Mission Oriented Robust Multi-Team Formation and Its Application to Robot Rescue Simulation

Tenda Okimoto*, Tony Ribeiro, Damien Bouchabou†, Katsumi Inoue††**
* Kobe University, Japan ** IRCCYN, École Centrale de Nantes, France,
† Paul Sabatier University, France, †† National Institute of Informatics, Japan
tenda@maritime.kobe-u.ac.jp, tony.ribeiro@irccyn.ec-nantes.fr, inoue@nii.ac.jp

Abstract

Team formation is the problem of selecting a group of agents, where each agent has a set of skills; the aim is to accomplish a given mission (a set of tasks), where each task is made precise by a skill necessary for managing it. In a dynamic environment that offers the possibility of losing agents during a mission, e.g., some agents break down, the robustness of a team is crucial. In this paper, the focus is laid on the mission oriented robust multi-team formation problem. A formal framework is defined and two algorithms are provided to tackle this problem, namely, a complete and an approximate algorithm. In the experiments, these two algorithms are evaluated in RMASBench (a rescue multi-agent benchmarking platform used in the RoboCup Rescue Simulation League). We empirically show that (i) the approximate algorithm is more realistic for RMASBench compared to the complete algorithm and (ii) considering the robust mission multi-teams have a better control on the fire spread than the sophisticate solvers provided in RMASBench.

1 Introduction

Team formation [Liemhetcharat and Veloso, 2012; Nair and Tambe, 2005; Vidal, 2004] is an important aspect of multi-agent systems. In many application problems, e.g., RoboCup Rescue [Kitano and Tadokoro, 2001; James *et al.*, 2015], unmanned aerial vehicles operations [George *et al.*, 2010], and team formation in social networks [Lappas *et al.*, 2009], the groups of agents must coordinate to solve them effectively. In a dynamic environment that offers the possibility of losing agents during a mission, e.g., an agent is injured in a rescue mission, the robustness of a team is crucial. How to form the robust teams that can continue to perform their missions (even if some agents break down) is the main purpose of this work.

In this paper, a novel framework called a mission-oriented robust multi-team formation problem is defined. In this problem, a set of agents and a set of missions are given, and the aim is to form robust teams which can achieve the given missions even if some agents break down, minimize the total cost and maximize the robustness of the teams.

Furthermore, two algorithms are provided to tackle this problem, namely a complete and an approximate algorithms, and also some heuristics are developed that exploit the specificities of the problem. The complete algorithm is based on a branch and bound technique [Lawler and Wood, 1966] and can guarantee to find all Pareto optimal teams, i.e., trade-off between the cost and robustness of the teams. However, since the number of Pareto optimal teams is often exponential, finding all Pareto optimal teams becomes easily intractable [Okimoto *et al.*, 2012]. Thus, a local search based approximate algorithm is also provided [Aarts and Lenstra, 2013].

In the experiments, the performances of two algorithms are evaluated in RMASBench [Kleiner *et al.*, 2013], a testbed for multi-agent coordination algorithms based on the existing RoboCup Rescue simulation platform (RSP) [Skinner and Ramchurn, 2010]. RMASBench provides a library of implementations of state-of-the-art distributed constraint optimization solvers such as DSA [Fitzpatrick and Meertens, 2003] and MaxSum [Farinelli *et al.*, 2008] and facilities to compare coordination algorithms. Our local search based approximate algorithm is then compared with those existing solvers.

As an application domain, we believe that forming rescue teams is promising. Consider the problem of forming rescue teams in a disaster area. There are a set of missions to be accomplished which changes dynamically and a set of rescue robots, where each robot has different skills to achieve the tasks of a mission, e.g., providing medical treatment, acting as a firefighter and driving a vehicle. Assume that their current positions are different. Forming robust and costly rescue teams by considering the distance to the disaster area, amounts to a robust multi-team formation problem.

Compared to [Okimoto *et al.*, 2015], the aim of our framework is to form multiple robust teams. Also, this paper proposes efficient algorithms and evaluates them on RoboCup Rescue simulations, while [Okimoto *et al.*, 2015] provides a standard algorithm and evaluates it with random problems. Our framework is similar to the multi-set multi-cover problem [Hua *et al.*, 2009] where the authors proposed an algorithm in which multiple sets can cover a multi-set. If we consider a mission as a set of tasks, finding a robust team to achieve the mission is equivalent to trying to cover the set of tasks multiple times. Compared to [Hua *et al.*, 2009], this paper considers the robustness and the cost of each team simultaneously (i.e. bi-objective optimization).

2 Mission-Oriented Robust Multi-Team Formation

In this section, a formal framework for mission-oriented robust multi-team formation is defined. Furthermore, the bi-objective optimization problem for this framework is pointed out, i.e., finding out every multi-team, which is optimally robust and/or cheap. The following definitions are similar to the definitions provided in [Okimoto *et al.*, 2015].

Definition 1 (Multi-Team Formation Problem) A *multi-team formation problem description* is defined by a tuple $MTF = \langle A, S, M, cost, skill \rangle$, where $A = \{a_1, ..., a_n\}$ is a set of agents, $S = \{s_1, ..., s_m\}$ is a set of skills (of agents), $M = \{m_1, ..., m_g\}$ is a set of missions where $m_i \subseteq 2^S$ for $1 \leq i \leq g$, $cost : 2^A \times M \rightarrow \mathbb{N}$ is a cost function, and *skill* is a mapping from A to 2^S. A set of agents $T \subseteq A$ is said to be a *team* and a pair (T, m) is called a *mission team* where $m \in M$.

In the following, two standard properties and the robustness of a mission team are defined.

Definition 2 (Mission Team Cost) Let $MTF = \langle A, S, M, cost, skill \rangle$ be a multi-team formation problem description. Given a mission team (T, m) and a non-negative integer c, (T, m) is called *c-costly*, if the cost of (T, m) is less than c :

$$cost\,(T, m) \leq c.$$

Definition 3 (Mission Team Validity) Let $MTF = \langle A, S, M, cost, skill \rangle$ be a multi-team formation problem description. Given a mission team (T, m), the mission team is said to be *valid* w.r.t. m, if (T, m) can accomplish m :

$$m \subseteq \bigcup_{a_i \in T} skill\,(a_i).$$

Definition 4 (Mission Team Robustness) Let $MTF = \langle A, S, M, cost, skill \rangle$ be a multi-team formation problem description. Given a mission team (T, m) and a non-negative integer k, (T, m) is said to be *k-robust* w.r.t. m, if for every set of agents $T' \subseteq T$, such that $|T'| \leq k$, $(T \setminus T', m)$ is valid w.r.t. m.

Now, a mission multi-team for multi-team formation and its properties are defined, i.e., cost, validity and robustness.

Definition 5 (Mission Multi-Team) Let $MTF = \langle A, S, M, cost, skill \rangle$ be a multi-team formation problem description. A set of mission teams is said to be a *mission multi-team* denoted MMT.

$$MMT = \{(T_i, m_i) \mid T_i \subseteq A, m_i \in M, \bigcap_{1 \leq i \leq g} T_i = \emptyset, 1 \leq i \leq g\}.$$

Definition 6 (Mission Multi-Team Cost) Let $MTF = \langle A, S, M, cost, skill \rangle$ be a multi-team formation problem description. Given a mission multi-team MMT and a non-negative integer c_M, MMT is said to be *c_M-costly* if the sum of the costs of all $(T_i, m_i) \in MMT$, $(1 \leq i \leq g)$, is less than c_M :

$$\sum_{1 \leq i \leq g} cost\,(T_i, m_i) \leq c_M.$$

Definition 7 (Mission Multi-Team Validity) Let $MTF = \langle A, S, M, cost, skill \rangle$ be a multi-team formation problem description. Given a mission multi-team MMT, MMT is said to be *valid* w.r.t. M, if each mission team (T_i, m_i) of MMT is valid w.r.t. m_i where $1 \leq i \leq g$.

Definition 8 (Mission Multi-Team Robustness) Let $MTF = \langle A, S, M, cost, skill \rangle$ be a multi-team formation problem description. Given a mission multi-team MMT and a non-negative integer k_M, MMT is said to be *k_M-robust* w.r.t. M, if for every mission team $(T_i, m_i) \in MMT$ where $1 \leq i \leq g$, (T_i, m_i) is at least k_M-robust w.r.t. m_i, i.e., for every k_i-robust mission team (T_i, m_i) of MMT,

$$k_M = min\{k_i \mid 1 \leq i \leq g\}.$$

When we consider the cost and the robustness of a mission multi-team together, we can view a multi-team formation as a bi-objective optimization problem. In this problem, generally, since trade-offs exists between two objectives, there does not exists an ideal mission multi-team which minimizes the cost c_M and maximizes the robustness k_M simultaneously. Therefore, the "optimal" mission multi-team of this problem is characterized by using the concept of *Pareto optimality*.

Definition 9 (Dominance) Let $MTF = \langle A, S, M, cost, skill \rangle$ be a multi-team formation problem description and MMT and MMT' be two mission multi-teams, where MMT is c_M-costly and k_M-robust and MMT' is c'_M-costly and k'_M-robust. *MMT dominates MMT'* if and only if $k_M \geq k'_M$ and $c_M < c'_M$, or $k_M > k'_M$ and $c_M \leq c'_M$.

Definition 10 (Pareto optimality) Let $MTF = \langle A, S, M, cost, skill \rangle$ be a multi-team formation problem description and MMT be a mission multi-team. MMT is said to be a *Pareto optimal mission multi-team* if there exists no mission multi-team MMT' that dominates MMT.

Definition 11 (Bi-objective optimization problem)

Input: A multi-team formation problem description $MTF = \langle A, S, M, cost, skill \rangle$.

Output: Find all Pareto optimal mission multi-teams (T, m).

3 Algorithms for Mission-Oriented Robust Multi-Team Formation

In this section, two algorithms for solving a mission oriented robust multi-team formation problem are proposed, namely complete and approximate algorithms. The former can guarantee to find all Pareto optimal mission multi-teams, while the latter can find good mission multi-teams very quickly.

3.1 Complete Algorithm

The complete algorithm is based on a branch and bound technique [Lawler and Wood, 1966] and can guarantee to find all Pareto optimal teams, i.e., trade-off between the cost and robustness of the teams. In this algorithm, we exploited the specificities of the multi-team formation problem MTF to design dedicated heuristics and consider each agent as a variable whose each domain value corresponds to a mission. Algorithm 1 shows the initialization of this algorithm. In the pseudo-code, S and As are initialized (i.e., we set them as empty), where S is a set of $MMTs$ and As represents an MMT (line 3-6). At the beginning, agents are not assigned to any mission, i.e., each mission multi-team is the empty set (line 8). In this algorithm, in order to create a search-tree, we assume that the ordering among agents is lexicographically

Algorithm 1 Multi-Teams Formation Branch & Bound

1: INPUT : A multi-team formation problem MTF = <$A, P, M, cost, skill$>.
2: OUTPUT : the set of all Pareto optimal mission multi-teams S
3: S: a set of MMTs // All Pareto optimal solutions
4: $S \leftarrow \emptyset$
5: As: a MMT // Current assignments
6: $As \leftarrow \emptyset$
7: **for** each mission m of M **do**
8: $\quad As \leftarrow As \cup \{(\emptyset, m)\}$ // All missions are added to As
9: **end for**
10: $solve(1, As, S, MTF)$
11: **Return** S

Algorithm 2 solve(N, As, S, MTF)

1: INPUT : An integer N, As an MMT, a set of mission multi-teams T and a multi-teams formation problem MTF = <$A, P, M, cost, skill$>.
2: // 1) All agents have been assigned
3: **if** $N > |A|$ **then**
4: $\quad$ **if** As is not valid **then**
5: $\quad\quad$ **Return**
6: $\quad$ **if** As is not dominated by any element of S **then** // Check As is dominated by the elements of S
7: $\quad\quad$ Remove all MMT from S which are dominated by As // Check As dominates the elements of S
8: $\quad\quad$ $S \leftarrow S \cup \{As\}$
9: $\quad\quad$ **Return**
10: a: the N^{th} agent of A
11: // 2) Assign agent N to a mission
12: **for** each (T, m) of As **do**
13: $\quad$ // 3) Check if a can accomplish a task of m
14: $\quad$ **if** $skill(a) \cap m == \emptyset$ **then**
15: $\quad\quad$ **Continue**
16: $\quad$ $T \leftarrow T \cup \{a\}$
17: $\quad$ // 4) Check the cost bound
18: $\quad$ **if** $cost(As) > maxCost$ **then**
19: $\quad\quad$ $T \leftarrow T \setminus \{a\}$
20: $\quad\quad$ **Continue**
21: $\quad$ // 5) Check the validity
22: $\quad$ $MinAgent :\leftarrow 0$
23: $\quad$ **for** each pair (T_i, m_i) of As **do**
24: $\quad\quad$ k : the required robustness for m_i
25: $\quad\quad$ **if** T_i is not valid w.r.t. m **then**
26: $\quad\quad\quad$ $MinAgent \leftarrow MinAgent + k + 1$
27: $\quad\quad$ **if** T_i is k'-robust w.r.t. m and $k' < k$ **then**
28: $\quad\quad\quad$ $MinAgent \leftarrow MinAgent + (k - k')$
29: $\quad$ **end for**
30: $\quad$ **if** $(|A| - N) < MinAgent$ **then**
31: $\quad\quad$ $T \leftarrow T \setminus \{a\}$
32: $\quad\quad$ **Continue**
33: $\quad$ // 6) Continue the search with the next agent
34: $\quad$ solve($N + 1$,As, S, MTF)
35: $\quad$ $T \leftarrow T \setminus \{a\}$
36: **end for**
37: **Return**

given. The solving starts with the first agent of the ordering among agents, i.e., root node in the search tree (line 10).

Algorithm 2 shows the details of the solving part. It takes an integer N as a parameter and tries to assign the N-th agent of A to each possible mission M (line 11,12). When an agent cannot perform any task of the mission m, this mission is ignored. Otherwise, the agent is added to the team T of $(T, m) \in As$ (line 13-16). If the team cost of As is larger than the cost bound by adding the agent in T, the next mission is considered (line 17-20). Then, it checks the validity of As (line 21-32). In this part, the basic idea is to detect the case when the current partial team assignment As cannot lead to a valid completion. To form a k-robust mission team for a mission, we need at least $k + 1$ agents. Thus, it can compute the minimal number of agents required to obtain a valid mission team for each remaining missions (line 22-29). When the number of remaining unassigned agents is lower than this minimal value, it is not possible to find a mission team with the current partial assignment. $(|A| - N)$ is the maximal number of agents that can still be added in a mission team. If this number is less than the minimal required $MinAgent$, it is useless to continue with the current mission teams. It cancels the last choice of adding agent ($T \leftarrow T \setminus \{a\}$), and continues with the next agent (line 30-32). This allows us to branch before assigning all agents and reduces the search space. If the current assignment passed all checks, it continues to solve with the next agent (line 34). When the last agent of A has been assigned in As, it provides a complete assignment. If As is valid and not (Pareto) dominated, As is added to S (line 2-9). When the previous mission teams of S are dominated by the current mission team As, they are removed from S. Finally, it backtracks (line 9 and 35) and continues the search (line 12).

Theorem 1 For the proposed complete algorithm, it holds (i) the required memory belongs to $O(n(m + 1)^n)$ and (ii) the required computation time belongs to $O(n(m + 1)^{n+2})$, where m is the number of missions and n is the number of agents.

Let us proof (i). In the worst case, all possible mission multi-teams are Pareto optimal, i.e., there exists $(m + 1)^n$ Pareto optimal mission multi-teams: each agent can belong to every possible teams or no team. A mission multi-team is composed by an assignment and a pair of integers: the cost and the robustness of it. Since an agent can only belong to one mission team, an assignment can be represented by a vector of integers of size n were each component encodes the team of one agent. In order to compute the robustness, for each pair $(T, m') \in As$, we store a vector where each component corresponds to a skill s of m' and the value of the component is the number of agents a' of T such that $s \in skill(a')$. The memory size of the Pareto optimal mission multi-teams is then bound by $O((2 + 2n)(m + 1)^n) = O(n(m + 1)^n)$. Since the complete algorithm only requires to store them during the solving, the required memory is bound by $O(n(m + 1)^n)$.

Next, let us proof (ii). Let ST be the number of partial assignments of MTF and j be a non-negative integer ($1 \leq j \leq n$). Consider that only one agent can change its mission team at a given time. When no pruning occurs, and since there are $(m + 1)^j$ possible assignments for the first j agents in the ordering, the number of partial assignments is given by $\sum_{j=1}^{n}(m + 1)^j = \frac{(m+1)^{n+1} - 1}{m + 1 - 1} - 1 = \frac{(m+1)^{n+1} - 1}{m} - 1$ that is bound by $O((m + 1)^{n+1})$. In each time, a new mission team (T, m') is assigned to an agent a in the current partial assignment As, the complete algorithm:

Algorithm 3 Multi-Teams Formation Local Search

```
1: INPUT: A multi-teams formation problem MTF = <A, P, M, cost, skill> and MaxNeighbor a non-negative integer
2: OUTPUT : the best MMT found
3: try ← 0
4: Create a MMT As randomly until As is valid
5: // Phase 1: Random search
6: do
7:     As' ← As
8:     do// 1.1) Generate a neighbor of As
9:         Select an agent a randomly
10:        m1 ← the mission of a
11:        Remove a from T1 in (T1, m1) ∈ As'
12:        Select a mission m2 randomly
13:        Add a to T2 in (T2, m2) ∈ As'
14:    while As' is not valid
15:    if As' dominates As then // 1.2) Better solution
16:        As ← As'
17:        try ← 0
18:    else// 1.3) Try another neighbor
19:        try ← try + 1
20: while try < MaxNeighbor
21: // Phase 2: Systematic search
22: do
23:    As' ← As
24:    improvement ← False
25:    for each agent a of A do
26:        m1 ← the mission of a
27:        Remove a from T1 in (T1, m1) ∈ As'
28:        for each mission m in M do
29:            Add a to T in (T, m) ∈ As'
30:            if As' dominates As then
31:                As ← As'
32:                improvement ← True
33:                Break
34:        end for
35:        if improvement == True then
36:            Break
37:    end for
38: while improvement == True
39: return As
```

- checks if the agent a can accomplish m'. This operation is linear and only requires to check if $skill(a) \cap m' \neq \emptyset$, i.e., it has a cost of the number of skills in the worst case.
- computes the new cost $cost\ (As)$. This operation is constant and just removes the cost of the previous team assigned to a and add one of the new assigned team.
- updates the robustness of T. This operation is linear. As stated before, T has a vector where each component corresponds to a skill s of m' and the component value is the number of agents a' of T such that $s \in skill\ (a')$, i.e. this requires at most s operations.
- checks the cost bound. This operation is constant.
- checks the validity of As. This operation requires to compute the minimum number of agents needed to make T valid for each mission team (T, m').

This number is the difference between the current robustness k' of T and the required robustness k. Both k' and k are known, so that the whole operation has a linear complexity, i.e., $O(m)$. In case no pruning occurs, ST represents the maximal occurrences of the five above operations. Solving a MTF with this complete algorithm is then bound by $|ST| \times (t + 1 + s + 1 + m) = O((m + 1)^{n+1} \times (t + s + m))$ where t is the number of skills. Finally, when As is a complete assignment, the algorithm checks if As is dominated by the current Pareto optimal mission multi-teams. This operation is linear in the size of them, i.e., it compares the pair $(cost\ (As), k)$ with all elements of them, which has a complexity of $O(2n(m + 1)^n)$. Thus, the runtime complexity is bounded by $O((m + 1)^{n+1} \times (t + s + m)) + 2n(m + 1)^n)$ which belongs to $O(n(m + 1)^{n+2})$.

3.2 Approximate Algorithm

Since the complexity of the complete algorithm is exponential in the number of agents (in the worst case), an approximate algorithm is also provided that is based on local search technique. Algorithm 3 shows the pseudo-code of it. The input is a multi-teams formation problem MTF and a non-negative integer $MaxNeighbor$ which is used to limit the search. First, it generates a MMT randomly until a valid one is found (line 4). Then, a neighbor MMT is computed by randomly switching the mission team of an agent (line 9-13). If the selected MMT is not valid, another neighbor is generated. When a valid neighbor dominates the current candidate, i.e., both its cost and its robustness are better, it becomes the new candidate (line 15-17) and the search continues. Otherwise, another neighbor MMT is considered (line 18-19). The phase 1 terminates after $MaxNeighbor$ improvement failures (line 5-20). The phase 2 is a systematic search which ensures to reach a local optimal (line 21-38). It avoids to miss a good mission multi-team that will happen to be near the current candidate. In this phase, every neighbors assignments is checked and the current candidate is changed to its neighbor when a better candidate is found. The search continues until the current candidate is not dominated by any of its direct neighbor. Finally, it outputs the local optimal mission multi-team.

4 Evaluation

In this section, the complete and approximate algorithms are evaluated with RMASBench [Kleiner *et al.*, 2013], benchmarking platforms for agent-based system. RMASBench is a testbed for multi-agent coordination algorithms based on the existing RoboCup Rescue simulation platform (RSP) [Skinner and Ramchurn, 2010]. It simulates urban searches and rescue scenarios using real city maps. Figure 1 shows one of the representative scenarios, where Paris is the playground. In fig. 1, the 18 red points are the firemen agents and the 5 black crosses locate the initial ignition points where the fire start to spread. The RMASBench simulation platform introduces a generic API for multi-agent coordination, which essentially provides facilities for exchanging messages among agents and decisions making. It provides a library of implementation of state-of-the-art solvers, e.g., DSA [Fitzpatrick and Meertens, 2003] and MaxSum [Farinelli *et al.*, 2008], as well as facilities to compare coordination algorithms.

4.1 Context

In RMASBench, the coordination problem is about assigning fire brigades to fires in order to mitigate damages on buildings and try to stop fires propagation. The benchmark poses

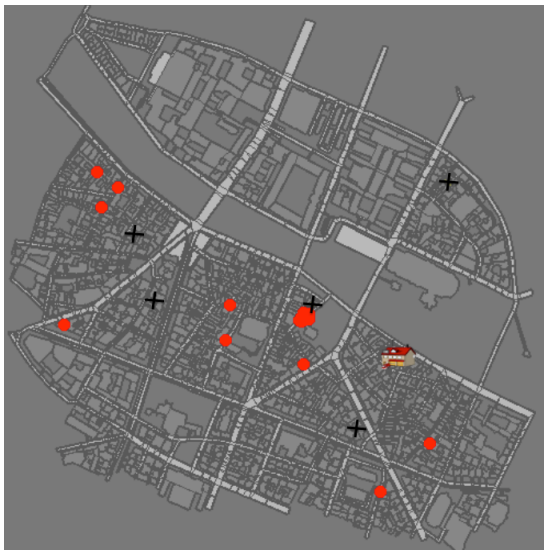

Figure 1: Initial state of the Paris map scenario with 18 firemen and 5 ignition points. Red points are firemen and crosses are ignition points.

crucial challenges for coordination. The agents evolve in a dynamic environment where a set of fires to be extinguished varies over the time. Fires may spread to neighboring buildings and grow in intensity as time goes, making them harder to extinguish and more likely to spread. The simulation is turn based (i.e. in each turn, the complete state of the world is known). It includes the positions of agents, the fires locations and the blockades places. Given the information, in each turn, the solver has to assign a target to each agent, i.e., a fire for each fireman and a blockade for each policeman. It has a strong spacial aspects: a fire brigade agent can operate on a given fire only if it is physically located in the proximity of that fire, hence agents must consider their travel time when coordinating. Moving to a distant location takes several turns, during which the state of the world changes. Agents have to change target because of the blockades appearance or buildings destruction. It includes the coalitional effects: several agents can work on the same fire, and the more agents work on the same fire the faster the fire can be extinguished. However, if too many units are allocated to the same fire, they might hinder each other or ignore other fires. The optimal mission teams of firemen have to be balanced, i.e., enough agents to fight the fire but not too many to avoid hinder.

An instance of the RMASBench can be modeled as a multi-team formation problem as follows. The set of agents contains only two kinds of agents, namely firemen and policemen. For each fireman agent a_i, $skill\ (a_i) = \{fire\}$ and for each policeman agent a_j, $skill\ (a_j) = \{blockades\}$. Each building on fire is represented by a mission $m_f = \{fire\}$ and each blockade by a mission $m_b = \{blockade\}$. For each mission, the required k-robustness is the fire intensity. The cost of a team is the sum of the distances that its members need to travel in order to reach the target. Since the number of missions explodes because of fire spread, it appends quickly that there is more missions than agents. In this case, it is not

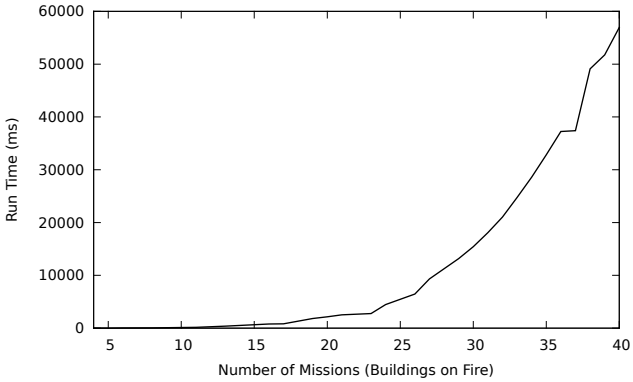

Figure 2: Run time of the complete algorithm.

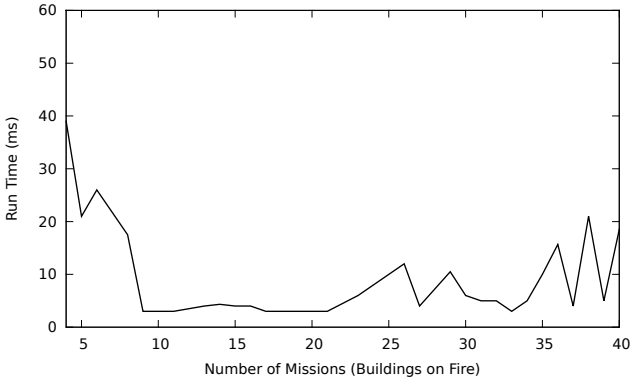

Figure 3: Run time of the approximate algorithm.

possible to assign a valid mission multi-team to each mission. To tackle this problem, we consider the number of unsatisfied missions during solving. For each turn, our two algorithms compute the mission multi-team so that the travel of team members is minimized and the fire fighting is maximized.

4.2 Experimental results

To evaluates the performances of complete and approximate algorithms, we focused on the benchmark scenario provided by RMASBench. Each experiment has been run on a computer Macbook Pro 13-inch mid 2010 (Intel(R) Core(TM) 2 Duo 2.4 GHz P8600). The scenario takes place on the map of Paris with 5 ignition points at start and 18 fire brigade agents (see Figure 1). The results show the simulation run for 300 turns and fire brigades remain idle for 23 turns before they become aware of all fires.

Figure 2 and 3 show the evolution of run time during the simulation. Solving time of our two algorithms are compared regarding the number of missions, i.e., the number of buildings on fire. We can see that the run time of the complete algorithm increases exponentially when the number of missions increases. This is why the number of solutions (mission multi-teams) increases exponentially. Compared to these results in fig.2, the approximate algorithm is more efficient. In this problem, the approximate algorithm re-uses the previous solution to start solving, and it allows us to generate a good

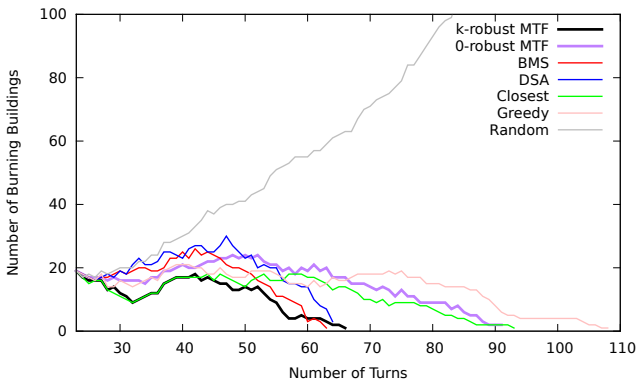

Figure 4: Evaluation of the approximate algorithm (k-robust MTF and 0-robust MTF) regarding the evolution of fire spread over time on the Paris scenario.

solutions quickly[1]. In the simulation, the number of fire increases quickly, thus the complete algorithm is too slow to solve a new problem for each turn. On the other hand, the approximate algorithm is a more realistic approach when it requires to solve the problem in a small amount of time.

Figure 4 shows the evolution of the number of buildings on fire during the simulation on the Paris scenario of fig.1. In these experiments, two versions of the approximate algorithms (i.e. k-robust MTF and 0-robust MTF) are compared with the five algorithms provided by RMASBench, namely BMS, DSA, Closest, Greedy and Random. The following is the quick explanations of those algorithms:

- The Random algorithm just assigns each agent to a fire location randomly.
- The Closest algorithm assigns each agent to its nearest fire.
- The Greedy algorithm does the same as Closest but considers the intensity of the fire and avoids to put many agents on the same target, i.e., when there is a fire with an intensity k and k agents assigned to it, it will prefer to assign the next agent to another fire location.
- BMS and DSA are the state-of-the-art solvers. These two algorithms optimize both the distance and the expected efficiency of each agent for each fire regarding its intensity. In those approaches, two metrics are combined into one value (scalarization) and they try to maximize this value.

The k-robust MTF approach uses the fire intensity k as the required robustness for each corresponding mission, as explained before, whereas the 0-robust MTF approach only requires one agent per fire. Both algorithms optimize the distance (i.e. the cost) and the robustness of each mission multi-team simultaneously. At each turn, we compute 100 local optimal solutions (i.e. mission multi-teams) and limit the local search by allowing 100 improvement failures ($MaxNeighbor$ = 100 in Algorithm 3). One of the non dominated mission multi-team we found is randomly selected amount those mission multi-teams.

In the experiments, the Random approach is the only one that did not succeed to stop the fire spread. The k-robust MTF succeeds to extinguish all fires after 66 turns, while Closest and Greedy need 92 and 108 turns respectively. The performances of 0-robust MTF are similar to those of Closest which finished in 92 turns, where 0-robust MTF finished in 91. BMS and DSA are the fastest to extinguish all fires, i.e., 63 and 64 turns. However, regarding the control of the fire spread, k-robust MTF gives better results. Indeed, the number of buildings on fire at the same time is lower for this algorithm: never exceeds 20 buildings where it reached 25 buildings for BMS and 30 buildings for DSA. Also, the k-robust MTF outperforms 0-robust MTF on both time and fire control. In Summary, these experimental results reveal that (i) our algorithm is more realistic for RMASBench compared to the complete algorithm and (ii) we empirically show that our algorithm can compete with the state-of-the-art solvers provided in RMASBench and considering the robust mission multi-teams (i.e. k-robust MTF) have a better control on the fire spread than the state-of-the-art solvers [2].

5 Conclusions & Future Works

In this paper, a novel framework for mission-oriented robust multi-team formation is introduced. The aim of this problem is to form robust teams which can achieve the given missions even if some agents break down, minimize the total cost and maximize the robustness of the teams. Furthermore, two algorithms for solving this problem is provided, namely, complete and approximate algorithms. Also, the complexity of the complete algorithm is discussed. In the experiments, these two algorithms are evaluated in RMASBench which is a testbed for multi-agent coordination algorithms based on the existing RoboCup Rescue simulation platform. Also, the performances of the approximate algorithm is compared with the existing solvers provided in RMASBench algorithms. We empirically showed that (i) the approximate algorithm is more realistic for RMASBench compared to the complete algorithm and (ii) our algorithm can compete with the state-of-the-art solvers provided in RMASBench and considering the robust mission multi-teams have a better control on the fire spread than the state-of-the-art solvers.

As a perspective for future research, we will develop efficient heuristics and algorithm which is specialized to the application problems. Also, we intend to apply our approach to some real-world problems, e.g., disaster medical assistance team formation and nurse rescheduling problem [Maenhout and Vanhoucke, 2011]. Moreover, we will consider to extend our framework to incorporate the solving of the job shop scheduling problem induced by the tasks of a mission. Given a limited amount of time for each mission, we could form robust mission multi-teams that can schedule their actions in order to accomplish their missions within the corresponding time limitations.

[1]The run time depends on the previous solution and the current situation that is given randomly.

[2]We observed that the essential result does not change in Kobe map scenario.

References

[Aarts and Lenstra, 2013] E. Aarts and J. Lenstra. Local search in combinatorial optimization. *Princeton University Press*, 2013.

[Farinelli *et al.*, 2008] 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 Conference on Autonomous Agents and Multiagent Systems*, pages 639–646, 2008.

[Fitzpatrick and Meertens, 2003] S. Fitzpatrick and L. Meertens. Distributed coordination through anarchic optimization. In *Distributed Sensor Networks*, pages 257–295. 2003.

[George *et al.*, 2010] J. George, J. Pinto, P. Sujit, and J. Sousa. Multiple uav coalition formation strategies. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, pages 1503–1504, 2010.

[Hua *et al.*, 2009] Q-S. Hua, D. Yu, F. Lau, and Y. Wang. Exact algorithms for set multicover and multiset multicover problems. In *Algorithms and Computation*, pages 34–44. 2009.

[James *et al.*, 2015] P. James, N. Ernesto, G. Julio, and G. Maria. Exploiting spatial locality and heterogeneity of agents for search and rescue teamwork. *Journal of Field Robotics*, pages 1–24, 2015.

[Kitano and Tadokoro, 2001] H. Kitano and S. Tadokoro. Robocup rescue: A grand challenge for multiagent and intelligent systems. *AI Magazine*, 22(1):39–52, 2001.

[Kleiner *et al.*, 2013] A. Kleiner, A. Farinelli, S. Ramchurn, B. Shiand F. Maffioletti, and R. Reffato. Rmasbench: benchmarking dynamic multi-agent coordination in urban search and rescue. In *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems*, pages 1195–1196, 2013.

[Lappas *et al.*, 2009] T. Lappas, K. Liu, and E. Terzi. Finding a team of experts in social networks. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 467–476, 2009.

[Lawler and Wood, 1966] E. Lawler and D. E. Wood. Branch-and-bound methods: A survey. *Operations research*, 14(4):699–719, 1966.

[Liemhetcharat and Veloso, 2012] S. Liemhetcharat and M. Veloso. Modeling and learning synergy for team formation with heterogeneous agents. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems*, pages 365–374, 2012.

[Maenhout and Vanhoucke, 2011] B. Maenhout and M. Vanhoucke. An evolutionary approach for the nurse rerostering problem. *Computers & OR*, 38(10):1400–1411, 2011.

[Nair and Tambe, 2005] R. Nair and M. Tambe. Hybrid bdi-pomdp framework for multiagent teaming. *Journal of Artificial Intelligence Ressearch*, 23:367–420, 2005.

[Okimoto *et al.*, 2012] T. Okimoto, Y. Joe, A. Iwasaki, T. Matsui, K. Hirayama, and M. Yokoo. Interactive algorithm for multi-objective constraint optimization. In *Proceedings of the 18th International Conference on Principles and Practice of Constraint Programming*, pages 561–576, 2012.

[Okimoto *et al.*, 2015] T. Okimoto, N. Schwind, M. Clement, T. Ribeiro, K. Inoue, and P. Marquis. How to form a task-oriented robust team. In *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems*, pages 395–403, 2015.

[Skinner and Ramchurn, 2010] C. Skinner and S. Ramchurn. The robocup rescue simulation platform. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, pages 1647–1648, 2010.

[Vidal, 2004] J. Vidal. The effects of co-operation on multiagent search in task-oriented domains. *Journal of Experimental and Theoretical Artificial Intelligence*, 16(1):5–18, 2004.