SAT Encoding for the 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. Robust team formation (RobTF), a generalization of the above problem, takes into account the dynamic nature of the environment, e.g. 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 the team remains efficient 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 change happens. Interestingly, the decision problems for TF and RobTF are both NP-complete, making them possible 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 $A = \{a_1, \ldots, a_n\}$ is a set of agents, $S = \{s_1, \ldots, s_m\}$ 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 threshold $c \in \mathbb{N}$ as input, whether there exists an efficient team T such that $f(T) \leq c$. This problem is equivalent to the well-known set cover problem [4].

In realistic settings, we may not be certain about the actual functionality of all agents from a team after computation and deployment. Some unexpected, exogenous events often occur: agents get sick or may be unable to do the job for various reasons. Yet we may mot be certain about the actual functionality of all agents from a team after formation and deployment. Thus it is important to consider resilience properties in TF, i.e., seek to form an efficient team that is proactive to changes, i.e., it remains efficient even when some agents are removed from the deployed team. It is why the notion of robustness has been introduced in [10].

A team T is k-robust if it remains efficient in any case where at most k agents are removed from it [10]. Robust TF 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. This involves introducing redundant agents which may not be necessary if the worst case does not happen. Interestingly, computing an optimal robust team is not harder than computing an optimal efficient team [10], since it only requires to cover every skill at least $k + 1$ times. Formally, we have:

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.

Robustness generalizes efficiency: a team is 0-robust iff it is efficient. Interestingly, despite this generalization, computing an optimal k-robust team (for any $k \geq 0$) does not lead to a computational shift. Indeed, the decision problem for robustness, which asks, given a TF problem description and c, k in $\mathbb{N}$, if there exists a k-robust team $T \subseteq A$ such that

$f(T) \leq c$, is NP-complete.

II. Team Formation Problem: Facilities Deployment

In [12], we artificially generate instances 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, facilities correspond to agents and the population to be covered correspond to a weighted skill (the weight depends on the density of the population at a precise location).

Each weighted 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 an already populated cell or to river 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(a) represents a populated map generated using this method: blue (resp. white, brown) cells are of water type (resp. mountain, field type). Different scales of brown correspond to different elevation degrees of land, only used to tune the probability of adding an individual to a land cell. The gray scales represent the number of individuals in a cell. The darker a cell, the more densely populated, so a pitch black cell contains 10 individuals.

Third, a populated map is translated into a weighted TF problem description $\langle A, S, f, w_\Sigma, \alpha \rangle$ 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' such that 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. The set of skills S and the skill weight function w_Σ (note that we considered a normalized weighted sum function) are simply defined as follows. One associates with each populated grid cell p (i.e., a cell that hosts at least one individual) a skill s_p; and the weight of each skill $w_\Sigma(s_p)$ is defined as the number of individuals in the grid cell 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 or equal to $i-1$. The generated instances were formed of 500 to 700 agents and 50 to 150 skills.

Given such an instance $\langle A, S, f, w_\Sigma, \alpha \rangle$, a team T corresponds to a set of facilities to be deployed on the corresponding map. Fig. 1 depicts for a given map instance the optimal team computed, that is respectively efficient (Fig. 1(b)) and 1-robust (Fig. 1(c)). For instance, the optimal efficient team (Fig. 1(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. Let us remark that in the context we are interested in w_Σ is not relevant since we search a team that satisfies all the skills. The case where w_Σ is relevant concerns other team formation behaviour that are out of the scope of this system description (interested readers can refer to [12] for other team formation behaviour).

III. SAT Encoding

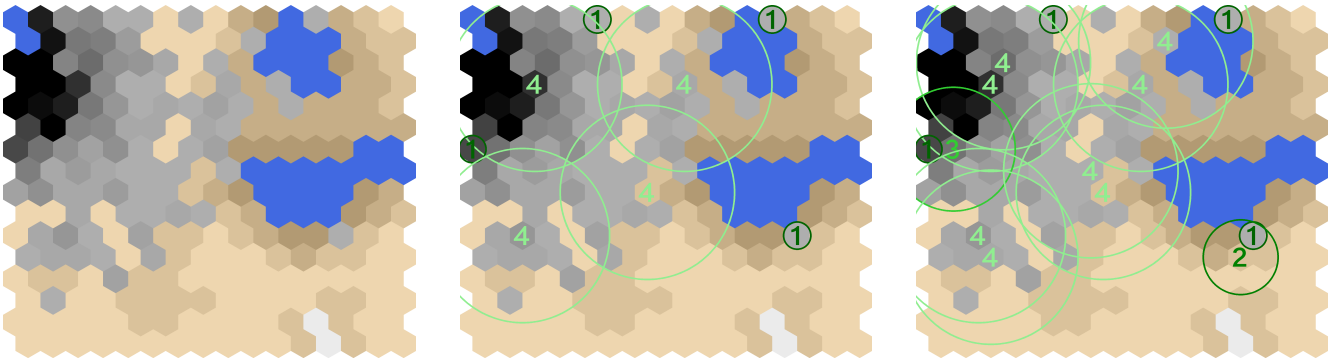
Because we are interest in the decision problem a bound $b \in \mathbb{N}$ is considered. The objective is then to find out a team such that its computed cost is less than b. To translate our problem into SAT we first associate a boolean variable p_i for each agent in $i \in A$, where p_i set to true (resp. false) means that agent i is (resp. is not) present in the deployed team. In the following we describe the encoding for the k robust case, the efficient case beeing the special case where $k = 0$.

Given an instance $\langle A, S, f, w_\Sigma, \alpha \rangle$ and $b \in \mathbb{N}$, a k robust team T exists if and only if:

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

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

Equation 1 ensures that enough agents are present in the deployed team. 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



(a) A populated map generated by our procedure. (b) An optimal efficient 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. 1. Optimal teams for different solution concepts.

general case where $k > 0$ the cardinalty constraints are encoded into SAT using the totalize encoding [2]. Equation 2 controls the team cost. 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 X benchmarks to the 2021 SAT Competition. The benchmarks follow the following naming convention:

- Efficient team: $tf\_\langle bench.tf\rangle\_\langle ratio\rangle\_\langle bound\rangle$, where $bench.tf$ is the team formation input that has been used to generate the CNF formula, $ratio$ is a ratio of the maximum cost we can deployed, and $bound$ is the given bound ($bound = ratio \times \sum_{i \in A} f(i)$);
- k robust team: $ktf\_\langle bench.tf\rangle\_\langle k\rangle\_\langle ratio\rangle\_\langle bound\rangle$, where $bench.tf$ is the team formation input that has been used to generate the CNF formula, k is the maximal number of agents we can lose, $ratio$ is a ratio of the maximum cost we can deployed, and $bound$ is the given bound ($bound = ratio \times \sum_{i \in A} f(i)$);

All the material used to generate the benchmarks are available at ???.

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 *Principles and Practice of Constraint Programming - CP 2003, 9th International Conference, CP 2003, Kinsale, Ireland, September 29 - October 3, 2003, Proceedings*, ser. Lecture Notes in Computer Science, F. Rossi, Ed., vol. 2833. Springer, 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 Workshops, Brussels, Belgium, December 10, 2012*, J. Vreeken, C. Ling, M. J. Zaki, A. Siebes, J. X. Yu, B. Goethals, G. I. Webb, and X. Wu, Eds. IEEE Computer Society, 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 *9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2010), Toronto, Canada, May 10-14, 2010, Volume 1-3*, W. van der Hoek, G. A. Kaminka, Y. Lespérance, M. Luck, and S. Sen, Eds., pp. 1503–1504.

[6] A. Ignatiev, A. Morgado, and J. Marques-Silva, "PySAT: A Python toolkit for prototyping with SAT oracles," in *SAT*, 2018, pp. 428–437.

[7] M. Kargar, A. An, and M. Zihayat, "Efficient bi-objective team formation in social networks," in *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2012, Bristol, UK, September 24-28, 2012. Proceedings, Part II*, P. A. Flach, T. D. Bie, and N. Cristianini, Eds., 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 Twenty-Sixth AAAI Conference on Artificial Intelligence, July 22-26, 2012, Toronto, Ontario, Canada*, J. Hoffmann and B. Selman, Eds. AAAI Press, 2012. [Online]. Available: http://www.aaai.org/ocs/index.php/AAAI/AAAI12/paper/view/5088

[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 2015, Istanbul, Turkey, May 4-8, 2015*, 2015, pp. 395–403.

[11] K. Perlin, "An image synthesizer," in *Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1985, San Francisco, California, USA, July 22-26, 1985*, 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 2021, virtual, May 3-7, 2021*, 2021.