SAT Benchmarks for the Robust Team Formation Problem

Nicolas Schwind
National Institute of Advanced Industrial Science and Technology
Tokyo, Japan
nicolas-schwind@aist.go.jp

Emir Demirović
Delft University of Technology
Delft, The Netherlands
e.demirovic@tudelft.nl

Katsumi Inoue
National Institute of Informatics
The Graduate University for Advanced Studies, SOKENDAI
Tokyo, Japan
inoue@nii.ac.jp

Jean-Marie Lagniez
Université d'Artois
CRIL-CNRS
Lens, France
lagniez@cril.fr

Abstract—**Team formation (TF) is the problem of deploying the least expensive team of agents while covering a set of skills. It is almost equivalent to the Set Cover problem. Robust team formation (RobTF), a generalization of TF, takes into account the dynamic nature of the environment. Indeed, after a team has been formed, agents may unexpectedly become unavailable due to failure or illness. A team is said to be k-robust if it still covers the set of skills even after k agents are removed from it. Robustness is clearly a desirable property as it allows the underlying system to remain functional after an unfortunate event occurs. The decision problem for RobTF consists, given two integers k, b, in finding a k-robust team whose deployment cost is not greater than b. Interestingly, this problem is NP-complete, making it an appropriate target for SAT solvers.**

Index Terms—**team formation, SAT encoding.**

I. Introduction

Team Formation (TF) consists in forming a team of agents with minimum cost so as to meet a certain set of requirements. We are given a set of agents, where each agent is associated with a set of skills and a deployment/hiring cost. The TF problem consists in finding a team T (i.e., a subset of agents) of minimal overall cost that is *efficient*, i.e., such that each skill is possessed by at least one agent from T. This is an important problem in multi-agent systems and has been studied in the contexts of RoboCup rescue teams [8], unmanned aerial vehicle operations [5], social networks [3], [7], online football prediction games [9], among others. This problem is well-known to be NP-hard [4], [10]. The (standard) TF problem [10] can be formally defined as follow:

Definition 1 (TF Problem Description)**.** A *TF problem description* is a tuple $\langle A, S, f, \alpha \rangle$ where $A = \{a_1, \dots, a_n\}$ is a set of agents, $S = \{s_1, \dots, s_m\}$ is a set of skills, $f : A \mapsto \mathbb{N}$ is a deployment cost function, and $\alpha : A \mapsto 2^S$ is an agent-to-skill function.

A *team* is a subset of agents $T \subseteq A$. One extends the cost function f to teams T as $f(T) = \sum_{a_i \in T} f(a_i)$. Likewise, the agent-to-skill function α is extended to teams T as $\alpha(T) = \bigcup_{a_i \in T} \alpha(a_i)$. A standard expected property in Team Formation is efficiency: a team $T \subseteq A$ is *efficient* if all skills from S are covered by T, i.e., when $\alpha(T) = S$. An optimal team for TF is an efficient team minimizing the cost function. The corresponding decision problem asks, given a TF problem description and a bound $b \in \mathbb{N}$ as input, whether there exists an efficient team T such that $f(T) \leq b$. This problem is equivalent to the well-known set cover problem [4] and is NP-complete [10].

In realistic settings, we may not be certain about the actual functionality of all agents from a team after deployment. Some unexpected, exogenous events often occur: agents get sick or may be unable to do the job for various reasons, and we may not be certain about the actual functionality of all agents from a team after deployment. Thus it is important to consider *robustness* properties in TF, i.e., seek to form an efficient team that is proactive to such unfortunate changes.

The notion of robustness in Team Formation has been introduced and formalized in [10]:

Definition 2 (Robust Team [10])**.** Given a TF problem description $\langle A, S, f, \alpha \rangle$ and $k \in \mathbb{N}$, a team T is said to be *k-robust* if for every set of agents $T' \subseteq T$ such that $|T'| \leq k$, the team $T \setminus T'$ is efficient.

Indeed, a team T is k-robust if it remains efficient in any case where at most k agents are removed from it. Robust TF (RobTF) consists in finding an optimal k-robust team, i.e., a k-robust team of minimal deployment cost. It provides a guarantee that the goal is fulfilled in the worst case given k. Robustness generalizes efficiency: a team is 0-robust if and only if it is efficient. Interestingly, checking whether a team is k-robust only requires to check that every skill is covered at least $k+1$ times [10], and this can be done in polynomial time. Hence, despite this generalization, finding an optimal k-robust

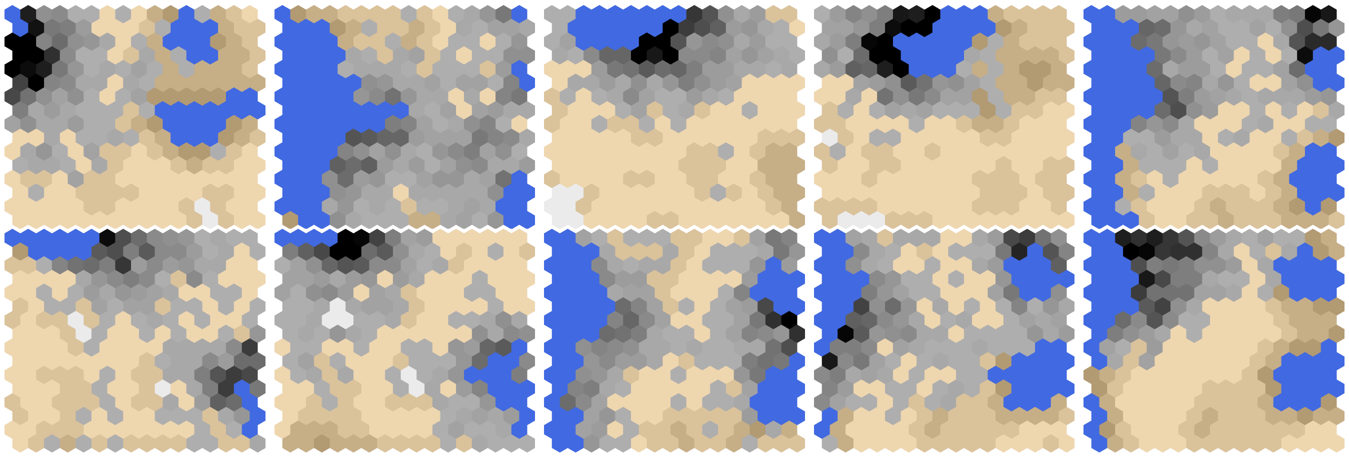


Fig. 1. Some examples of populated maps.

team (for any $k \geq 0$) does not lead to a computational shift compared to finding an optimal efficient team. Let us formalize the decision problem for RobTF, which is the problem of interest here:

Definition 3 (DPRobTF).

- **Input:** A TF problem description, two integers k, b.
- **Question:** Does there exist a k-robust team $T \subseteq A$ such that $f(T) \leq b$?

Theorem 1 ([10]). DPRobTF is NP-complete.

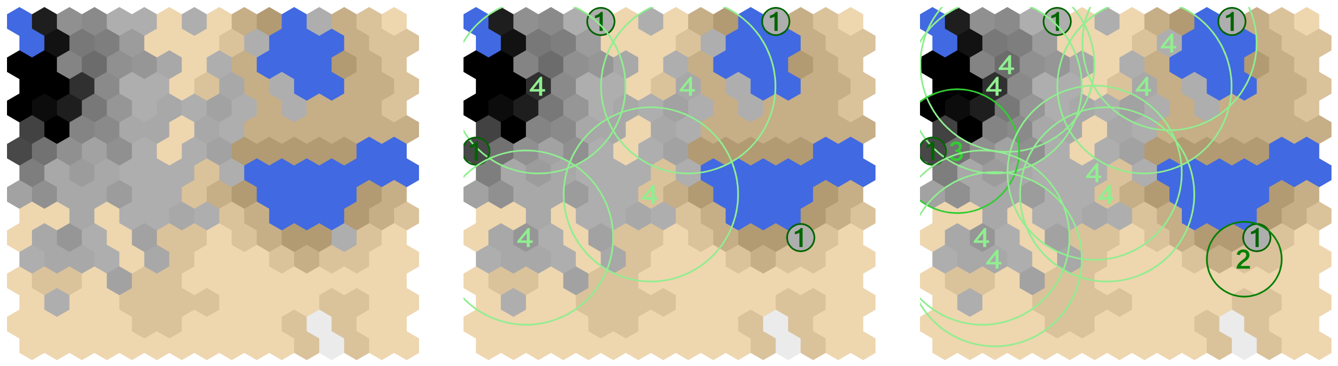
II. Robust Facility Deployment

In [12], we have introduced a generator of TF problem descriptions modeling facility deployment problem instances. This problem consists in deploying a set of facilities (e.g., health centers, antennas, schools, shelters) on a populated map so as to maximize a certain population coverage while minimizing the overall deployment cost [1]. The problem is of particular importance, e.g., for mobile phone operators which aim to deploy a set of cell towers in an urban environment. Finding an optimal efficient team allows one to find a facility deployment of minimal cost while providing a service coverage over the whole population. In such an application context, each populated cell corresponds to a skill to be covered, each facility at a given location corresponds to an agent, and the skills associated with a facility are the populated cells in its range. Each TF problem instance was synthesized following three steps.

First, one generates an elevation map made of water parts, lands and mountains. A 16x16 grid of numbers is created using Perlin noise [11], a procedural texture primitive commonly used by visual effects artists to increase the appearance of realism in computer graphics. The grid is then converted into a hexagonal grid for which each cell is associated with a "type" depending on the range of its value according to the grid. A low (resp., mid, high) value is interpreted as a water cell (resp., a mountain cell, a land cell).

Second, the map is populated by iteratively adding an individual on the grid. Initially, three individuals are added in three different land cells randomly chosen, provided that the cell is next to a water cell. Then, a new individual is added at random following a probability distribution. The water cells and the cells that already host 10 individuals cannot host a new individual. Among the remaining cells, the closer to a densely populated cell or to water cell, the higher its probability to welcome a new individual. The process is repeated 600 times which at last corresponds to the total population in the map. Fig. 1 depicts ten populated maps generated using this method: blue (resp. white, brown) cells are of water type (resp. mountain, field type). The scales of brown correspond to different elevation degrees of land. The scales of gray represent the number of individuals in a cell: the darker a cell, the more densely populated, so a pitch black cell contains 10 individuals. The different scales of brown/gray do not have an impact on the final TF instance, but were used to tune the probability of adding an individual to a land cell, so that to make the instance more realistic.

Third, a populated map is translated into a weighted TF problem description $\langle A, S, f, \alpha \rangle$ as follows. We consider four types of agents $type_1, \ldots, type_4$. Each type $type_i$ of agent corresponds to a facility that has a deployment cost equal to i and a cover range equal to $i - 1$. For instance, a cell tower of type $type_3$ has a deployment cost equal to 3, and when it is deployed in a certain cell C on the grid, it provides the required service to anyone that is in a cell C' where the distance between C and C' is at most 2; the distance between two cells on the grid corresponds to the length of the shortest path between C and C'. So for each type of facility $type_i$ and each grid cell j that is not of water type, one considers an agent a_i^j of cost $f(a_i^j) = i$ which corresponds to a facility of type $type_i$ to be potentially deployed in the cell j. This defines the set A of agents and the cost function f. To define the set of skills S, one simply associates each populated grid cell p (i.e., a cell that hosts at least one individual) with a skill s_p. Lastly, the agent-to-skill mapping α is defined as follows. An agent a_i^j has the skill s_p if the grid cell p is within the reach of the facility a_i^j, i.e., if the distance between the grid cells j and p is less than or equal to $i - 1$. The generated instances



(a) A populated map generated by our procedure. (b) An optimal 0-robust team T_1 ($f(T_1) = 20$, $rc(T_1, 1) = 13$, $pc(T_1, 1) = .20$). (c) An optimal 1-robust team T_2 ($f(T_2) = 41$, $rc(T_2, 1) = 0$, $pc(T_2, 1) = 1$).

Fig. 2. Optimal 0-robust and 1-robust teams for a given instance.

were formed of 500 to 700 agents and 50 to 150 skills.

Given such an instance $\langle A, S, f, \alpha \rangle$, a team T corresponds to a set of facilities to be deployed on the corresponding map. Fig. 2 depicts for a given map instance the optimal team computed, that is respectively 0-robust, or equivalently, efficient (Fig. 2(b)) and 1-robust (Fig. 2(c)). For instance, the optimal 0-robust team (Fig. 2(b)) is formed of eight agents corresponding to four facilities of type $type_4$ and four facilities of type $type_1$. Each such facility is represented by a label on the corresponding grid cell ranging from 1 to 4, corresponding to its type / deployment cost. The circle drawn around each deployed facility represents the populated area that is covered by it, i.e., the set of skills possessed by the corresponding agent.

III. SAT Encoding

We are interested in formalizing the decision problem DPRobTF given in Definition 3 as a SAT problem. To do so, we first associate a boolean variable p_i with each agent in $a_i \in A$, where p_i is true if and only if the agent a_i is present in the deployed team. The encoding is described as follows. Given a TF instance $\langle A, S, f, \alpha \rangle$ and $k, b \in \mathbb{N}$, a k-robust team T exists if and only if:

$$\bigwedge_{s_j \in S} \sum_{a_i \in A | s_j \in \alpha(i)} p_i > k \tag{1}$$

$$\sum_{a_i \in A} f(i) \times p_i \leq b \tag{2}$$

Equation 1 ensures that at least $k + 1$ agents present in the deployed team the team. This equation precisely ensures that the team is k-robust [10]. It consists in a set of cardinality constraints encoded into CNF using PySAT [6]. In the case where $k = 0$, the cardinality constraints are encoded as clauses. For the general case where $k > 0$ the cardinality constraints are encoded into SAT using the totalize encoding [2]. Equation 2 ensures that the cost of the deployed team is not more than the bound b. It consists in one pseudo-boolean constraint that has been encoded into SAT using the encoding type *best* provided by PySAT.

IV. Benchmarks

We submitted 20 benchmarks to the 2021 SAT Competition. The benchmarks are named as $ktf\_\langle bench.tf \rangle\_\langle k \rangle\_\langle ratio \rangle\_\langle b \rangle$, where:

- $bench.tf$ is the TF instance used to generate the CNF formula,
- k is requirement of k-robustness,
- $ratio$ is a ratio (a value between 0 and 1) of the overall cost the set of all agents, used to define the bound b; for instance, one sets a ratio of 0.6 when one wants the cost of the deployed team to be less than or equal to 60% of the total cost of all agents.
- b is the bound defined as $b = ratio \times \sum_{a_i \in A} f(a_i)$.

All the materials used to generate the benchmarks are available at https://github.com/jm62300/team-formation.

References

[1] A. Ahmadi-Javid, P. Seyedi, and S. S. Syam, "A survey of healthcare facility location," *Comput. Oper. Res.*, vol. 79, pp. 223–263, 2017.

[2] O. Bailleux and Y. Boufkhad, "Efficient CNF encoding of boolean cardinality constraints," in *Proceedings of the 9th International Conference on Principles and Practice of Constraint Programming (CP'03)*, F. Rossi, Ed., 2003, pp. 108–122.

[3] F. Farhadi, E. Hoseini, S. Hashemi, and A. Hamzeh, "Teamfinder: A co-clustering based framework for finding an effective team of experts in social networks," in *12th IEEE International Conference on Data Mining Workshops (ICDM'12)*, 2012, pp. 107–114.

[4] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[5] J. M. George, J. Pinto, P. B. Sujit, and J. B. Sousa, "Multiple UAV coalition formation strategies," in *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'10)*, 2010, pp. 1503–1504.

[6] A. Ignatiev, A. Morgado, and J. Marques-Silva, "PySAT: A Python toolkit for prototyping with SAT oracles," in *Proceedings of the 21st International Conference on Theory and Applications of Satisfiability Testing (SAT'18)*, 2018, pp. 428–437.

[7] M. Kargar, A. An, and M. Zihayat, "Efficient bi-objective team formation in social networks," in *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases Part II (ECML-PKDD'12)*, 2012, pp. 483–498.

[8] H. Kitano and S. Tadokoro, "Robocup rescue: A grand challenge for multiagent and intelligent systems," *AI Mag.*, vol. 22, no. 1, pp. 39–52, 2001.

[9] T. Matthews, S. D. Ramchurn, and G. Chalkiadakis, "Competing with humans at fantasy football: Team formation in large partially-observable domains," in *Proceedings of the 26th AAAI Conference on Artificial Intelligence (AAAI'12)*, 2012, pp. 1394–1400.

[10] 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 2015 International Conference on Autonomous Agents and Multiagent Systems (AAMAS'15)*, 2015, pp. 395–403.

[11] K. Perlin, "An image synthesizer," in *Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH'85)*, 1985, pp. 287–296.

[12] N. Schwind, E. Demirović, K. Inoue, and J.-M. Lagniez, "Partial robustness in team formation: Bridging the gap between robustness and resilience," in *Proceedings of the 2021 International Conference on Autonomous Agents and Multiagent Systems (AAMAS'21)*, 2021, to appear.