperative agents involved in rescue missions is an important open research problem. We consider the RoboCup Rescue Simulation (RCS) challenge, where teams of agents perform urban rescue operations. Previous approaches typically cast such problem as separate single-team allocation problems. However, different teams have complementary capabilities, and therefore some kind of inter-team coordination is desirable for high-quality solutions. Our contribution considers inter-team coordination using Max-Sum. We present a methodology that allows teams in RCS to efficiently assess joint allocations. Furthermore, we show how to reduce the algorithm's computational complexity from exponential to polynomial time by using Tractable High Order Potentials. To the best of our knowledge this is the first time where it has been shown that MS can be run in polynomial time in the RCS challenge without relaxing the problem. Experiments with fire brigades and police agents show that teams employing inter-team coordination are significantly more effective than uncoordinated teams. Moreover, the evaluation shows that our BMS and THOPs method achieves up to 2.5 times better results than other state-of-the-art methods.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*multiagent systems, coherence and coordination*

General Terms

Algorithms, Performance

Keywords

multi-agent task allocation, factor graph, max-sum, robocup rescue

1. INTRODUCTION

In many practical applications, such as rescue, surveillance and environmental monitoring, agents with different capabilities must cooperate in dynamic and unpredictable environments [23]. Hence, the coordination problem faced by teams of rescue agents has been addressed in the literature from various perspectives and with a wide variety of solution techniques.

Appears in: *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2015), Bordini, Elkind, Weiss, Yolum (eds.), May 4–8, 2015, Istanbul, Turkey.*

However, most previous work either proposes approaches to enable teamwork between heterogeneous agents [25, 13, 10] or focuses on specific coordination techniques for a single team of agents (e.g., task assignment [22, 15] or resource allocation [12]).

In particular, a standard model for the task allocation problem in the context of rescue agent teams is the Extended Generalized Assignment Problem [22, 3]. However, such model cannot properly encode synergies and interferences among agents working on related tasks. For instance, EGAP cannot express that it is possible for two agents to perform the same task, but less desirable than letting them perform separate tasks (because otherwise they could interfere on each other). Since capturing such synergies is essential for effective cooperation in rescue missions, other works model such problem as a coalition formation problem [19, 20]. However, this approach suffers from the exponential growth of possible coalitions with the number of agents, hence its application to realistic, large-scale rescue scenarios is problematic.

Recently, a body of work focuses on ad-hoc teamwork [24, 9], where a team of agents must coordinate with very little or no prior information about their team mates (e.g., without having a shared communication protocol or knowing each other capabilities). Here, we take a different approach and focus on specific task assignment techniques for agents that might be part of different teams but have a relevant amount of prior information on each other capabilities, which is the typical case for urban rescue scenarios.

Task assignment problems in such settings can be conveniently formalized as a Distributed Constraint Optimization problem, as proposed in [8], where agents must compute a joint assignment that maximizes the team performance. For example, in the rescue domain, fire brigades must decide which fire to tackle so to mitigate damages to the buildings and civilians. While DCOPs offer a wealth of solution techniques, the dynamic nature of our reference scenario and the strict time constraints under which agents must operate favor the use of heuristic solutions [29, 11, 4]

An important advantage of using the DCOP model is that there exist several readily available heuristic approaches that typically provide good quality solutions. Moreover, the model is inherently distributed, providing the necessary robustness for such scenarios. In this perspective, we observe that the Max-Sum (MS) algorithm has been applied to a significant variety of application domains with successful results. Some examples include UAVs task assignment [2], Radar coordination [6], and task assignment in RoboCup Rescue [19, 8].

However, the use of MS for large scale applications and real-time constraints, such as the emergency response scenario we consider here, is significantly limited by the computational bottleneck associated with the message update operations. In particular, the MS algorithm has a computational complexity $O(d^n)$, where d is the

number of tasks that each agent may perform (i.e., the size of associated variable domain) and n is the number of agents that can perform a given task (i.e., the arity of the associated constraint). Such issue has been addressed with various approaches that includes modifications to the algorithm, i.e., the Fast-Max-Sum approach [19], or simplifications of the DCOP model, i.e., the pruning approach proposed in [8] or the function meta reasoning method proposed in [28].

Specifically, the Fast Max-Sum (Fast MS) algorithm proposed in [19] is equivalent (in terms of solution quality) to the MS approach, but reduces the computational effort from $O(d^n)$ to $O(2^n)$ (and the message size from d to 2). However, the computation remains exponential and hence it can not be applied to scenarios where the number of agents that can execute the same task is high, as it is the case in our fire-fighter coordination problem, where each fire-fighting unit can potentially work on every possible fire. In contrast, the approaches that simplify the DCOP model [8, 28], essentially consider, in the message update phase, only a sub-set of the possible agents that can perform a task while completely ignoring the others. While this has the important benefit of reducing the arity of the constraint (and hence decouple the computational complexity of the message update phase from the number of agent in the system), such approaches can not guarantee that the quality of the returned solution is the same as the one that the MS approach working on the original model would provide.

Now, MS originates and is widely used in the graphical models community, where its exponentiality issue was also considered simultaneously. There, Tarlow et al. have shown [26] that the computation associated to MS can be reduced to polynomial time (between $O(n)$ and $O(n \log n)$) for some specific types of factors (or constraints), known as *Tractable Higher Order Potentials* (THOPs). Crucially, this approach does not simplify the model on which the MS approach is defined. In contrast, it exploits the specific structure of THOPs hence providing a faster approach to update messages that is guaranteed to return the same solution that the standard MS approach would return. While not all DCOP functions can be represented using THOPs, previous works [16, 17] shows that THOPS are expressive enough for several application domains.

One important contribution of this paper is to propose a THOP only model for the task allocation problem related to the RCS challenge, hence obtaining a polynomial time approach that provides the same solution of the standard MS algorithm.

Now, a distinctive feature of THOPs is that they are defined over binary variables, and hence the approach is typically named Binary Max-Sum (BMS). While BMS is a promising approach for agent coordination, modeling large coordination problem using such approach is not straightforward nor intuitive. For example, usually a modeler finds much easier to work with a single d-valued variable that encodes which fire is a firefighter servicing than with d boolean variables encoding the same information. As the size of the problem grows, these issues become more evident, hampering the ability of the modeler to accurately do his work. Hence, it is crucial to devise effective design methodologies for BMS to become more widespread.

Against this background, in this paper we take an important step in that direction, by using BMS to enable effective multi-team coordination in the rescue scenario. Although the idea of coordinating different teams by means of coordination variables is not new (for instance in the loosely coupled planning literature [1], or the cooperative control community [21]), no methodology has been proposed up-to-date for building multi-team coordination models using BMS. To illustrate the methodology, we consider the realistic scenario represented by the RCS challenge, where a team of police officers and a team of firefighters must join forces to mitigate damages to a city after a natural disaster.

Furthermore our work shows that typical inter-team coordination interactions can be effectively modeled by means of THOPs allowing MS messages to be assessed in polynomial time in such complex scenarios.

The main contributions of this work are the following:

- We develop a THOP-only model for the firefighters task allocation problem associated to the RoboCup Simulation (RCS) domain. Despite previous work in this area [8, 19], to the best of our knowledge this is the first time where it has been shown that MS can be run in polynomial time for such task without simplifying the associated model (i.e., the DCOP representation).
- We present a methodology that eases the design of THOP-only models for large problems with several (interrelated) functional areas such as rescue teams. Following this methodology, we develop a THOP-only multi-team coordination model for the police and firefighters RCS problem.
- We empirically show that BMS obtains better results than other state-of-the-art DCOP algorithms (operating on a standard DCOP model), preventing more than twice as much damage to the city.

The rest of the paper is organized as follows. Section 2 provides background on MS and THOPs. Section 3 presents the RCS problem we tackle. Section 4 describes how the DCOP model to coordinate firefighters from [8] can be mapped to a THOP-only model. Section 5 presents our methodology to handle large problems, and develops a complete model for the police and firefighter forces. Section 6 reports our empirical findings. Finally, Section 7 concludes the paper.

2. BACKGROUND

MS is an approximate optimization algorithm that has been applied to different coordination problems with good empirical results. In this section we review MS and explain how, in some specific cases, its complexity can be reduced from exponential to polynomial time.

Let $T = \langle t_1, \ldots, t_n \rangle$ be a sequence of variables, with each variable t_i taking values in a finite set $\mathcal{D}_i$, its *domain*. The joint domain $\mathcal{D}_T$ is the cartesian product of the domain of each variable. We use $\mathbf{t_i}$ to refer to a possible assignment of t_i, that is $\mathbf{t_i} \in \mathcal{D}_i$ and $\mathbf{T}$ to refer to a possible joint assignment for variables in T, that is $\mathbf{T} \in \mathcal{D}_T$. Given a sequence of variables $W \subseteq T$, a factor c is a function $c : \mathcal{D}_W \to [-\infty, \infty)$. We refer to the variables in the scope of c as T_c. The set of factors of the problem is denoted by C. A function $g : \mathcal{D}_X \to \mathbb{R}$ is said to *decompose additively* if it can be broken as a sum of local terms, that is, whenever there is a set of local terms F (referred to as the additive decomposition F) such that $g(\mathbf{X}) = \sum_{f \in F} f(\mathbf{X}_f)$,

As an example, say we have three variables, x_1 with domain $\mathcal{D}_1 = \{0, 1\}$, x_2 with domain $\mathcal{D}_2 = \{0, 1, 2\}$ and x_3 with domain $\mathcal{D}_3 = \{1, 2\}$. The possible states for x_1 are $\mathbf{x}_1 \in \mathcal{D}_1 = \{0, 1\}$. We can define the function $g(x_1, x_2, x_3) = x_1 \cdot x_1 + x_1 \cdot x_2 - x_2 \cdot x_3$. It is easy to see that g decomposes additively by taking $F = \{f_1, f_2, f_3\}$, where $f_1(x_1) = x_1 \cdot x_1$, $f_2(x_1, x_2) = x_1 \cdot x_2$, and $f_3(x_2, x_3) = -(x_2 \cdot x_3)$.

The MS algorithm provides an approximate solution to the problem of maximizing a function that decomposes additively as a sum of functions with smaller scope:

$$\begin{array}{ll} \text{maximize} & g(\mathbf{X}) = \sum_{c \in C} c(\mathbf{T}_c) \\ \text{subject to} & \mathbf{t_i} \in \mathcal{D}_i \qquad\qquad \forall i \in \{1, \ldots, n\}\ , \end{array}$$

where $\mathbf{T_c}$ contains the values assigned by $\mathbf{T}$ to the variables in the scope of factor c.

MS operates on the so-called *factor graph*, a bipartite graph between variables and factors. It provides an approximate solution in two stages. First, messages are sent from variables to factors and from factors to variables. This step is repeated until the messages no longer change or a specified number of iterations is reached. After that, MS determines the best state for each variable independently.

At the first stage, MS assesses the message from variable t to factor c as

$$\mu_{t \to c}(\mathbf{t}) = \sum_{c' \in \mathcal{N}(t) \setminus \{c\}} \mu_{c' \to t}(\mathbf{t})\ , \tag{1}$$

where $\mathcal{N}(t)$ stands for the factors that have variable x in its scope and $\mu_{c' \to t}$ stands for the last message received by variable t from factor c'. MS assesses the message from c to t as

$$\mu_{c \to t}(\mathbf{t}) = \max_{\mathbf{W}} \left(c(\mathbf{t}, \mathbf{W}) + \sum_{w \in W} \mu_{w \to c}(\mathbf{w}) \right)\ , \tag{2}$$

where W is the set of variables in the scope of factor c excluding t, and $\mathbf{W}$ is a joint assignment for the variables in W.

At the second stage, MS assesses the preferred assignment for each variable as

$$\mathbf{t}^* = \arg\max_{\mathbf{t}} \sum_{c \in \mathcal{N}(t)} \mu_{c \to t}(\mathbf{t}).$$

The standard MS algorithm can be binarized as follows. A standard variable with d possible values becomes d boolean variables, all connected by a new factor that assures that only one boolean variable will be active. A standard factor involving k standard variables has (in the worst case) d^k entries. With boolean variables the number of entries increases up to $2^{d \times k}$. It is clear that both, standard MS and BMS have the same expressive power and can solve the same kind of problems.

Now, the computational bottleneck for MS is in equation 2, that takes time exponential in the number of variables in the scope of the factor. Recently, several works [16, 17, 26] have shown that for some specific types of factors, called Tractable Higher Order Potentials (THOPs), this time can be reduced to polynomial provided that all the variables in the scope are binary. In the following we consider binary variables and say that a variable is active (resp. inactive) if it is assigned to value 1 (resp. 0). It is worth noting that not every factor can be expressed using THOPs. However, it turns out that the RCS coordination problem we consider here can be formulated using THOPs only.

In particular, here we will employ two types of THOPs, *cardinality potentials* and *composite potentials*. Cardinality Potentials (CP) are factors whose value for an assignment depends on how many variables are active in that assignment, but does not depend on which particular variables are active. In general a CP can be expressed as $c(\mathbf{T_c}) = c'(n_c(\mathbf{T_c}))$, where $n_c(\mathbf{T}_c) = \sum_{t_i \in T_c} \mathbf{t_i}$ stands for the number of active variables in assignment $\mathbf{T_c}$. Tarlow et al. have shown [26] that for CPs, messages from equation 2 can be assessed in time $O(N \log N)$ time, where N is the number of variables in the scope of the factor. Particular cases of CPs are strict cardinality constraints such as *OneAndOnlyOne*, *AtMostOne* and *AllZeros*. Tarlow et al. [26] note that for some of these strict cardinality constraints, even more efficient algorithms to assess the messages are available, and, in particular, Pujol-Gonzalez et al. [17] shows that for *OneAndOnlyOne factors* the messages can be assessed in time $O(N)$. Finally, Tarlow et al. [26] also propose a technique for the composition of THOPs that allows to efficiently assess messages for any factor that can be constructed by the context-specific composition of smaller THOPs. These THOPs are known as *composite potentials*.

Figure 1: Example scenario.

3. PROBLEM DESCRIPTION

The RCS Platform is a benchmarking environment that simulates an urban search and rescue scenario where rescue forces (police patrols, ambulances and fire brigades) must coordinate their actions. Specifically, police patrols can unblock roads, fire brigades can extinguish fires, and ambulance agents can rescue trapped civilians. RCS creates a realistic simulation environment that presents significant aspects of dynamism (e.g., fires spread across a city), uncertainty (e.g., the behavior of fires is determined by a number of factors that may not be perfectly sensed or modeled), and issues of scale (e.g., tens of rescue agents and possibly hundreds of fires in a large urban area) [7].

In this paper we focus on the coordination problem of fire brigades, police patrols, and their interactions. Regarding fire brigades, a first element to consider is travel time: the closer a fire brigade is to a fire, the sooner they will be able to work on it. Moreover, the more fire brigades acting on one fire, the faster they will contain it. However, beyond a certain number of fire brigades, the contribution of each additional one is less significant. Now, a key issue for the fire fighting activity is that fires evolve and spread over time. A crucial insight on fire spreading is that new fires are more likely to extend than older ones, and older fires are fierier. Hence, fire brigades should try to prioritize new fires to prevent them from spreading as much as possible. Overall, fire brigades must cooperate to ensure that an adequate number of agents is allocated to each fire considering fire fieriness and travel time. Unlike fires, road blockades do not evolve over time unless some agent is acting on them. Moreover, in the version we used (2013), the simulation defines that having more police agents working on the same blockade is not beneficial. Hence, police patrols should spread out as much as possible to free roads as fast as they can.

Now, when we consider the whole picture, police patrols and fire brigades must coordinate their actions so that fire brigades can tackle important fires, which might be not reachable due to blockades, and police patrols should focus on blockades that might be far away but crucial for the fire fighting activity. For example, consider the scenario depicted in Figure 1, which will serve as a running example throughout our paper. In this scenario we have:

- two police patrols $P = \{\rho_1, \rho_2\}$ (blue circles);
- two fire brigades $A = \{\alpha_1, \alpha_2\}$ (red circles);

- four blockades $B = \{\beta_1, \beta_2, \beta_3, \beta_4\}$ (yellow circles) that prevent agents from transiting the road they are blocking; and
- two ignition points $F = \{\varphi_1, \varphi_2\}$, which are buildings that are on fire at the beginning of the simulation.

When considering the police patrol coordination problem without taking the fire fighting activity into account, a good allocation for this scenario would be $(\rho_1 \to \beta_4), (\rho_2 \to \beta_3)$ because both agents work on their closest blockade. However, if we consider the overall goal of the rescue agents (including fire fighters), a better allocation would be $(\rho_1 \to \beta_2), (\rho_2 \to \beta_1)$, enabling fire brigade agents to choose which fire to attend at their will.

4. THE FIRE BRIGADES TEAM MODEL

In this section we develop a THOP-only model for the coordination of the firefighters team, disregarding police forces entirely. With this aim, we first present an improved version of the general DCOP model in [8]. Thereafter we show how this model can be binarized, thus allowing us to run BMS instead of MS and reducing the complexity from exponential to linearithmic time.

4.1 DCOP model

Solving the firefighters problem amounts to deciding which fire each fire brigade should attend. These decisions can be encoded by a set of decision variables $Y = \{y_a | a \in A\}$, where each variable y_a takes values in F. Note that $y_{\alpha_1} = \varphi_2$ means that brigade α_1 is assigned to fire φ_2. Thus, our objective is to find an assignment $\mathbf{Y}$ that maximizes the team utility $u(\mathbf{Y})$, that is

$$\begin{array}{ll} \text{maximize} & u(\mathbf{Y}) \\ \text{subject to} & \mathbf{y}_a \in F. \qquad \forall a \in A \end{array}$$

In a DCOP model, the team's utility is expected to be decomposed as a sum of factors. In our case, a natural decomposition is to introduce two kinds of factors:

- *fire factors*, that specify the gain obtained when firefighters are allocated to some fire; and
- *cost factors*, that specify the cost for agents to reach different fires.

Thus, our team utility is

$$u(\mathbf{Y}) = \sum_{f \in F} e_f(\mathbf{Y}) - \sum_{f \in F} \sum_{a \in A} r_{af}(\mathbf{y_a}), \tag{3}$$

where $e_f(\mathbf{Y})$ are fire factors and $r_{af}(\mathbf{y_a})$ are cost factors.

4.1.1 Fire factors

As explained in Section 3, some fires are more relevant than others. Hence, to asses $e_f(\mathbf{Y})$, we first compute a value v_f for each fire f, corresponding to the utility obtained by assigning a brigade to put it out. Next, notice that more than one brigade can be assigned to the same fire. A simple model for e_f is to multiply v_f by the number of brigades that are assigned to fire f, namely $n_f(\mathbf{Y})$. Nonetheless, depending on the fieriness and size of a fire, there is a threshold t_f in the number of fire brigades that can successfully cooperate in extinguishing it. Thus, we consider that an assignment of fire brigades to a fire f is penalized when more than t_f fire brigades are assigned to fire f. Moreover, this penalty increases with the number of additional agents beyond the threshold. Combining all these assessments, $e_f(\mathbf{Y})$ is calculated as

$$e_f(\mathbf{Y}) = v_f \cdot n_f(\mathbf{Y}) - \kappa \cdot [\max(0, n_f(\mathbf{Y}) - t_f)]^\gamma, \tag{4}$$

where $\kappa > 0$ and $\gamma \geq 1$ are arbitrary coefficients.

4.1.2 Cost factors

All fire brigades are equally capable in RCS. Thus, the cost for agents to reach each fire depends only on their distance and whether that fire is reachable or not. As a result, we evaluate the cost of assigning a brigade to a fire as proportional to the square of the distance between them.[1] Additionally, a blockade may prevent fire brigade a from reaching fire f. We discourage choosing blocked fires by introducing an additional cost M when a cannot reach f. Thus, r_{af} is assessed as

$$r_{af}(\mathbf{y_a}) = \begin{cases} \nu d_{af}^2 + o_{af} M & \text{if } \mathbf{y_a} = f \\ 0 & \text{otherwise} \end{cases}, \tag{5}$$

where d_{af} is the normalized distance between brigade a and fire f, $\nu \geq 0$ is an arbitrary coefficient, and o_{af} is a constant with value 1 when agent a cannot reach fire f or 0 otherwise.

At this point we have a DCOP model. We can now compute allocations using MS by instantiating the e_f and r_{af} factors and exchanging messages between them and the variables in Y. However, recall that computing a factor's messages in MS takes exponential time on the number of variables involved in that factor. Because the e_f factors depend on all of the problem's variables, running MS on this model takes an exponential time on the number of fire brigades. From this follows that for most scenarios MS can not compute a solution within the one second time limit enforced by the RoboCup simulator.

4.2 THOP-only model

Next we show how to exactly encode the previous model in a binary form and only using THOPs. As a result, we will be able to run BMS in polynomial, instead of exponential, time. With this aim, we first split each decision variable y_a into $|F|$ binary variables z_{af}. Variable z_{af} is active (set to 1) in a solution whenever agent a is assigned to fire f, and it is inactive (set to 0) otherwise. Unlike our representation above, with this set of binary variables we can encode that a single brigade is assigned to two or more different fires simultaneously. Hence, we must add a set of constraints (one per brigade) to prevent this from happening:

$$\sum_{f \in F} \mathbf{z_{af}} = 1 \quad \forall a \in A\,. \tag{6}$$

With a slight abuse of notation, we now redefine the factors of the DCOP model to operate over the binary variables in Z instead of the n-ary variables in Y. Likewise equation (3), the utility function $u(\mathbf{Z})$ is split into fire factors e_f and cost factors r_{af}. The binarized versions of the utility functions in equations (4) and (5) are the following:

$$e_f(\mathbf{Z_{.f}}) = v_f \cdot n_f(\mathbf{Z_{.f}}) - \kappa \cdot [\max(0, n_f(\mathbf{Z_{.f}}) - t_f)]^\gamma \tag{7}$$

$$r_{af}(\mathbf{z_{af}}) = \mathbf{z_{af}}(\nu d_{af}^2 + o_{af} M) \tag{8}$$

where $Z_{.f} = \{z_{af} | a \in A\}$ is the set of variables that relate to fire f.

At this point, we can represent the entire firefighters coordination problem as the following optimization problem:

$$\begin{array}{ll} \text{maximize} & \sum_{f \in F} e_f(\mathbf{Z_{.f}}) - \sum_{f \in F} \sum_{a \in A} r_{af}(\mathbf{z_{af}}) \\ \text{subject to} & \sum_{f \in F} \mathbf{z_{af}} = 1 \qquad \forall a \in A \\ & \mathbf{z_{af}} \in \{0, 1\} \qquad \forall a \in A \ \forall f \in F \end{array}$$

[1] We normalize distances so that the largest distance between any two points in a scenario is 1.

Notice that a fire factor e_f does not depend on the specific fire brigades attending it, but only on how many. Hence e_f factors fulfill the condition to be Cardinality Potentials, and their BMS messages can be computed in $O(N \log(N))$ time using the procedure described in [26]. Likewise, the consistency constraints in equation (6) are *OneAndOnlyOne* THOPs (as described in Section 2), and the agent factors r_{af} depend only on one variable.

As a result, it is now possible to assess an approximate solution to our problem by applying BMS to the THOP-only model. Furthermore, the complexity of each iteration of the algorithm is reduced from the $O(|F||F|^{|A|})$ time of MS over the DCOP model to $O(|F||A| \log |A|)$ time of BMS with the THOP-only model.

5. INTER-TEAM COORDINATION

In the rescue scenario we consider here, a single-team (i.e., the fire-fighters) can take better decisions by considering the operations of other rescue teams (i.e., the police forces). Consider our example in Figure 1. Fire brigades alone will try to avoid blocked fires. However, this completely disregards the fact that police agents will be removing blockades in the meantime so that fire brigades may be able to reach blocked fires in the near future. To capture the interdependencies between teams' decisions, we argue that it is necessary to perform inter-team coordination.

Here we present a methodology to enact inter-team coordination to make team decisions considering a shared goal. Our methodology is intended to help the designer build a representation of the complete inter-team coordination problem as a single utility function. This is not an easy endeavor because, as the number of "teams" grows, the global utility function becomes more and more complex, possibly becoming unmanageable. Our methodology proposes a modular construction of the global utility function following the next steps:

1. *Define independent coordination models* for each team involved in the inter-team coordination.

2. *Identify the coordination objects* that capture the interdependencies between teams. Such objects will serve to create coordination variables, which are meant to act as interfaces between single-team coordination models.

3. *Extend single-team coordination models* to connect them to the coordination variables.

At the end of this process, the global utility function is readily obtained by simply adding up the extended single-team coordination models into a single function. Since, as we show below, the resulting global utility function decomposes additively as a sum of functions, the teams involved will be able to apply MS to assess their decisions. A distinctive advantage of our methodology is that, once coordination variables are defined, the designer does not need to consider the whole inter-team coordination problem anymore. That is, each team independently connects its intra-team coordination model with the coordination variables. Therefore, our methodology avoids the design complexity explosion.

Next we exemplify the application of our methodology to the coordination of a team of fire brigades and a team of police forces. For space reasons, we only present the THOP model in this part.

5.1 Define Independent Coordination Models

The first step in our methodology consists in separately defining the coordination models for each individual team involved in inter-team coordination. In Section 4 we already introduced a coordination model for a team of fire brigades. Hence, we just need to develop a coordination model for a team of police forces.

The Police Team Model

Recall from section 3 that police patrols can remove blockades from roads, freeing the paths for other types of agents to move along. Hence, it is critical that policemen coordinate between them to remove blockades as quickly as possible. Thus, the coordination problem faced by the policemen team is to decide the assignment of patrols to blockade removal tasks. As in the case of fire brigades and fires, we encode an allocation of patrols to blockades using a set of binary variables $X = \{x_{pb} | p \in P, b \in B\}$ where x_{pb} is active if patrol p is assigned to blockade b and inactive otherwise. Obviously, a patrol can not remove more than one blockade at a time. Also, in RCS multiple patrols cannot work on the same blockade at the same time. Hence, the goal of the police team coordination problem is to compute the best allocation of patrols to blockades where each patrol is assigned to at most one blockade and each blockade is not assigned to more than one policemen. Thus, the following constraints must be enforced:

$$\sum_{b \in B} \mathbf{x_{pb}} \leq 1 \quad \forall p \in P \tag{9}$$

$$\sum_{p \in P} \mathbf{x_{pb}} \leq 1 \quad \forall b \in B \tag{10}$$

Similar to fire brigades, the utility of an allocation $u(\mathbf{X})$ can be decomposed in *blockade factors* e_b and *cost factors* r_{pb}, namely

$$u(\mathbf{X}) = \sum_{b \in B} e_b(\mathbf{X_{.b}}) - \sum_{b \in B} \sum_{p \in P} r_{pb}(\mathbf{x_{pb}}). \tag{11}$$

Since all patrols are assumed equally capable in RCS and blockades do not have distinguishing characteristics, we assign a utility $v_B > 0$ to attending any blockade. This utility is obtained whenever some patrol is assigned to remove that blockade. Thus:

$$e_b(\mathbf{X_{.b}}) = \begin{cases} v_B & \text{if } n_b(\mathbf{X_{.b}}) \geq 1 \\ 0 & \text{otherwise.} \end{cases} \tag{12}$$

where $n_b(\mathbf{X}_{.b})$ is the number of patrols assigned to blockade b.

The cost of assigning a patrol p to service blockade b is analogous to what we did for the fire brigades. First, we introduce a constant o_{pb} that is 0 if the blockade is directly accessible to the patrol (no other blockade appears in the path between p and b), or 1 if the path from p to b is obstructed by some other blockade. Then the cost is proportional to the square of the distance d_{pb} between the patrol and the target blockade, with an additional penalty Q if the path is obstructed:

$$r_{pb}(\mathbf{x_{pb}}) = \mathbf{x_{pb}}(d_{pb}^2 + o_{pb}Q) \ . \tag{13}$$

With these definitions, we can represent the full police forces coordination model as the following optimization problem:

$$\begin{array}{lll} \text{maximize} & \sum_{b \in B} e_b(\mathbf{X_{.b}}) - \sum_{b \in B} \sum_{p \in P} r_{pb}(\mathbf{x_{pb}}) & \\ \text{subject to} & \sum_{b \in B} \mathbf{x_{pb}} \leq 1 & \forall p \in P \\ & \sum_{p \in P} \mathbf{x_{pb}} \leq 1 & \forall b \in B \\ & \mathbf{x_{pb}} \in \{0, 1\} & \forall p \in P \ \forall f \in F. \end{array}$$

This problem can also be entirely encoded with THOPs. The reasoning for the e_b and r_{pb} factors is analogous to the e_f and r_{af}

factors in the previous section. Then, the constraint in equation (9) can be mapped to a factor of the form

$$AtMostOne_p(\mathbf{X}_{\cdot p}) = \begin{cases} 0 & \text{if } \sum_{b \in B} \mathbf{x}_{pb} \leq 1 \\ -\infty & \text{otherwise,} \end{cases} \quad (14)$$

and the same holds for the constraint in equation (10).

Although no reference to the *AtMostOne* factor as being a THOP appears in the literature, the expressions for the messages going out from it are simple and can be derived similarly to those of the *OneAndOnlyOne* factor in [17]. The assessment of the messages for an *AtMostOne* factor in a BMS iteration can then be done in time $O(|B|)$.

5.2 Identify the Coordination Objects

In our RoboCup example the coordination objects between fire brigades and policemen are blockades. On the one hand, police forces should prioritize blockades that are actually preventing fire brigades from performing their duties. On the other hand, fire brigades would like to know which blockades will be removed in the near future to make better decisions. Thus, we create a binary coordination variable c_b for each blockade b as a means of representing the coordination objects relating police patrols and fire brigades. The coordination variable for a blockade b must become active whenever the blockade is to be removed in the near future, or inactive otherwise.

In our particular example, these variables represent everything our police forces and fire brigades need to know to coordinate with each other. In other words, the coordination variables can be understood as representing the *common language* between our individual teams. Such language is intended to enable the fire brigades team and the policemen team to exchange information about their interdependencies regarding blockades.

5.3 Extending Single-Team Models

The third step in our methodology is to extend the independent team models by connecting them to the coordination variables. Hereafter we extend both the fire brigades team model and the police patrols team model to take coordination variables into account.

5.3.1 Extending the Fire Brigades Team Model

Fire brigades can modify their utility function provided that they know which blockades will be removed by police patrols. In particular, the penalty associated to a blocked fire should be removed whenever police patrols are planning to remove the blockade that prevents the fire brigade from accessing it. The interface between the fire brigades model and the coordination variables can be done by simply adding an additional factor s_{afb} whenever fire brigade a is being prevented from reaching fire f by blockade b.

$$s_{afb}(\mathbf{z_{af}}, \mathbf{c_b}) = \begin{cases} M & \text{if } \mathbf{c_b} \text{ is active and } \mathbf{z_{af}} \text{ is active} \\ 0 & \text{otherwise} \end{cases}. \quad (15)$$

The role of this factor is essentially to cancel out the penalty in equation 5 when the blockade b that is preventing fire fighter a to reach fire f is being attended by some police agent.

5.3.2 Extending the Police Team Model

The internal variables of the police team should be consistent with the semantics of the coordination variables c_b above. Specifically, c_b must only be active if some police agent is attending b. That is, variable c_b is an indicator of whether any of the variables in $X_{\cdot b}$ are active. We can enforce this by adding a new *Indicator* factor $I_b(\mathbf{c_b}, \mathbf{X_{\cdot b}})$ for each blockade, defined as

$$I_b(\mathbf{c_b}, \mathbf{X_{\cdot b}}) = \begin{cases} 0 & \text{if } \mathbf{c_b} = 1 \text{ and } \sum_{p \in P} x_{pb} \geq 1 \\ 0 & \text{if } \mathbf{c_b} = 0 \text{ and } \sum_{p \in P} x_{pb} = 0 \\ -\infty & \text{otherwise} \end{cases} \quad (16)$$

I_b is not known to be a THOP, but we can derive expressions for its messages by noticing that it is a composite factor[2] where c_b defines two partitions depending on whether it is active or not. When c_b is active, the factor is a *Cardinality Potential* that yields a utility of 0 when exactly one of the $X_{\cdot b}$ variables is active, or $-\infty$ otherwise. When c_b is inactive, the factor is an *AllInactive*[3] factor between the variables in $X_{\cdot b}$. Since the *Indicator* potential is a composite factor, and in each of the partitions defined by c_b we have a THOP, we can efficiently assess the messages out of this factor in time $O(|B| \log |B|)$.

At this point, we can construct the global utility function for inter-team coordination. This results from adding the objective function that results from extending the fire brigades team coordination model with the objective function that results from extending the policemen team coordination model. Notice that such global utility function represents the whole problem. Since this function has been built as an additive composition of functions, we can readily apply BMS to solve the inter-team coordination problem. The execution of BMS yields an exchange of information from team to team regarding coordination variables. Messages from brigades to patrols will represent how much interested brigades are in police forces removing a blockade, whereas messages from patrols to brigades convey the police team's cost of removing a blockade.

6. EMPIRICAL EVALUATION

In this section we empirically evaluate the performance of our task allocation mechanism for the fire and police teams. With this aim, we compare BMS with the methods implemented in the RMAS-Bench [8] platform, namely DSA and Greedy. DSA is a local search algorithm executed in parallel by the participant agents. It employs a stochastic parameter, DSA_p, to control the amount of parallelism allowed while running. This simple algorithm provides surprisingly good results in many different problems, and its main advantage is that it requires very low computation and communication efforts. In contrast, Greedy represents a simple greedy allocation where each agent chooses the target that maximizes its individual utility, without coordinating at all. These algorithms operate over a standard (n-ary) DCOP model that suits them better and is equivalent to the binary one presented here. We omit the details for space reasons, but the entire source code used in this evaluation is available for the interested reader [18].

We do not compare against standard MS because as explained in [8] and in section 4.1.2, it incurs in exponential costs and fails to assess a solution within the one second time limit enforced by the simulator[4]. Although Fast MS [20] reduces this exponentiality from $O(d^n)$ to (2^n), unfortunately it is still unable to fulfill this time limit. Furthermore, note that BMS will assess the same solution as both MS and Fast MS only at a lower cost.

6.1 Experimental setup

We run experiments on the standard scenarios used in the 2013 RoboCup competition, namely Paris and Kobe. However, we dis-

[2]A composite potential in [26],

[3]The messages for the *AllInactive* factor are trivial to derive. The messages to all its variables are simply $-\infty$.

[4]As explained in section 1, they tame MS costs by relaxing version of the problem. In contrast, we work on the unrelaxed problem.

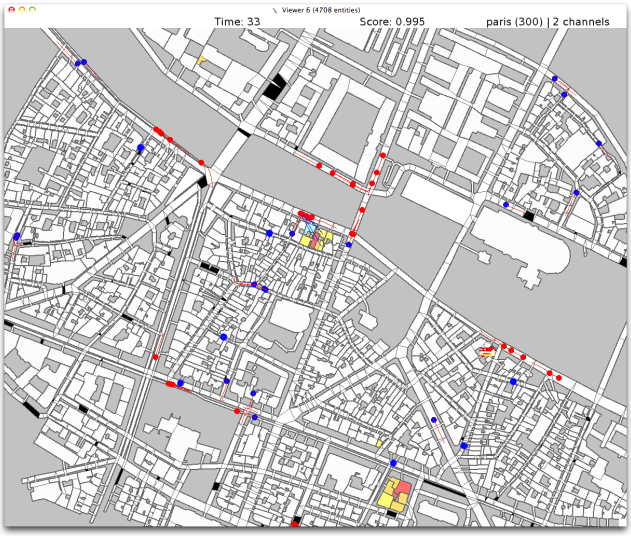


Figure 2: RCS viewer showing a running simulation. Red dots are fire brigades, blue dots are police patrols, blacked out areas are blocked roads and colored buildings are buildings on fire.

card the scenarios' elements that are irrelevant for our evaluation, i.e., everything but ignition points, police forces and fire brigades. The algorithms have one second to compute an allocation at each simulation step, and the simulation finishes either when all fires are extinguished or after 300 steps. Additionally, we randomly block 5% of the roads at the beginning of the simulation. Hence, the order in which police forces remove these blockades may have a noticeable impact on the results, depending on how well the coordination mechanism works.

After an extensive empirical evaluation, we fixed the utility function's parameters to κ=2, γ=1.4, and ν=10, because these provided the best results for all algorithms. Moreover, the utility of each fire is $v_f = 4 - I_f$, where $I_f \in \{1, 2, 3\}$ is the fieriness of fire f as reported by the simulator, and t_f is the area of fire f divided by 100. Intuitively, the fierier a fire is, the longer a building has been burning, and the less valuable it is to contain. Additionally, we scale all the utilities of the police forces model by 10^{-3} to give more relevance to the fire brigades team than to the police agents team. With the same objective, we set $M = 100$, and $Q = 50$, so that blockades preventing fire brigades from reaching fires are more important than blockades in the path of police agents.

Regarding algorithms' parameters we set the number of maximum iterations for DSA and BMS to 100. Lowering the number of iterations to 50 slightly decreased the quality of both algorithms, whereas increasing them to 1000 did not improve the results, wasting resources in both cases. We also employ the Anytime framework from [30] to keep track of the best global solution (assignment) each algorithm has found during all the iterations, and use that as the final result. After testing with DSA_p values ranging from 0.1 to 0.9 (in 0.1 increments), we chose 0.1 because it yielded the best results overall. In our DSA implementation agents always prefer to switch tasks whenever they are in conflict. In fact, agents tend to switch too much if a large p is used, explaining the rather low DSA_p value. It is well known that MS has problems to reach convergence on factor graphs containing many loops. To overcome these issues, we use a damping factor [5] of 0.9 in BMS. [5]

[5] The use of a damping factor δ in max-sum is a well-known technique that helps stabilize the algorithm. When using damping, each sent message $\hat{\mu}^i_{c\to t}$ is a weighted average between the latest computed message and the message sent in the previous iteration, namely $\hat{\mu}^i_{c\to t} = \delta\mu^i_{c\to t} + (1-\delta)\hat{\mu}^{i-1}_{c\to t}$.

Table 1: Statistics for Greedy, DSA and BMS averaged over 30 runs in the Paris scenario (agents start acting after 25 iterations).

Algorithm	Greedy	DSA	BMS
Score	5.29±0.79 %	2.94±0.43 %	1.13±0.18 %
NCCCs	0.00±0.00 k	7.51±0.11 k	79.62±0.81 k
Num. Msgs	0.00±0.00 k	91.08±0.92 k	536.31±8.84 k
Total bytes	0.00±0.00 Kb	711.60±7.16 Kb	4189.92±69.03 Kb
CPU time	16.14±0.45 ms	220.91±7.02 ms	726.37±15.09 ms

Figure 2 shows a snapshot of a running simulation in the Paris map. The white areas represent buildings and roads. A blacked out road is currently blocked, and cannot be transited until some police patrol (blue dot) clears it. In contrast, colored buildings represent fires that the firefighters (red dots) must extinguish.

Against this background, we use the following metrics to evaluate the performance of the different algorithms:

- *Score*: is the main performance metric used by the RoboCup simulator. It evaluates the percentage of damage suffered by the city, with 100% meaning that it has been completely destroyed;
- *NCCCs [14]*: captures the per-iteration average amount of non-parallelizable computation performed by the agents;
- *Num msgs*: tracks the average number of messages sent between all agents in a single iteration;
- *Total bytes*: is the average number of bytes per iteration sent between all agents.

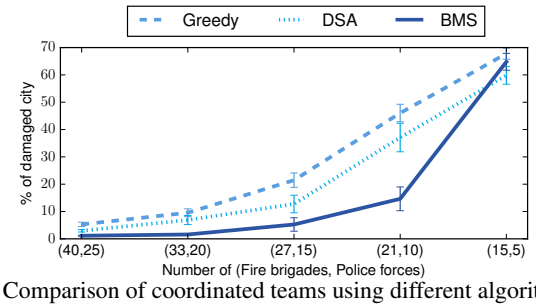
In the following section we summarize the main results of our experimental evaluation.

6.2 Results

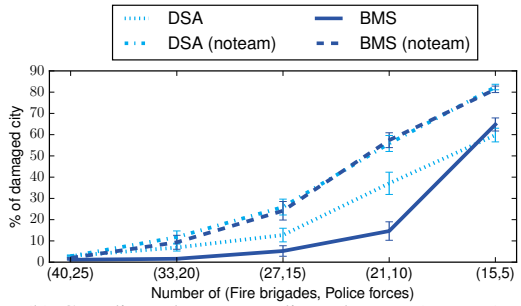
Table 1 shows the results we obtained on the Paris map with a start time of 25 simulation steps[6] averaged over 30 simulations. BMS clearly achieves the best score, with only 1.13% of the city damaged. In contrast, more than twice as many buildings burn down when using DSA, and more than four times with the Greedy algorithm. This gain in quality comes at a cost though. Greedy agents obtain the worst results in quality but require no coordination resources. DSA requires few computational resources and relatively low communications, whereas BMS computes an order of magnitude more than DSA, and requires substantially more bandwidth. Nonetheless, taking into account that an iteration of the RCS represents one minute of real time, all these costs are within an acceptable range. We also experimented with a closest allocation method, where an agent is assigned to the closest task. Such method has been frequently used in the RCS as a benchmarking approach for task allocation (e.g., see [15]). With that method fire agents spend too much time watering down old fires (which are unlikely to spread), and an average of 49% of the city gets damaged. This result indicates that our utility function is properly capturing the characteristics of the problem.

Next we assess the behavior of the different algorithms when the amount of fire and police agents decreases. Overall, Figure 3(a)

[6] This prevents agents from executing any action for 25 time steps, so that fires have time to spread.



(a) Comparison of coordinated teams using different algorithms



(b) Coordinated vs uncoordinated teams (noteam)

Figure 3: Performance comparison when decreasing the agent resources. Results are averaged over 30 runs and error bars represent the standard error of the mean.

shows that BMS coordinated agents achieve significantly better results than DSA and Greedy in all but the worst conditions. Namely, BMS prevents around 2.5 times more damages than DSA, and around 4 times more damages than Greedy. This trend continues when the amount of available police and fire brigades decreases. For instance, BMS saves 85.4% of the city when there are only 21 fire brigades and 10 police patrols, whereas DSA saves only 62.9% of the buildings and Greedy an even lower 54%. However, there is a point where there are so few agents that the situation is helpless and most of the city gets damaged. In this scenario, we observe this effect when we reduce the resources to 15 fire brigades and 5 police agents. At this point all algorithms obtain fairly similar results, and DSA becomes the best strategy thanks to its greedy but still coordinated nature.

While Figure 3(a) shows that both DSA and BMS outperform the Greedy strategy given a sensible number of rescue agents, we cannot be sure whether this is purely because of the algorithm or thanks to the inter-team coordination. Therefore, we repeated the above experiments using BMS and DSA, but now without the inter-team coordination constraints. Figure 3(b) shows the results we obtained, where "DSA (noteam)" and "BMS (noteam)" represent the corresponding algorithms but without the inter-team constraints. The results are particularly revealing. On the one hand, the algorithms without inter-team coordination perform similarly. BMS provides slightly better results when there are more agents, but the differences become insignificant (according to a Wilcoxon signed-rank [27] test with $p = 0.01$) when the number of agents decreases past 27 fire brigades and 15 police patrols. On the other hand, both algorithms perform notably better when employing our inter-team coordination constraints, thus validating our inter-team methodology and the resulting model.

Furthermore, the gains from inter-team coordination are clearly larger for BMS than for DSA. This difference has a simple explanation. Notice that the inter-team constraints require an agent (especially police patrols) to temporarily worsen its own individual outcome (by attending a farther blockade) to realize the inter-team gains (to allow a firefighter to reach a more important fire). Now recall that DSA is essentially a greedy algorithm. In contrast, Max-Sum is the only known local-state DCOP algorithm that does not operate in a greedy manner. Therefore, the Max-Sum algorithm (and BMS by extension) is intrinsically better equipped to exploit these coordination situations where some temporary individual sacrifice must be made to achieve a greater outcome.

Finally, we conducted similar experiments in the 2013's Kobe scenario. The observed trends are the same, but the differences between algorithms are smaller because the problem is easier: in Kobe there is only one fire focus and the map is small and much easier to navigate.

7. CONCLUSIONS

We presented a methodology that allows multiple teams to make joint allocations by enabling them to coordinate during the task allocation process. Using this methodology, we developed a model for inter-team coordination of firefighters and police forces. We have shown that some typical inter-team coordination interactions can be effectively modeled by means of THOPs allowing MS messages to be assessed in polynomial time in such complex scenarios. To the best of our knowledge this is the first time where it has been shown that MS can be run in polynomial time without simplifying the associated model. Experiments with fire brigades and police agents show that teams employing inter-team coordination are significantly more effective than uncoordinated teams. Moreover, the evaluation shows that our BMS and THOPs method achieves up to 2.5 times better results than other state-of-the-art methods. Furthermore, the gains from inter-team coordination are clearly larger for BMS than for alternative algorithms with a greedy inspiration. Therefore, we have seen that the Max-Sum algorithm (and BMS by extension) is intrinsically better equipped to exploit these coordination situations.

Acknowledgments

Work funded by projects DAMAS (TIN2013-45732-C4-4-P), COR (TIN2012-38876-C02-01), the Generalitat of Catalunya grant 2009-SGR-1434, and the Ministry of Economy and Competitivity grant BES-2010-030466.

REFERENCES

[1] Ri Brafman and Carmel Domshlak. From One to Many: Planning for Loosely Coupled Multi-Agent Systems. *International Conference on Automated Planning and Scheduling*, pages 28–35, 2008.

[2] F. M. Delle Fave, A. Rogers, Z. Xu, S. Sukkarieh, and N. R. Jennings. Deploying the max-sum algorithm for decentralised coordination and task allocation of unmanned aerial vehicles for live aerial imagery collection. In *2012 IEEE International Conference on Robotics and Automation*, pages 469–476, 2012.

[3] F. dos Santos and A.C. Bazzan. Towards efficient multiagent task allocation in the robocup rescue: a biologically-inspired approach. *Autonomous Agents and Multi-Agent Systems*, 22(3):465–486, 2011.

[4] A. Farinelli, A. Rogers, A. Petcu, and N. R. Jennings. Decentralised coordination of low-power embedded devices using the max-sum algorithm. In *Proceedings of the 7th international joint conference on Autonomous agents and*

multiagent systems-Volume 2, pages 639–646. International Foundation for Autonomous Agents and Multiagent Systems, 2008.

[5] B. Frey and D. Dueck. Clustering by passing messages between data points. *Science*, 315(5814):972–976, 2007.

[6] Y. Kim, M. Krainin, and V. Lesser. Application of max-sum algorithm to radar coordination and scheduling. In *Workshop on Distributed Constraint Reasoning*, 2010.

[7] H. Kitano, S. Tadokoro, I. Noda, H. Matsubara, T. Takahashi, A. Shinjou, and S. Shimada. Robocup rescue: Search and rescue in large-scale disasters as a domain for autonomous agents research. In *Systems, Man, and Cybernetics, 1999. IEEE SMC'99 Conference Proceedings.*, volume 6, pages 739–743. IEEE, 1999.

[8] A. Kleiner, A. Farinelli, S. Ramchurn, B. Shi, F. Maffioletti, and R. Reffato. Rmasbench: benchmarking dynamic multi-agent coordination in urban search and rescue. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pages 1195–1196. International Foundation for Autonomous Agents and Multiagent Systems, 2013.

[9] Somchaya Liemhetcharat and Manuela Veloso. Modeling and learning synergy for team formation with heterogeneous agents. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 365–374. International Foundation for Autonomous Agents and Multiagent Systems, 2012.

[10] Somchaya Liemhetcharat and Manuela Veloso. Weighted synergy graphs for effective team formation with heterogeneous ad hoc agents. *Artificial Intelligence*, 208:41–65, 2014.

[11] R. T. Maheswaran, J. P. Pearce, and M. Tambe. Distributed algorithms for DCOP: A graphical game-based approach. In *Proceedings of the Seventeenth International Conference on Parallel and Distributed Computing Systems*, pages 432–439, 2004.

[12] R. Mailler, V. Lesser, and B. Horling. Cooperative negotiation for soft real-time distributed resource allocation. In *Proceedings of AAMAS'03*, pages 576–583, 2003.

[13] Leandro Soriano Marcolino, Albert Xin Jiang, and Milind Tambe. Multi-agent team formation: diversity beats strength? In *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, pages 279–285. AAAI Press, 2013.

[14] A. Meisels, E. Kaplansky, I. Razgon, and R. Zivan. Comparing performance of distributed constraints processing algorithms. In *Proc. AAMAS-2002 DCR Workshop*, pages 86–93, 2002.

[15] James Parker and Maria Gini. Tasks with cost growing over time and agent reallocation delays. In *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-agent Systems*, AAMAS '14, pages 381–388, Richland, SC, 2014. International Foundation for Autonomous Agents and Multiagent Systems.

[16] T. Penya-Alba, J. Cerquides, J. A. Rodriguez-Aguilar, and M. Vinyals. A Scalable Message-Passing Algorithm for Supply Chain Formation. In *AAAI*, pages 1436–1442, 2012.

[17] M. Pujol-Gonzalez, J. Cerquides, P. Meseguer, J. A. Rodriguez-Aguilar, and M. Tambe. Engineering the decentralized coordination of UAVs with limited communication range. In *CAEPIA*, 2013.

[18] Marc Pujol-Gonzalez, Alexander Kleiner, Alessandro Farinelli, Sarvapali Ramchurn, Bing Shi, Fabio Maffioletti, and Riccardo Reffato. RMASBench: Multi-agent coordination benchmark. https://github.com/RMASBench/RMASBench, 2012–2015.

[19] S. D. Ramchurn, A. Farinelli, K. S. Macarthur, and N. R. Jennings. Decentralized coordination in robocup rescue. *The Computer Journal*, 53(9):1447–1461, 2010.

[20] S. D. Ramchurn, M. Polukarov, A. Farinelli, N. Jennings, and C. Trong. Coalition formation with spatial and temporal constraints. In *International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2010)*, pages 1181–1188, 2010.

[21] Wei Ren, Randal W. Beard, and Timothy W. McLain. Coordination Variables and Consensus Building in Multiple Vehicle Systems. *Lecture Notes in Control and Information Sciences*, 309/2005:439–442, 2005.

[22] P. Scerri, A. Farinelli, S. Okamoto, and M. Tambe. Allocating tasks in extreme teams. In *Proc. of AAMAS 05*, pages 727–734, Utrecht, Netherland, 2005.

[23] N. Schurr, J. Marecki, P. Scerri, J. P. Lewi, and M. Tambe. *Programming Multiagent Systems*, chapter The DEFACTO System: Coordinating Human-Agent Teams for the Future of Disaster Response, page 296. Springer, 2005.

[24] P. Stone, G. Kaminka, S. Kraus, and J. Rosenschein. Ad hoc autonomous agent teams: Collaboration without pre-coordination. In *Proceedings of the Twenty-Fourth Conference on Artificial Intelligence*, July 2010.

[25] M. Tambe. Towards flexible teamwork. *Journal of Artificial Intelligence Research (JAIR)*, 7:83–124, 1997.

[26] D. Tarlow, I. E. Givoni, and R. S. Zemel. HOP-MAP : Efficient Message Passing with High Order Potentials. In *13th AISTATS*, volume 9, pages 812–819, 2010.

[27] F Wilcoxon. Individual comparisons of grouped data by ranking methods. *Journal of economic entomology*, 39(6):269, 1946.

[28] Harel Yedidsion, Roie Zivan, and Alessandro Farinelli. Explorative max-sum for teams of mobile sensing agents. *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, pages 549–556, 2014.

[29] W. Zhang, G. Wang, Z. Xing, and L. Wittenburg. Distributed stochastic search and distributed breakout: properties, comparison and applications to constraint optimization problems in sensor networks. *Artificial Intelligence*, 161(1-2):55–87, January 2005.

[30] R. Zivan. Anytime local search for distributed constraint optimization. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 3*, pages 1449–1452. International Foundation for Autonomous Agents and Multiagent Systems, 2008.