Binary max-sum for multi-team task allocation in RoboCup Rescue

Marc Pujol-Gonzalez[1], Jesus Cerquides[1], Alessandro Farinelli[2], Pedro Meseguer[1], and Juan A. Rodriguez-Aguilar[1]

[1] IIIA-CSIC
Campus de la UAB, E-08193 Bellaterra, Spain
{mpujol,cerquide,pedro,jar}@iiia.csic.es
[2] Department of Computer Science, University of Verona
Strada Le Grazie, 15 37134 Verona, Italy
alessandro.farinelli@univr.it

Abstract. Coordination of agents involved in rescue missions is an important open research problem. We focus on the RoboCup Rescue Simulation (RCS) challenge, where different teams of agents perform urban rescue operations. Previous approaches typically cast such coordination problem as separate single-team allocation problems, and solve them separately. Our first key contribution is to focus on the max-sum approach, which has been successfully applied in this setting. We show that it is possible to reduce the computational complexity associated to max-sum from exponential to polynomial time. Our empirical evaluation shows that, by using our approach, the fire brigades team obtains significantly better results when compared to state-of-the-art approaches. Our second key contribution is a methodology that allows teams in RCS to make joint allocations. Specifically, our approach supports a modular design, where teams are independently modeled and subsequently connected via well-defined coordination points. To the best of our knowledge, this is the first task-assignment approach in the literature that enables teams in RCS to make simultaneous joint allocations. Experiments with fire brigades and police agents show that teams employing inter-team coordination are significantly more effective than uncoordinated teams.

1 Introduction

In many practical applications, such as rescue, surveillance and environmental monitoring, agents with different capabilities must form teams that cooperate in dynamic and unpredictable environments. For example, in rescue missions, agents with different capabilities must join forces to mitigate damages and help civilians [13]. Hence, how to allocate tasks to agents so that their performance is maximized is a crucial issue in these scenarios.

A standard model for the task allocation problem in the context of rescue agent teams is the Extended Generalized Assignment Problem (EGAP) [12]. This model considers situations where each agent can perform several tasks, subject to its own resource constraints (*e.g.*, a fire brigade can be in charge of several fires if it has enough water to work on all of them). It also includes execution constraints for groups of tasks (*e.g.*, k agents must work simultaneously on the same fire to extinguish it). Moreover, the model can capture the dynamic nature of these domains, where tasks can enter and exit the system during the mission execution (*e.g.*, fires can spread or be extinguished).

In this paper we also focus on dynamic task allocation, but we do not cast our problem as an EGAP. The key element that differentiates our work is that the (E)GAP model cannot fully capture the synergies among rescue agents working on the same task. It can express that agents should somehow synchronize their task execution for the solution to be valid. However, while it can represent that exactly (or no more than) k agents should be allocated to a task, it cannot represent that having $k + 1$ agents is still a valid solution, even though less desirable.

Capturing such synergies is essential for effective cooperation in rescue missions. One way to do so is to address the problem as a coalition formation problem, where agents must form coalitions to work on tasks [11]. Nonetheless, this approach focuses on domains where task deadlines constrain the number of agents that can be useful for a task (i.e., which can reach the task before its deadline) and would not scale in scenarios (such as the fire brigades task allocation problem) where the deadline for a task is not easy to foresee and all agents can potentially be useful on every task.

Consequently, here we take a different approach, and model our task allocation problem as social welfare maximization problem where the objective function is the sum of utility for each task, and the utility for a task is a function of the agents assigned to the task. Moreover, while all previous approaches focus on task allocation for agents belonging to the same rescue team (i.e., fire fighters), here we consider task allocation across multiple teams.

We propose the use of the max-sum approach as a solution technique. Max-sum has been recently applied to a significant variety of application domains with successful results, such as UAVs task assignment [2], Radar coordination [4], and the RoboCup Rescue [10,6]. However, in its basic form, max-sum exhibits an exponential complexity in the number of agents that must directly coordinate their actions.

Recently, it has been shown [14] that the computation associated to max-sum can be reduced to polynomial time (between $O(n)$ and $O(n^2)$ time) for some specific types of functions (called *Tractable Higher Order Potentials*, or THOPS in short). A distinctive feature of this approach is that it only uses binary variables, and hence it is typically named Binary Max-Sum (BMS). Such efficient version has been used to model a number of application domains with very good results [3,8,9].

Against this background, we propose a novel formalization of the rescue coordination problem based on BMS and only using THOPs, hence reducing the computational complexity of the solution process from exponential to polynomial time. Our approach independently represents each team and then makes them interface through well-defined coordination points, enabling agents of different teams to coordinate by sharing only relevant information. We evaluate our approach using the RoboCup Rescue Simulator through the RMASBench [6] benchmarking platform. In more detail, this paper makes the following contributions:

1. We first build a THOP version of the model designed in [6] for the coordination of the fire brigades team, showing how different coordination situations can be modeled as THOPs.
2. We empirically show that BMS with THOPs is much more efficient than standard max-sum, and therefore can operate quicker and provide better results. This is shown by using only the fire brigades team and comparing with the results with the approach proposed in [6] on standard scenarios (i.e., Paris and Kobe) used for the latest RoboCup Rescue competition. Specifically, in the best case our approach is able to save 50% more of the city than the state-of-the-art max-sum version.
3. We show how, by using the concept of coordination points, inter-team coordination can be achieved without altering the single team models. This allows a designer to tackle large problems without dealing with an explosion of design complexity.
4. We empirically validate the inter-team coordination between fire brigade and police agents. We show that by using our approach for inter-team coordination, rescue forces achieve significant improvements in the overall performance across a wide range of scenarios. For example, in the Paris scenario our approach saves up to 5 times as many buildings than with teams operating separately.

The rest of the paper is organized as follows. Section 2 provides background on max-sum. Section 3 presents the problem we tackle. Section 4 describes how the BMS approach can be applied to task allocation within a rescue team, and reports our empirical findings. Section 5 details our approach for inter-team coordination along with its empirical evaluation. Finally, Section 6 concludes the paper.

2 Background

Max-sum is an approximate optimization algorithm for unconstrained optimization that has been applied in different coordination problems. We review max-sum, and we see how, in some specific scenarios, its computational complexity can be reduced from exponential to polynomial. We also present a simple technique to transform constrained optimization problems into unconstrained ones.

2.1 Max-sum

Let $X = \langle x_1, \ldots, x_n \rangle$ be a sequence of variables, with each variable x_i taking states in a finite set $\mathcal{D}_i$ known as its *domain*. The joint domain $\mathcal{D}_X$ is the cartesian product of the domain of each variable. We use $\mathbf{x}_i$ to refer to a possible state of x_i, that is $\mathbf{x}_i \in \mathcal{D}_i$ and $\mathbf{X}$ to refer to a possible state for each variable in X, that is $\mathbf{X} \in \mathcal{D}_X$. We say that $\mathbf{X}$ is an assignment. Given a sequence of variables $Y \subseteq X$, a factor f is a function $f : \mathcal{D}_Y \to [-\infty, \infty)$. We refer to the variables in the scope of f as X_f. The set of factors of the problem is denoted by F.

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

$$\text{maximize} \quad \sum_{f \in F} f(\mathbf{X}_f) \tag{1}$$

$$\text{subject to} \qquad \mathbf{x}_i \in \mathcal{D}_i \qquad \forall i \in \{1, \ldots, n\} \quad , \tag{2}$$

where $\mathbf{X}_f$ contains the states assigned by $\mathbf{X}$ to the variables in the scope of factor f.

Max-sum provides an approximate solution in two steps. First, messages are sent from variable to factor and from factor to variable. This step is repeated until the messages no longer change or until a specified number of iterations is reached. After that, max-sum determines the best state for each variable independently.

At the first stage, max-sum assesses the message from variable x to factor f as

$$\mu_{x \to f}(\mathbf{x}) = \sum_{g \in \mathcal{N}(x) \setminus \{f\}} \mu_{g \to x}(\mathbf{x}) \ , \tag{3}$$

where $\mathcal{N}(x)$ stands for the factors that have variable x in its scope, $\mathbf{x}$ stands for a state of x, and $\mu_{g \to x}$ stands for the last message received by variable x from factor g. Max-sum assesses the message from f to x as follows:

$$\mu_{f \to x}(\mathbf{x}) = \max_{\mathbf{Y}} \left(f(\mathbf{x}, \mathbf{Y}) + \sum_{y \in Y} \mu_{y \to f}(\mathbf{y}) \right) \quad , \tag{4}$$

where Y is the set of variables in the scope of factor f excluding x, and $\mathbf{Y}$ is the joint state for all the variables in Y.

At the second stage, max-sum assesses the preferred states for each variable as $\arg\max_{\mathbf{x}_i} \sum_{f \in \mathcal{N}(x_i)} \mu_{f \to x_i}(\mathbf{x}_i)$.

2.2 Binary max-sum

The complexity of max-sum is driven by equation 4, that takes time exponential in the number of variables in the scope of the factor. Recently, several works [8,9,14] have shown that for some specific types of factors, called Tractable Higher Order Potentials (THOPS) in [14], this time can be reduced to polynomial provided that all the variables in the scope are binary. In the following we assume that variables are binary and say that in an assignment a variable is active (resp. inactive) if it takes value 1 (resp. 0).

Cardinality potentials (CPs) 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

$$f(\mathbf{X}_f) = g(n_f(\mathbf{X}_f)) \, , \tag{5}$$

where $n_f(\mathbf{X}_f) = \sum_{x_i \in X_f} \mathbf{x}_i$ stands for the number of active variables in assignment $\mathbf{X}_f$. Tarlow et al. [14] shows that for CPs, messages from equation 4 can be assessed in time $O(N \log N)$, where N is the number of variables in the scope of the factor.

Particular cases of CPs are strict cardinality constraints. Tarlow et al. [14] note that for some of these strict cardinality constraints, even more efficient algorithms to assess the messages are available. *OneAndOnlyOne* factors[3] enforce that one and only one of the variables in the factor should be active. Pujol-Gonzalez et al. [9] show that for this THOP the messages can be assessed in time $O(N)$. Tarlow et al. [14] 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*.

2.3 Max-sum for constrained optimization

Let us consider the optimisation problem expressed in section 2.1 by equations 1 and 2. Say now that, for instance, we add a side constraint to that problem to impose that one and only one of the variables in X has to be active ($\sum_{x \in X} \mathbf{x} = 1$). To apply max-sum, we need to transform the new constrained optimization problem into an unconstrained optimization one. We can do that by making the side constraint part of the objective function. If so, the objective function will decompose again additively as a sum of factors so the conditions for max-sum will apply.

Formally, we can represent the side constraint above by means of a factor *OneAndOnlyOne* that takes value 0 when the condition is satisfied and $-\infty$ otherwise, namely:

$$OneAndOnlyOne(\mathbf{X}) = \begin{cases} 0 & \text{if } \sum_{x \in X} \mathbf{x} = 1 \\ -\infty & \text{otherwise} \end{cases} .$$

Now we can reformulate the problem as the following unconstrained optimization one:

$$\begin{array}{ll} \text{maximize} & \sum_{f \in F} f(\mathbf{X}_f) + OneAndOnlyOne(\mathbf{X}) \\ \text{subject to} & \mathbf{x}_i \in \mathcal{D}_i \qquad\qquad \forall i \in \{1, \ldots, n\} \, . \end{array}$$

3 Problem description

The RoboCup Rescue Simulation Platform (RSP) is a benchmarking environment that simulates an urban search and rescue scenario where rescue forces (police patrols, ambulances and fire brigades)

[3] OneAndOnlyOne factors are named *selection functions* in [9].

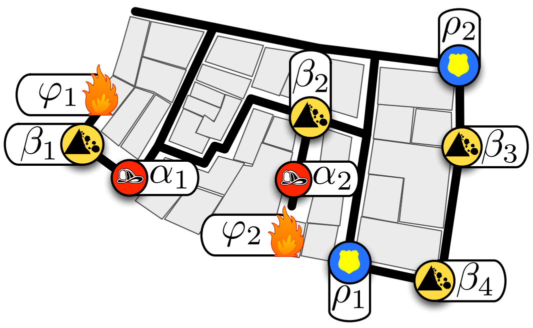


Fig. 1. Example scenario.

must coordinate their actions. Specifically, police patrols can unblock roads, fire brigades can extinguish fires, and ambulance agents can rescue trapped civilians. RSP 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 and blockades in a large urban area) [5].

In this paper we focus on the coordination problem of fire brigades, police patrols, and their interactions. On the one hand, regarding the coordination of 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, past 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. On the other hand, considering the actions of police patrols, a key element is that blockades on roads do not evolve over time. Moreover, unlike in the fire fighting situation, 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.

When we consider the whole picture, police patrols and fire brigades should 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: i) two police patrols $P = \{\rho_1, \rho_2\}$; ii) two fire brigades $A = \{\alpha_1, \alpha_2\}$; iii) four blockades $B = \{\beta_1, \beta_2, \beta_3, \beta_4\}$ that prevent agents from transiting the road they are blocking; and iv) 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 $(\rho_1 \rightarrow \beta_4), (\rho_2 \rightarrow \beta_3)$ 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 $(\rho_1 \rightarrow \beta_2), (\rho_2 \rightarrow \beta_1)$ so that fire brigade agents α_1 and α_2 can work on fires ϕ_1 and ϕ_2 respectively.

4 Coordinating a rescue team

In this section we cast the coordination problem faced by a team of rescue agents as an optimization problem. Then we show that this optimization problem can be efficiently solved by means of binary max-sum. First, section 4.1 formalizes the fire brigade coordination problem as an optimization problem. Next, section 4.2 shows that it is possible to encode such optimization problem into a binary form, by means of THOPs, so that binary max-sum can efficiently solve it. Finally, in Section 4.3 we empirically show that binary max-sum (with THOPS) is much more efficient that standard max-sum, and therefore can operate faster and provide better results.

4.1 Coordinating a team of fire brigades

In this section we formalize the fire brigade coordination problem as a problem of maximizing the joint utility of the whole team of fire brigades. The formalization is inspired by the informal description introduced in [6]. Solving the coordination problem amounts to deciding to which fire assign each fire brigade. Thus, 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. Thus, $y_{\alpha_1} = \varphi_2$ means that brigade α_1 is assigned to fire φ_2. Our objective is finding an assignment $\mathbf{Y}$ that maximizes the team utility $u(\mathbf{Y})$.

We model the utility of the fire brigade as a sum with a utility term per fire. Thus, $u(\mathbf{Y}) = \sum_{f \in F} u_f(\mathbf{Y})$, where $u_f(\mathbf{Y})$ represents the utility obtained from extinguishing fire f by means of assignment $\mathbf{Y}$. Furthermore, $u_f(\mathbf{Y})$ can be split in two terms, namely $u_f(\mathbf{Y}) = e_f(\mathbf{Y}) - r_f(\mathbf{Y})$. The first term, $e_f(\mathbf{Y})$, represents the utility that stems from extinguishing fire f. The second term, $r_f(\mathbf{Y})$, evaluates the amount of resources (the cost) expended on extinguishing fire f. Obviously, we are interested in extinguishing each fire in the best possible way whilst minimizing the cost.

To assess $e_f(\mathbf{Y})$ notice that not all fires are equally relevant. We assign a value v_f to each fire f, corresponding to the utility obtained by assigning a brigade to put it out. Moreover, observe that more than one brigade can be assigned to the very 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, there is a limit in the number of fire brigades that can successfully cooperate in extinguishing a particular fire. This is because, beyond some point, involving more fire brigades has a slight effect on the outcome of the extinguishing activity. Such limit depends on a number of factors, including the fierceness of a fire as well as the area of the building burning in flames. Here we refer to this threshold as t_f. Thus, we will 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, we consider that this penalty increases with the number of additional agents beyond the threshold. By combining the previous arguments we propose to assess $e_f(\mathbf{Y})$ as

$$e_f(\mathbf{Y}) = v_f \cdot n_f(\mathbf{Y}) - \kappa \cdot [\max(0, n_f(Y) - t_f)]^\gamma \tag{6}$$

where $\kappa \geq 0$ and $\gamma \geq 1$ are arbitrary coefficients that control the penalty increase.

As to $r_f(\mathbf{Y})$, the cost expended in extinguishing fire f, stems simply from the addition of the costs of those brigades that are assigned to it:

$$r_f(\mathbf{Y}) = \sum_{a \in A} r_{af}(\mathbf{y}_a) \tag{7}$$

Since in our problem all brigades are considered equivalent except in their location, we will evaluate the cost of assigning a brigade to a fire as proportional to the square of the distance between them[4]

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

$$r_{af}(\mathbf{y}_a) = \begin{cases} \nu d_{af}^2 & \text{if } y_a = f \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

where d_{af} is the normalized distance between brigade a and fire f, and $\nu \geq 0$ is an arbitrary coefficient. Thus, the closer the brigade, the smaller the cost of assigning it.

We can readily apply max-sum by creating a factor encoding for each fire f (encoding u_f) and exchanging messages between those factors and the variables in Y. However, since each u_f takes as inputs all of the variables of the problem, the assessment of the messages from factors to variables takes exponential time. In order to solve this problem, Kleiner et al. [6] proposed to heuristically reduce the set of fires to which a brigade can be assigned. In that way, each factor u_f will only have in its scope the variables of those agents that can help extinguish it. This reduces the complexity of the algorithm but: (i) it does also reduce the set of assignments under consideration; and (ii) it requires heuristic knowledge that can make it difficult to apply a similar solution in other coordination scenarios. In the following section we show that by encoding the problem into binary form and making use of THOPs we can accrue a further reduction in the time complexity without reducing the assignments under consideration and without requiring any heuristic knowledge.

4.2 Binarization

In order to binarize our problem, we simply split each decision variable y_a into $|F|$ binary variables z_{af}. Variable z_{af} is active (set to 1) in a solution whenever we assign brigade a to fire f and 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. Since this is not possible in our simulation, we need to add a constraint for each brigade, guaranteeing that it can only be assigned to a single fire. Therefore, the following set of consistency constraints have to be satisfied:

$$\sum_{f \in F} \mathbf{z}_{af} = 1 \qquad \forall a \in A. \tag{9}$$

Now, we need to encode the utility function $u(\mathbf{Z})$. As in the previous section, this function breaks as a sum with a utility term per fire, and thus $u(\mathbf{Z}) = \sum_{f \in F} u_f(\mathbf{Z})$. It is relevant to note that $u_f(\mathbf{Z})$, the utility obtained from extinguishing fire f by assignment $\mathbf{Z}$, only depends on which brigades are assigned to fire f. Thus, it does not depend on all of the variables in Z, but only on the set of variables that relate to fire f, namely $Z_{.f} = \{z_{af} | a \in A\}$. Hence, the scope of each factor u_f can be reduced to $Z_{.f}$, and we can talk of $u_f(\mathbf{Z_{.f}})$. Along the lines of the non-binary formalization, we have now that $u_f(\mathbf{Z_{.f}}) = e_f(\mathbf{Z_{.f}}) - r_f(\mathbf{Z_{.f}})$, where

$$e_f(\mathbf{Z_{.f}}) = v_f \cdot n_f(\mathbf{Z_{.f}}) - \kappa \cdot [\max(0, n_f(\mathbf{Z}_{.f}) - t_f)]^\gamma, \tag{10}$$

$$r_f(\mathbf{Z_{.f}}) = \sum_{a \in A} r_{af}(\mathbf{z}_{af}), \text{ and } r_{af}(\mathbf{z}_{af}) = \nu d_{af}^2 \mathbf{z}_{af}.$$

Now we are ready to cast the coordination problem faced by the fire brigades as the following constrained optimization problem:

$$\begin{array}{lll} \text{maximize} & \sum_{f \in F} e_f(\mathbf{Z}_{.f}) - \sum_{f \in F} \sum_{a \in A} r_{af}(\mathbf{z}_{af}) & \\ \text{subject to} & \sum_{f \in F} \mathbf{z}_{af} = 1 & \forall a \in A \\ & \mathbf{z}_{af} \in \{0,1\} & \forall a \in A \ \forall f \in F \end{array}$$

In order to apply max-sum, we need to transform the problem above into an unconstrained binary problem. However, note that each of the consistency constraints in equation 9 can be readily represented by a OneAndOnlyOne factor, as described in section 2.2 and thus, its corresponding max-sum messages can be efficiently assessed.

For each fire f, factor e_f does only depend on the number of variables active in its domain, but it does not depend on which of those variables are active. This can be readily noticed by analyzing the expression of e_f in equation 10. We observe that the dependence from $\mathbf{Z}_{\cdot\mathbf{f}}$ is always through $n_f(\mathbf{Z}_{\cdot\mathbf{f}})$, and hence the condition to be a CP shown in equation 5 is fulfilled. As a consequence, the messages out of factor e_f can be efficiently assessed following the procedure described in [14].

To summarize, it is possible to assess an approximate solution to our problem, without heuristically discarding assignments, by applying binary max-sum. Furthermore, each iteration of the algorithm can be executed in time $O(|F||A|\log|A|)$.

4.3 Empirical evaluation

In this section we empirically evaluate the performance of our task allocation mechanism for a team of fire brigades. With this aim, we compare our Binary Max-Sum (BMS) method with the state-of-the-art algorithms in [6], namely the Distributed Stochastic Algorithm (DSA) [15] and Max-Sum (MS) on standard scenarios used in the 2013 RoboCup competitions, namely Paris and Kobe. However, we discard the scenarios' elements that are irrelevant for our empirical evaluation, i.e., everything except for ignition points and fire brigades.

DSA: is essentially a greedy local search algorithm, executed in parallel by the participant agents. However, DSA introduces a stochastic parameter, DSA_p, to control the amount of parallelism allowed when running the algorithm. Basically, agents throw a dice before trying to perform any local improvement. If the dice is lower than DSA_p, then they can try to improve. Otherwise, they must not switch targets in the current iteration. In our case, there is an agent for each fire brigade. Initially, all brigades randomly choose some fire as their target and communicate their choices to all other brigades. Thereafter, the algorithm runs a number of greedy improvement iterations. During these iterations, each brigade receives the choices of the others. Then, provided it passed the dice throw, the brigade re-evaluates all its options considering the choices that others have made. If there is some target that yields a higher utility, it changes its target and communicates this new choice to other brigades. The greedy improvement iterations terminate when either a maximum number of them has been run, or no brigade is willing to change anymore. 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.

Max-sum: is the algorithm explained in Section 2.1. The difference between MS and BMS is that MS executes the non-binarized model, and therefore does not use any of the THOPs presented in the above sections. However, the implementation provided by RMASBench employs a particular function encoding (based on [10]), which exploits the factors' structure to reduce the computational effort from $O(d^n)$ to $O(2^n)$. To clarify, this implementation is computationally more efficient than standard max-sum. Nonetheless, even with this improvement, agents are not able to compute a solution within the time interval of our simulation step (i.e., 3 seconds), because the computational requirements are still exponential. To overcome this limitation we relax the problem by removing some of the dependencies between functions and variables. Following [6], we only consider k fire brigades per fire as options during the allocation process. Hence, the computational effort for the MS algorithm is not exponential in the number of agents present in the system, but only in k (i.e., $O(2^k)$).

For each algorithm, we keep track of the best global solution (assignment) it has found during all the iterations, and use that as the final result. This is possible because in RMASBench all agents

Metric	DSA	Max-sum	Binary max-sum
Score	6.92% [±1.00%]	5.45% [±0.61%]	**3.64% [±0.37%]**
Utility wins[7]	20.51% [±1.46%]	8.88% [±0.78%]	**73.91% [±1.65%]**
NCCCs	**3.88k [±0.04k]**	145,216k [±1,256k]	125.77k [±1.07k]
Num. msgs	69.02k [±0.83k]	**11.67k [±1.01k]**	395.14k [±4.26k]
Total bytes	552.16Kb [±6.70Kb]	**319.72Kb [±2.37Kb]**	3,161.16Kb [±34.04Kb]

Table 1. Statistics for DSA, MS and BMS averaged over 30 runs in the Paris scenario (agents start acting after 35 iterations). The best result for each metric is in bold. Numbers in brackets report the standard error of the mean.

can communicate with each other. Along the same line, we introduce a final sequential optimization procedure where each fire brigade is given the opportunity to make a single target switch based on the allocations of other brigades[5].

As for the parameters of the utility function for both DSA and MS, after an extensive empirical evaluation, we set κ=2, γ=2, and ν=10. 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. Regarding algorithms' parameters we set the number of maximum iterations[6] for all the algorithms to 100. We set $DSA_p = 0.6$ because it yielded the best results in our tests. Additionally, we set $k = 4$ for the MS algorithm because this is the largest value allowing it to provide results within the simulation step.

Finally, we use the following metrics to evaluate the performance of the algorithms: i) *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; ii) *Utility wins*: counts the percentage of iterations where each algorithm obtains the best configuration in terms of utility; iii) *NCCCs [7]*: captures the per-iteration average amount of non-parallelizable computation performed by the agents; iv) *Num msgs*: tracks the average number of messages sent between all agents in a single iteration; and v) *Total bytes*: is the average number of bytes per iteration sent between all agents.

Results. Table 1 shows results obtained on the Paris map with a start time of 35 simulation steps[8] averaged over 30 simulations. BMS achieves the best results in both quality measures, i.e. *Score* and *Utility wins*. Particularly, BMS's result indicates that 3.6% of the city has been damaged. In contrast, 5.4% of the city is destroyed when using MS (50% more than BMS), and 6.9% of the buildings are destroyed when using DSA (91.7% more than BMS). Additionally, BMS fins the allocation with highest utility (out of the three) in 73.9% of the iterations. Unsurprisingly, these gains in quality come at a cost. DSA, being the simplest algorithm, obtains the worst results in quality, but it requires few computational resources and relatively low communications. MS's increase in quality requires a large increase in computational power. In contrast, BMS computes more than DSA, yet requires substantially more bandwidth than the other algorithms. Nonetheless, taking into account that an iteration of the RoboCup Simulator represents one minute of real time, all these costs are within an

[5] Notice that any given city has a relatively small number of fire brigades, so the number of steps for this procedure is not a bottleneck for the approach.

[6] Previous work shows that typically such algorithms reach good solutions within the first 50 iterations for problems of similar scale[1].

[7] Adds to $>$100% because ties count as wins for both algorithms.

[8] This means we allow fire brigades to execute any action after 35 time steps. This parameter allows us to tune the difficulty of a scenario leaving the main elements (i.e., map, fire brigades, ignition points) fixed

acceptable range. For instance, the $3,161$Kb sent by BMS amounts to a bandwidth requirement of 10.5Kbps.

We also experimented with a higher starting time for the agents. The results (omitted for space reasons) are very similar to those in Table 1, maintaining the same proportions and best/worst algorithms for each metric. Finally, we experimented with 2013's Kobe scenario. This scenario is much easier for the fire brigades, because there is only one fire focus and the map is small and easy to navigate. BMS is still the best method quality-wise, but the results are much tighter, with MS and DSA being only 3% worse.

To summarize, BMS provides the highest quality solutions. Given its affordable resource requirements, and the fact that higher quality means saving buildings in this scenario, BMS stands as the method of choice for single-team task allocation in the settings where we experimented.

5 Inter-team coordination

At this point we have shown that binary max-sum is valuable to solve a single-team coordination problem. However, it is rather unrealistic to think that a team can make its own decisions. In general, teams depend on each other to accomplish their tasks. Consider again our example in Figure 1. We know that fire brigades try to avoid blocked fires when coordinating. However, this completely disregards the fact that police agents will be removing blockades in the meantime so that the fire brigades may be able to reach blocked fires in the near future. Therefore, to capture the interdependencies between teams' decisions, we argue that it is necessary to enable teams to perform inter-team coordination.

In this section we present a methodology to enact inter-team coordination that enables the teams involved to make their decisions considering a shared goal. With this aim, 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. To overcome this challenge, our methodology proposes a modular construction of the global utility function following the next steps:

1. *Define single-team coordination models* for each team involved in the inter-team coordination. Section 4 has already illustrated this step by showing how fire brigades coordinate between themselves.
2. *Identify a common language.* At this step the designer must identify the coordination objects capturing 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 embed the common language.* The next step for the designer is to embed coordination variables into each single-team coordination model defined by step 1.

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 max-sum 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.

In the rest of this section we exemplify the application of our methodology to the coordination of a team of fire brigades and a team of policemen.

5.1 Define single-team 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 4.2 we already introduced a coordination model for a team of fire brigades. We start by extending this model to take into account blocked roads, and then we focus on defining a coordination model for a team of policemen.

Including blockades in the fire brigades model. Since a blockade may prevent fire brigade a from attending fire f, we should apply a penalizing factor to fire brigades allocated to blocked fires. That is, when fire brigade a is *being prevented* from reaching fire f by blockade b we have to introduce a penalty $-M$ whenever a is assigned to f. Thus, whenever a blockade is in the way from a to f we add to the cost r_{af} an additional factor

$$r'_{af}(\mathbf{z}_{af}) = \begin{cases} 0 & \text{if } \mathbf{z}_{af} \text{ is inactive,} \\ -M & \text{otherwise.} \end{cases}$$

Defining 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 for 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.

Additionally, a patrol can not remove more than one blockade at a time. Therefore, 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. Thus, the following constraint needs to be enforced:

$$\sum_{b \in B} \mathbf{x}_{pb} \leq 1 \qquad \forall p \in P \,. \tag{11}$$

Similar to the case of fire brigades, the utility of an allocation ($u(X)$ in this case) takes a factor for each blockade, $u(\mathbf{X}) = \sum_{b \in B} u_b(\mathbf{X}_{.b})$. Furthermore, the blockade factors can be broken as $u_b(\mathbf{X}_{.b}) = e_b(\mathbf{X}_{.b}) - r_b(\mathbf{X}_{.b})$. An important limitation in RoboCup Rescue is that policemen efforts do not stack up. That is, two patrols working together will not be any better at removing a blockade than a single one. Since all patrols are assumed to be equivalent in their ability to remove blockades and blockades do not have distinguishing characteristics, we assign a utility $v_B > 0$ to attending any blockade b. This utility is obtained whenever one or more patrols are assigned to remove that blockade. Thus

$$e_b(\mathbf{X}_{.b}) = \begin{cases} v_B & \text{if } n_b(\mathbf{X}_{.b}) \geq 1 \\ 0 & \text{otherwise} \end{cases} \quad ,$$

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

Similarly to what we did for the fire brigades team with blockades, when measuring the cost of assigning a patrol p to servicing blockade b, we consider two different options. On one hand, if the blockade is directly accessible to the patrol (that is, when no other blockade appears in the path between p and b) then the cost is proportional to the square of the distance between them, namely d_{pb}. On the other hand, if the blockade is not directly accessible we sum an additional penalty Q

to the squared distance. Thus, the evaluation of the amount of resources spent on some blockade b becomes $r_b(\mathbf{X}_{\cdot b}) = \sum_{p \in P} r_{pb}(\mathbf{x}_{pb})$, where

$$r_{pb}(\mathbf{x}_{pb}) = \begin{cases} 0 & \text{if } \mathbf{x}_{pb} \text{ is inactive,} \\ \eta d_{pb}^2 & \text{if } \mathbf{x}_{pb} \text{ is active and blockade } b \text{ is accessible from } p, \\ \eta d_{pb}^2 + Q & \text{otherwise} \end{cases}$$

and η is a proportionality constraint.

To be able to apply max-sum to the above problem, we need to incorporate the constraints as part of the objective function, such as described in section 2.2. For each patrol p we have a constraint enforcing that the patrol can be assigned to at most one of the blockades, that is $\sum_{b \in B} \mathbf{x}_{pb} \leq 1$. Each of those constraints can be transformed into a factor defined as

$$AtMostOne_p(\mathbf{X}_{\cdot p}) = \begin{cases} 0 & \text{if } \sum_{b \in B} \mathbf{x}_{pb} \leq 1 \\ -\infty & \text{otherwise.} \end{cases} \tag{12}$$

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 [9]. The assessment of the messages for an AtMostOne factor in a max-sum iteration can then be done efficiently in time $O(|B|)$.

5.2 Identifying a common language

The second step in our methodology consists in identifying the coordination objects capturing the interdependencies between teams. At this point it should be clear that the coordination objects in our RoboCup example 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. Therefore, 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 if that is not the case. In our particular example, these variables are already enough to 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 common language is intended to enable the fire brigades team and the policemen team to exchange information about their inter-dependencies regarding blockades.

After establishing that common language, in the next section we show how to modify the existing fire brigades team binary model to include blockades, and to take into account these coordination variables. Later on we provide a model for police team coordination that also incorporates blockades and coordinates by means of those variables.

5.3 Extending single-team coordination

The third step in our methodology consists in extending individual team models to embed coordination variables. Hereafter we extend both the fire brigades team model and the police patrols team model introduced in section 5.1 to take coordination variables into account.

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 they know that the 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.

$$s_{afb}(\mathbf{z}_{af}, \mathbf{c}_b) = \begin{cases} M & \text{if } \mathbf{c}_b \text{ is active and } \mathbf{z}_{af} \text{ is active} \\ 0 & \text{otherwise.} \end{cases} \tag{13}$$

For instance, in the example in Figure 1, blockade β_1 is preventing brigade α_1 from reaching fire φ_1. Therefore, a new factor $s_{\alpha_1\varphi_1\beta_1}$ is required to introduce a penalty if blockade β_1 is not being attended (and thus c_{β_1} is inactive). Notice that the interface variables c_{β_2} and c_{β_3} are not connected to any factor. This represents the fact that, in our example, fire brigades do not care about these blockades at all.

Extending the police team model. In the case of the police team, their internal variables should be consistent with the semantics of the coordination variables c_b above. Specifically, c_b should be active when some police agent is attending b, or inactive otherwise. That is, variable c_b is an indicator of whether any of the variables in $X_{.b}$ are active. We can enforce this by adding a new *Indicator* factor $I_b(\mathbf{c}_b, \mathbf{X}_{.b})$ for each blockade, defined as

$$I_b(\mathbf{c}_b, \mathbf{X}_{.b}) = \begin{cases} 0 & \text{if all of the variables in } \mathbf{X}_{.b} \text{ and } \mathbf{c}_b \text{ are inactive, or} \\ & \quad \text{at least one variable in } \mathbf{X}_{.b} \text{ is active and } \mathbf{c}_b \text{ is active} \\ -\infty & \text{otherwise} \end{cases}$$

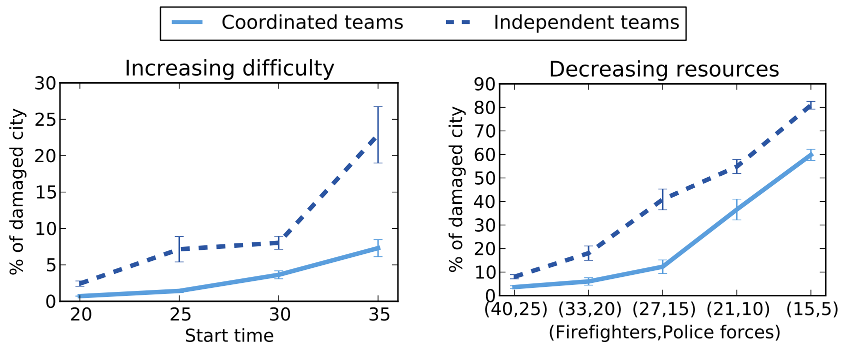
This factor is not known to be a THOP, but we can derive expressions for its messages by noticing that it is a composite factor [14] where c_b defines two partitions, the first one when c_b is active and the second one when c_b is inactive. In the first partition, the factor is an *AtLeastOne* between the variables in $X_{.b}$, which we have just shown that it is a THOP. In the second partition, the factor is an *AllInactive*[9] factor between the variables in $X_{.b}$. Since the *Indicator* potential is a composite factor and we have a THOP in each of the partitions defined by c_b, we can assess the messages of this factor in $O(|B|)$ time.

At this point, we are ready to construct the global utility function for our inter-team coordination problem. 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 binary max-sum to solve the inter-team coordination problem. The execution of binary max-sum 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 the police team's cost of removing a blockade. Binary max-sum convergence ends up with an agreement between teams. In the next section we empirically evaluate the benefits of our inter-team coordination approach.

5.4 Empirical evaluation

In this section we pursue to experimentally validate the methodology described above, by evaluating the performance of the inter-team coordination mechanism developed. With this aim, we performed

[9] A trivial derivation shows that the messages sent by an *AllInactive* are simply $-\infty$.



(a) Increasing difficulty (b) Decreasing resources

Fig. 2. Performance comparison between independent teams and coordinated teams in the Paris map. Results are averaged over 30 runs and error bars represent the standard error of the mean.

a number of experiments to check the differences in performance between coordinated and independent teams.

We run the experiments on the same scenarios used in Section 4.3. However, in this case we they also include police agents as defined in the official competition. 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.

We also employ the same parameters presented in Section 4.3. This includes both the algorithm's parameters and the parameters for the fire brigades utility model. Additionally, we set the η proportionality constant to 10^{-3}. This gives 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.

Results. Our first experiment evaluates performance of the coordinated teams (CTs) and independent teams (ITs) as the scenario becomes harder. We use the official Paris scenario, which has 40 brigades and 25 patrols, and test with starting times of 20, 25, 30, and 35 iterations (the longer the start time, the harder the scenario becomes because fires have more time to spread). Figure 2(a) shows that the percentage of the city damaged by fire is smaller for CTs on all the scenarios. Depending on the difficulty of the scenario, the burned area with ITs is from 2.2 times up to 5 times larger than with CTs. Also, CTs seem to be able to cope better with more difficult scenarios. Again, the increase in preserved area comes at a cost: the CTs approach requires up to twice (in the worst case) as much communication as the ITs, albeit it uses about the same amount of computation.

Along the same lines, our next experiment tested the performance of both approaches when the available number of rescue agents decreases (there are less rescuing resources). Figure 2(b) shows that there is a similar trend in this case. In fact, the CTs are able to outperform ITs even with a lower number of rescue agents (*e.g.*, 18.05% of the city damaged when 33 fire brigades and 20 patrols operate using ITs versus 12.28% when 27 fire brigades and 15 patrols operate in coordination).

Similar tests were also conducted in the Kobe scenario leading to very similar results, showing significant differences between using ITs and CTs.

6 Conclusions

We have shown that in some complex task allocation scenarios it is possible to reduce the computational complexity associated to MS from exponential to polynomial time. We have empirically evaluated this approach using the RoboCup Rescue simulator, where BMS achieves better results than other state-of-the-art methods.

Additionally, we have presented a methodology that allows multiple teams to make joint allocations by enabling them to coordinate during the task allocation process. Experiments with fire brigades and police agents show that teams employing inter-team coordination are significantly more effective than uncoordinated teams.

Acknowledgments. Work funded by projects RECEDIT (TIN2009-13591-C02- 02), AT (CSD2007-0022), COR (TIN2012-38876-C02-01), MECER (201250E053), the Generalitat of Catalunya grant 2009-SGR-1434, and the Ministry of Economy and Competitivity grant BES-2010-030466.

References

1. 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.
2. D. Fave et al. 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. B. J. Frey and D. Dueck. Clustering by passing messages between data points. *Science*, 315(5814):972–976, 2007.
4. Y. Kim, M. Krainin, and V. Lesser. Application of max-sum algorithm to radar coordination and scheduling. In *Workshop on Distributed Constraint Reasoning*, 2010.
5. 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 *IEEE International Conference on Systems, Man, and Cybernetics, 1999.*, volume 6, pages 739–743. IEEE, 1999.
6. 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.
7. A. Meisels, E. Kaplansky, I. Razgon, and R. Zivan. Comparing performance of distributed constraints processing algorithms. In *Workshop on distributed constraint reasoning DCR 2002*, pages 86–93, 2002.
8. 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.
9. 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.
10. 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.
11. 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.
12. 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.
13. 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.
14. D. Tarlow, I. E. Givoni, and R. S. Zemel. HOP-MAP: Efficient Message Passing with High Order Potentials. In *International Conference on Artificial Intelligence and Statistics*, volume 9, pages 812–819, 2010.
15. 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):55–87, 2005.