Partial Robustness in Team Formation

Bridging the Gap between Robustness and Resilience

Schwind, Nicolas; Inoue, Katsumi; Demirović, E.

Publication date
2021

Document Version
Final published version

Published in
Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems

Partial Robustness in Team Formation: Bridging the Gap between Robustness and Resilience

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, Tokyo, Japan
The Graduate University for Advanced Studies, SOKENDAI, Tokyo, Japan
inoue@nii.ac.jp

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

ABSTRACT

Team formation is the problem of deploying the least expensive team of agents while covering a set of skills. Once a team has been formed, some of the agents considered at start may be finally defective and some skills may become uncovered. Two solution concepts have been recently introduced to deal with this issue in a proactive manner: one may form a team which is *robust* to changes so that after some agent losses, all skills remain covered; or one may opt for a *recoverable* team, i.e., it can be "repaired" in the worst case by hiring new agents while keeping the overall deployment cost minimal. In this paper, we introduce the problem of *partially robust team formation* (PR-TF). Partial robustness is a weaker form of robustness which guarantees a certain degree of skill coverage after some agents are lost. We analyze the computational complexity of PR-TF, and provide a complete algorithm for it. The performance of our algorithm is empirically compared with the existing methods for robust and recoverable team formation, on a number of existing benchmarks and some newly introduced ones. Partial robustness is shown to be an interesting trade-off notion between (full) robustness and recoverability in terms of computational efficiency, skill coverage guarantees after agent losses, and repairability.

KEYWORDS

Team Formation, Robustness, Resilience

ACM Reference Format:
Nicolas Schwind, Emir Demirović, Katsumi Inoue, and Jean-Marie Lagniez. 2021. Partial Robustness in Team Formation: Bridging the Gap between Robustness and Resilience. In *Proc. of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021), Online, May 3–7, 2021*, IFAAMAS, 9 pages.

1 INTRODUCTION

Team Formation (TF) consists in forming a team of agents with minimum cost so as to meet a certain set of requirements. In its most abstract form, the problem is equivalent to the set covering and hitting set problems [11]. 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 problem is well-known to be **NP**-hard [11, 20].

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

Robustness [20] and recoverability [8] are complementary notions in TF, with both their own advantages and drawbacks. A team T is k-robust if it remains efficient in any case where at most k agents are removed from it [20]. 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. Interestingly, computing an optimal robust team is not harder than computing an optimal efficient team [20]. However, the deployment cost of a robust team may be prohibitively high, as it requires to cover every skill at least $k+1$ times.

Forming an optimal *recoverable* team is a cheaper alternative [8]: a team T is k-recoverable if, whichever k agents from T are lost, one can "repair" the residual team by hiring available agents while keeping the overall cost (i.e., the cost of the initial deployment of T plus the recovery cost in the worst case) minimal. Demirović et al. [8] empirically showed that the overall deployment cost is effectively lower than the initial deployment cost of a robust team. However, the problem of computing a recoverable team is $\Sigma_3^{\mathbf{P}}$-hard [8], making it unapplicable in practice. Moreover, recoverability does not provide any coverage guarantee during the disaster phase. As a result, a large amount of skills may become uncovered for a certain amount of time during which the system loses most of its functionality. This is a situation we want to avoid.

Let us introduce an example illustrating the problem and notions.

Example 1.1. The organizers of a special exhibition have a budget of 900 to hire professional translators: the attendees are expected to be from China (50%), Japan (40%), and France (10%). A number of translators are available for hire. A type 1 translator possesses exactly one of the three language skills. It costs 100 to hire a type 1 Chinese or Japanese translator, and 150 for a French translator. Let us denote by C (resp. J, F) a type 1 translator having the Chinese

Proc. of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021), U. Endriss, A. Nowé, F. Dignum, A. Lomuscio (eds.), May 3–7, 2021, Online.

(resp. Japanese, French) language ability. A type 2 translator has exactly two of these language skills, denoted respectively by CJ, CF, and FJ. It costs 180 to hire CJ, 230 for CF or FJ. A candidate must be paid in advance and should be contacted at least one day ahead of the event, otherwise she can only be on site during the afternoon and for the same price.

One of the cheapest alternative (plan I) is to hire the team $\{C, FJ\}$. Doing so, attendees from China can get help from the C agent, while attendees from France and Japan can rely on the FJ agent. This plan corresponds to an optimal team for the standard TF problem and costs 330. In the unfortunate case where both of these agents fall sick on the day of the event, another pair of translators can be hired in emergency (additionally pricing 330). However, under the same circumstances a slightly cheaper solution exists (plan II), which consists in forming the team $\{C, F, J\}$. At an initial cost of 350, in the worst case where two agents including F are absent on the day of the exhibition, one can hire for the afternoon a type 2 agent possessing the two languages skills that have been lost (e.g., if we lack agents F and J from the hired team, one would ask help from an FJ agent.) Plan II is slightly more costly than plan I (350 vs. 330), but the recovery price is 230 instead of 330, making this alternative arguably preferable than the first one. Plan II corresponds to an optimal 2-recoverable team. However, in both plans I and II, if we were to lose two agents at the last minute then the whole morning would lack at least two language abilities (all of them in the case of plan I). Assuming we would like to be *robust* to a loss of two agents, one of the cheapest plan is, e.g., to form the team $\{C, CJ, CF, JF, JF\}$ (plan III): since each language skill is possessed by at least three agents from the team, losing two agents would not compromise the goal. Plan III corresponds to an optimal 2-robust team. However, plan III is quite expensive: it costs 970, so it goes above the budget of 900 and the organizers cannot afford it.

The organizers would be happy with an alternative that is more affordable than plan III, while still being "robust" to potential losses. Noteworthy, only 10% of the attendees are expected to be from France, and French translators are more costly than the other ones. Thus a reasonable alternative is to hire the team $\{CJ, CJ, CJ, F\}$ (plan IV), costing 690. Losing two agents from it would still guarantee Chinese and Japanese language translation services, while losing F would only result in a coverage loss of 10% among the attendees. In addition, repairing the team would not cost more than 150 in the worst case (one would need to hire an F agent), so the overall cost of 840 would still remain under the budget constraints.

Our paper aims to introduce the solution concept illustrated above in plan IV. A team T is said to be $\langle k, t\rangle$-*partially robust* ($t \in [0, 1]$) if whenever k agents are removed from T, some "proportion" (reflected by t) of the overall set of skills remains covered. *Partially Robust TF* (PR-TF) is the problem to form an optimal $\langle k, t\rangle$-partially robust team. Plan IV in the above example corresponds to an optimal $\langle 2, .9\rangle$-partially robust team. This notion generalizes (full) robustness: a team is $\langle k, 1\rangle$-partially robust if and only if it is k-robust. Computationally speaking, we show that the decision problem related to PR-TF is $\Sigma_2^{\mathbf{P}}$-complete, thus it lies "in-between" robust TF and recoverable TF. We empirically show that forming an optimal partially robust team has the advantages of both robustness and recoverability. Indeed, on the one hand, a partially robust team can be computed much more efficiently than a recoverable one, and by definition it provides a skill coverage guarantee in the disaster phase; on the other hand, the overall cost of a partially robust team is shown to be much cheaper than the initial deployment cost of a "fully" robust team.

The proofs of propositions are available online [1].

2 RELATED WORK

Robustness and TF, both as separately and in combination, have attracted attention in a variety of contexts. In the following we focus on works that are closely related to our setting.

Stochastic optimization [27] is a general term for optimization methods dealing with models that include randomness. Randomness is captured in a given set of possible *scenarios* along with the probability of each scenario taking place. The goal is to compute a solution that minimizes the cost on average across all scenarios. Our setting could be framed as a scenario-based method, where there are exponentially many scenarios, each corresponding to a specific combination of agent loss, and the task is to optimize the worst case, i.e., minimize the maximum objective value across each scenario. Our approach avoids exhaustive enumeration through a cut generation approach, effectively including only the scenarios that are necessary for computing the optimal solution.

Robustness and recoverability have been studied in system design: robustness [5] is the ability to withstand adversarial conditions without negative impact on performance, and recoverability [25] is the capability to restore the functionality of the system after disturbance. Recoverability can be seen as a generalization of robustness.

In combinatorial optimization, a related resilience definition been studied under the name *super solutions* [13], where the task is to compute an (a, b)-solution such that if any a variables lose their values, a new satisfying assignment may be constructed by selecting new values for the a variables and changing at most b other variable values. Initially, the focus was on $(1, 0)$-super solutions, corresponding to robust solutions, and recently a generic approach for arbitrary values of a and b has been proposed [6] based on Logic-based Bender Decomposition [14]. This approach is similar to Counter-Example-Guided Abstraction Refinement (CEGAR) [7, 15, 16], in the more canonical problem of QBF solving, briefly discussed in Sec. 4 before introducing our algorithm. The algorithm for recoverable TF [8] and our method are similar in essence: initially a simplified problem is considered, and in each iteration of the algorithm a new solution is computed, based on which a new cut is generated and added to the problem, and the process restarts until optimality is proven. An alternative approach is to explicitly encode each scenario as done in a propositional logic setting [4], but this becomes infeasible for larger a and b values. While super solutions are desirable, the high computational complexity may render them impractical. This is the key issue we address in this paper, i.e., introduce a notion capturing aspects of resilience while reducing the computational burden.

Coalition structure generation [24] is a problem related to TF, where a set of agents is to be partitioned into a number of teams to maximize utility. Robustness [21] and stochastic [26] notions have been proposed for this setting. Other variants of TF have been

[1] https://nicolas-schwind.github.io/SDIL-AAMAS21-proofs.pdf

proposed in the literature, but to the best of our knowledge, there are no other works that bear a tight connection to ours.

3 PRELIMINARIES

This section recalls some preliminaries about basic notions of computational complexity, the Team Formation (TF) problem, and its extensions to Robust TF [20] and Recoverable TF [8].

3.1 Computational Complexity

We assume that the reader is familiar with the complexity class **NP** (see [22] for more details). Higher complexity classes are defined using oracles. In particular, $\Sigma_2^{\mathbf{P}} = \mathbf{NP}^{\mathbf{NP}}$ (resp. $\Sigma_3^{\mathbf{P}}$) corresponds to the class of decision problems that are solved in polynomial time by non-deterministic Turing machines using an oracle for **NP** (resp. $\Sigma_2^{\mathbf{P}}$) in polynomial time.

3.2 Team Formation

Let us formalize the (standard) TF problem [20].

Definition 3.1 (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 DP-TF 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 [11]:

Proposition 3.2. *[20]* DP-TF *is* **NP***-complete.*

3.3 Robust TF

Definition 3.3 (Robust Team [20]). 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 if and only if 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 (labeled DP-RobTF) asks, given a TF problem description and c, k in $\mathbb{N}$, if there exists a k-robust team $T \subseteq A$ such that $f(T) \leq c$. Then:

Proposition 3.4 ([20]). DP-RobTF *is* **NP***-complete.*

This problem still lies in **NP** because checking whether a given team T is k-robust, despite its combinatorial nature, is equivalent to checking whether each skill from S is possessed by at least $k+1$ agents from T; and this task can be performed in polynomial time. The goal of robust TF (RobTF) is to find an *optimal* k-robust team, i.e., a k-robust team T such that $f(T)$ is minimal.

3.4 Recoverable TF

Recoverable TF (RecTF) consists in finding a team that can be repaired after k agents are removed from it. The notion is based on an extension of a TF problem description (cf. Def. 3.1):

Definition 3.5 (RecTF Problem Description [8]). A *RecTF problem description* is a tuple $\langle A, S, f, \alpha, h \rangle$ where $\langle A, S, f, \alpha \rangle$ is a TF problem description and $h : A \mapsto \mathbb{N} \cup \{+\infty\}$ is a recovery cost function.

So a RecTF problem description considers in addition a recovery cost function h that defines the cost of deploying a "rescue" team. It adds more flexibility to the framework: some agents a_i may be deployed at a higher cost later in an emergency situation ($h(a_i) > f(a_i)$) or not be available at all ($h(a_i) = +\infty$). Similarly to f, for any team T one sets $h(T) = \sum_{a_i \in T} h(a_i)$. Given a team $T \subseteq A$ and $T' \subseteq T$, $rcS(T, T')$ is defined as the cost of the cheapest team T_{rec} such that $(T \setminus T') \cup T_{rec}$ is efficient:

$$rcS(T, T') = \min_{T_{rec} \subseteq A \setminus T, (T \setminus T') \cup T_{rec} \text{ is efficient}} h(T_{rec}).$$

The *k-recovery cost* of T is then defined as the highest value $rcS(T, T')$ for any set T' of size lower or equal to k:

$$rc(T, k) = \max_{T' \subseteq T, |T'| \leq k} rcS(T, T').$$

Definition 3.6 (Recoverable Team [8]). Given a RecTF problem description $\langle A, S, f, \alpha, h \rangle$ and non-negative integers k, r, a team T is said to be *$\langle k, r \rangle$-recoverable* if T is efficient and $rc(T, k) \leq r$.

Recoverability generalizes robustness: if $h(a_i) > 0$ for any $a_i \in A$, any team T is $\langle k, 0 \rangle$-recoverable if and only if T is k-robust. The decision problem for recoverability (labeled DP-RecTF) asks, given a TF problem description and c, r in $\mathbb{N}$, if there exists a $\langle k, r \rangle$-recoverable team $T \subseteq A$ such that $f(T) \leq c$. It turns out that this problem is much harder than the robustness counterpart:

Proposition 3.7 ([8]). DP-RecTF *is* $\Sigma_3^{\mathbf{P}}$*-complete.*

The goal of RecTF is to compute an *optimal* k-recoverable team, which is defined as a team T that is $\langle k, r \rangle$-recoverable which minimizes its *overall cost* $f(T) + rc(T, k)$.

4 PARTIAL ROBUSTNESS IN TF

We introduce a new solution concept for TF called *partial robustness*. Intuitively, a team is partially robust if it is efficient, and if after removing a certain number of agents from it, the residual team covers a certain proportion of the set of all skills. Thus, it makes sense to associate each skill with a *weight* to emphasize its relative importance:

Definition 4.1 (Weighted TF Problem Description). A *weighted TF problem description* is a tuple $\langle A, S, f, w, \alpha \rangle$ where $\langle A, S, f, \alpha \rangle$ is a TF problem description and $w : 2^S \mapsto [0, 1]$ is a skill weight function such that $w(S) = 1$ and w is monotone, i.e., $\forall S_1, S_2 \subseteq S$, $w(S_1) \leq w(S_1 \cup S_2)$.

For every $s_j \in S$, $w(\{s_j\})$ is simply denoted by $w(s_j)$. A natural way to define w is to consider a *normalized weighted sum function* w_Σ, satisfying $w_\Sigma(S') = \sum_{s_j \in S'} w_\Sigma(s_j)$ for each $S' \subseteq S$, and $\sum_{s_j \in S} w_\Sigma(s_j) = 1$. Accordingly, it satisfies the conditions from Def. 4.1 and we used it in all benchmarks presented in Sec. 6.

The *coverage* of a team T, denoted by $cov(T)$ is defined as:

$$cov(T) = w(\alpha(T)).$$

The *k-partial coverage* of a team, denoted by $pc(T, k)$, is defined as:

$$pc(T, k) = \min_{T' \subseteq T, |T'| \leq k} cov(T \setminus T').$$

We are ready to define formally the notion of partially robust team:

Definition 4.2 (Partially Robust Team). Given a weighted TF problem description $\langle A, S, f, w, \alpha \rangle$, $k \in \mathbb{N}$ and a rational number $t \in [0, 1]$, a team T is said to be $\langle k, t \rangle$-*partially robust* if T is efficient and $pc(T, k) \geq t$.

So a team is $\langle k, t \rangle$-partially robust if whenever k agents are removed from it, the coverage of the residual team is not lower than t. For instance, if one wants to guarantee that 95% of the weighted sum of all skills is covered after a loss of k agents, one simply considers a normalized weighted sum function $w = w_\Sigma$ and sets $t = 0.95$. Whereas k-robustness only essentially reports a binary value (either the team is k-robust or it is not), partial robustness provides in a sense more information regarding the team's robustness. Noteworthy, partial robustness generalizes robustness since a team is k-robust if and only if it is $\langle k, 1 \rangle$-partially robust.

The decision problem for partial robustness (labeled DP-PR-TF) asks, given a TF problem description, non-negative integers c, k, and rational number $t \in [0, 1]$, if there exists a $\langle k, t \rangle$-partially robust team $T \subseteq A$ such that $f(T) \leq c$. We show below that the computational complexity of this problem lies "in-between" the robustness and recoverability counterparts:

Proposition 4.3. DP-PR-TF *is* $\Sigma_2^{\mathbf{P}}$*-complete.*

Optimality is defined similarly to the standard and robust TF cases: T is an *optimal* $\langle k, t \rangle$-partially robust team if T is $\langle k, t \rangle$-partially robust and $f(T)$ is minimal.

Example 4.4 (continued). Table 1 summarizes the deployment cost, recovery cost, overall cost and coverage of the teams $T_1, \ldots, T_4$, which respectively correspond to plans I, ..., IV described in the introductive example[2]. The team T_1 is an optimal efficient team: it has the lowest deployment cost $f(T_1)$ among all possible efficient teams. Likewise, the team T_3 is an optimal 2-robust team and T_4 is an optimal $\langle 2, .9 \rangle$-partially robust team. The team T_2 is an optimal 2-recoverable team: it has the lowest overall cost $f(T_2) + rc(T_2, 2)$ (note again that the criterion of optimality in RecTF slightly differs from the other notions since it also considers the recovery cost.)

The optimal $\langle 2, .9 \rangle$-partially robust team T_4 has the following interesting features compared to the other teams: (i) by definition it provides a 2-partial coverage of 0.9, which is much higher than T_1 and T_2; (ii) while covering 90% of the weighted sum of skills if 2 agents are lost in the worst case, its deployment cost is only $690/970 = 71\%$ of the one of the optimal 2-robust team T_3; and (iii) its overall cost $f(T_4) + rc(T_4, 2)$ remains cheaper than the deployment cost of T_3 ($f(T_3)$).

[2]The precise formalization in terms of weighted TF problem description is rather straightforward, it is not provided here to avoid the introduction of heavy notations.

	T_1 (plan I) $\{C, FJ\}$	T_2 (plan II) $\{C, F, J\}$	T_3 (plan III) $\{C, CJ, CF, JF, JF\}$	T_4 (plan IV) $\{CJ, CJ, CJ, F\}$
$f(T_i)$	330	350	970	690
$rc(T_i, 2)$	330	230	0	150
$f(T_i) + rc(T_i, 2)$	660	580	970	840
$pc(T_i, 2)$	0	0.1	1	0.9

Table 1: Comparison in terms of deployment cost, recovery cost, overall cost and coverage of the teams $T_1, \ldots, T_4$ corresponding to plans I, ..., IV in the introductive example.

5 ALGORITHM

This section provides a procedure to compute an optimal $\langle k, t \rangle$-partially robust team, given a weighted TF problem description $\langle A, S, f, w, \alpha \rangle$, a non-negative integer k and a rational number $t \in [0, 1]$. Our approach is similar to Counter-Example-Guided Abstraction Refinement (CEGAR) [7, 15, 16], one of the most successful approaches for QBF (Quantified Boolean Formulae) solving. Indeed, DP-PR-TF is in $\Sigma_2^{\mathbf{P}}$ (cf. Prop. 4.3), and so it is similar in structure to a 2QBF problem $\exists X \forall Y \varphi$, where $Var(\varphi) = X \cup Y$ [18]. Such problems have a natural interpretation as a two person game between an "existential" player and a "universal" player [12]: the existential player assigns values to the variables in X and the universal player assigns values to the variables in Y. The goal of the existential player is to find out a valuation of X that cannot be refuted by the universal player. More precisely, the 2QBF is valid if and only if there is a valuation ω_X of X (existential player) and there is no valuation ω_Y of Y (universal player) such that $\omega_X \cup \omega_Y$ makes φ false. In a CEGAR-based algorithm, computing a solution is done by iteratively searching for a solution of an "abstracted", simplified problem. If such an "abstract" solution is found, one needs to check whether it is an actual solution of the original problem by searching for a counter-example. If no counter-example is found to the abstract solution, then it is a solution to the original problem. Otherwise, the abstract solution is blocked by taking advantage of the counter-example to refine the abstacted problem. Observe that an abstract solution is in fact a "candidate" solution to the original problem (existential player), and checking that it is an actual solution to the original problem is made by the universal player.

Our method for finding an optimal $\langle k, t \rangle$-partially robust team is similar in essence. It iterates over the set of efficient teams (the "abstract", candidate solutions chosen by an existential player) in an increasing deployment cost order. For each such candidate team, one tries to "break" it by removing k agents from it so that the coverage of the residual team is strictly lower than t. Instead of exploring one by one the set of efficient teams, we also propose a refinement process ("improved cut") that exploits the counter-example to filter a set of spurious teams at once. The process is iterated until a candidate team that cannot be broken is found.

5.1 Main Procedure

The outline of our algorithm is given in Algorithm 1. Let us first explain the core of the procedure (the details of the procedures *initConstraints* in line 2 and *generateConstraint* in line 8 will be explained later in this section.) Initially, one computes an efficient team T_{cur} of minimal cost, i.e., an optimal efficient team (cf. procedure *solveTF* in line 3). In line 5, the procedure *breakTeam* searches

for a set $T' \subseteq T_{cur}$ such that $|T'| \leq k$ and $cov(T_{cur} \setminus T') < t$, that is, it seeks to remove k agents from T_{cur} so that the coverage degree of the residual team is less than the input threshold t. If such a set T' does not exist, it means that T_{cur} is an optimal $\langle k, t\rangle$-partially robust team and the algorithm returns it (line 7). Otherwise, T' serves as a certificate that T_{cur} is not $\langle k, t\rangle$-partially robust. In line 8, a cut is generated so that T_{cur} is removed from the list of candidate teams to be returned. The procedure then searches for another efficient team of minimal cost while excluding the previously computed team T_{cur}. The same process is iterated until one of the following conditions occurs: the procedure *breakTeam* returns *null*, which means that the team T_{cur} computed at the corresponding iteration is proven to be $\langle k, t\rangle$-partially robust and returned in line 7; or the procedure *solveTF* in line 9 returns *null*, which means that there is no $\langle k, t\rangle$-partially robust team, and *null* is returned in line 10.

Algorithm 1: Computing an Optimal Partially Robust Team

input: A weighted TF problem description $\langle A, S, f, w, \alpha\rangle$, a non-negative integer k, a rational number t
ouput: An optimal $\langle k, t\rangle$-partially robust team

1 **begin**
2 $C \longleftarrow initConstraints(\langle A, S, f, w, \alpha\rangle, k, t)$
3 $T_{cur} \longleftarrow solveTF(C)$
4 **while** $T_{cur} \neq null$ **do**
// Search $T' \subseteq T_{cur}$ s.t. $|T'| \leq k$, $cov(T_{cur} \setminus T') < t$
5 $T' \longleftarrow breakTeam(T_{cur}, k, t)$
6 **if** $T' = null$ **then**
// T_{cur} is an optimal $\langle k, t\rangle$-partially robust team
7 **return** T_{cur}
// T_{cur} is not $\langle k, t\rangle$-partially robust
8 $C \longleftarrow C \cup generateConstraint(T_{cur})$
9 $T_{cur} \longleftarrow solveTF(C)$
// There is no $\langle k, t\rangle$-partially robust team
10 **return** *null*

Let us now explain in more details the procedures *initConstraints*, *solveTF*, *breakTeam*, and *generateConstraint* involved in the algorithm. Our model considers a set X of n binary variables $X = x_1, \ldots, x_n$, n being the total number of agents in A. An assignment of values to the variables from X corresponds to a team T where $a_i \in T$ if and only if $x_i = 1$.

initConstraints$(\langle A, S, f, w, \alpha\rangle, k, t)$. This procedure initializes a set of linear contraints C on X characterized by:

$$\forall s_j \in S \quad \sum_{x_i \in X, s_j \in \alpha(a_i)} x_i \geq 1 \tag{1}$$

This set of constraints precisely encodes the conditions of team efficiency, i.e., an assignment of X satisfies the set C if and only if it corresponds to an efficient team.

solveTF(C). This procedure is called in lines 3 and 9 and simply searches for an assignment of X that minimizes the cost of the corresponding team under the set of constraints C:

$$minimize \sum_{x_i \in X} f(a_i) \cdot x_i \tag{2}$$

breakTeam(T_{cur}, k, t). This procedure searches for a set $T' \subseteq T_{cur}$, $|T'| \leq k$, such that $cov(T_{cur} \setminus T') < t$. This is done by generating a new model on a set X' of $|T_{cur}|$ binary variables $X' = \{x'_i \mid a_i \in T_{cur}\}$, so that an assignment of values to the variables from X' represents a subset of agents $T' \subseteq T_{cur}$ to be removed from T_{cur}, i.e., $a_i \in T'$ if and only if $x'_i = 1$. The procedure finds an assignment of X' satisfying a set of constraints characterized by the following pair of equations:

$$\sum_{x'_i \in X'} x'_i \leq k \tag{3}$$

$$w\Big(\bigcup_{x'_i \in X', x'_i = 0} \alpha(a_i)\Big) < t \tag{4}$$

Equation 3 requires that $|T'| \leq k$, and Equation 4 requires that $cov(T_{cur} \setminus T') < t$. So accordingly, *breakTeam*(T_{cur}, k, t) returns such a subset T' if it exists, otherwise it returns *null*.

generateConstraint(T_{cur}). This procedure simply forbids the previously computed team T_{cur} to be selected again at the next search iteration. It generates the following constraint which states that a candidate team must contain at least one agent that does not belong to T_{cur}:

$$\sum_{x_i \in X, a_i \in A \setminus T_{cur}} x_i \geq 1 \tag{5}$$

To summarize, Algorithm 1 explores all efficient teams by increasing deployment cost order, checking for each one of them whether it is $\langle k, t\rangle$-partially robust, and returns it as soon as one such team is found. Accordingly, it returns an optimal $\langle k, t\rangle$-partially robust team, if it exists.

5.2 Cut Generation

We now present a cut, i.e., an improvement of the procedure *generateConstraint*(T_{cur}) to prune from the search space a set of assignments corresponding to teams that are guaranteed not to be $\langle k, t\rangle$-partially robust. Before presenting the cut, let us introduce a useful preliminary result:

Proposition 5.1. *Given a weighted TF problem description $\langle A, S, f, w, \alpha\rangle$, $k \in \mathbb{N}$ and a rational number t, a team $T \subseteq A$ is $\langle k, t\rangle$-partially robust if and only it is efficient and for each $S' \subseteq S$ such that $w(S \setminus S') < t$, we have that $|\{a_i \in T \mid \alpha(a_i) \cap S' \neq \emptyset\}| \geq k + 1$.*

Prop. 5.1 above states that a necessary and sufficient condition for any team T to be $\langle k, t\rangle$-partially robust team is that for any subset of skills S' such that the weight of $S \setminus S'$ is strictly below t, T must contain at least $k + 1$ agents who possess at least one of the skills from S'. Our improved cut consists in generating a constraint that excludes teams which do not satisfy this condition. To this end, one replaces the procedure *generateConstraint*(T_{cur}) (cf. Equation 5) in line 8 of the algorithm by a new procedure *generateConstraint*(T', k, t) which generates a constraint characterized by the following equation:

$$\sum_{x_i \in X, \alpha(a_i) \cap \alpha(T') \neq \emptyset} x_i \geq k + 1 \tag{6}$$

Doing so, Algorithm 1 prunes a number of teams that are not $\langle k, t\rangle$-partially robust (which also includes the team T_{cur} last computed), and returns an optimal $\langle k, t\rangle$-partially robust team accordingly, if it exists. The idea is similar to the cut considered in [8] to compute a recoverable team, which consists in excluding those

that do not include at least one agent from a certain set of skills without which the team cannot be recovered (or otherwise would be suboptimal).

In the next section, this improved cut will be shown to significantly reduce the runtime of the whole procedure.

6 EMPIRICAL STUDY

Our algorithm to find an optimal partially robust team was implemented, tested and compared with the other solution concepts, namely efficiency, robustness and recoverability, on various sets of instances. The existing solution concepts were implemented following the same encodings and algorithms as described in [8]. To evaluate the impact of the parameter t on the computed $\langle k, t\rangle$-partially robust teams, we set $t \in \{0.99, 0.95, 0.90\}$.

6.1 Benchmarks

First, we have tested all sets of instances (40 instances in total) that have been considered in [8]: (i) one set of ten small instances (30 agents / 11 skills) which correspond to the robust TF instances given in [20]; (ii) two sets of ten medium instances (100 agents / 20 skills and 150 agents / 30 skills) which are set covering instances used in [3, 28]; (iii) one set of ten large instances (1000 agents / 200 skills) which are classical instances found in the *OR Library* [2] under the *scp4x* package, used in set covering works (e.g., [10, 17, 19]). For all these instances, we set $w(s_j) = 1/|S|$ for each skill, and used a normalized weighted sum function $w = w_\Sigma$ (cf. 4), i.e., every $S' \subseteq S$, $w(S') = \sum_{s_j \in S'} w(s_j)$.

In addition, we have artificially generated a set of 100 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 in a certain area corresponds to a weighted skill (the weight depends on the density of the population at that specific 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 [23], 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 in the grid. A low (resp., mid, high) value is interpreted as a water cell (resp., a land cell, a mountain cell).

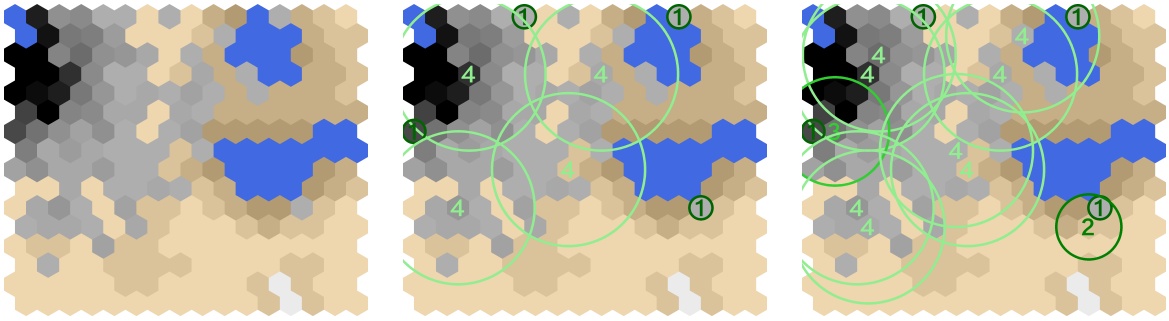
Second, the map is populated by iteratively adding an individual on the grid. Initially, three individuals are added in three different land cells randomly chosen, provided that the cell is next to a water. Then, a new individual is added at random following a probability distribution. The closer to an already populated cell, the higher its probability to welcome a new individual. The water cells and the cells that already host 10 individuals cannot host 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. brown, white) cells are of water type (resp. land, mountain 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 than 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 in our experiments, that is respectively efficient (Fig. 1(b)), 1-robust (Fig. 1(c)), $\langle 1, .99\rangle$-partially robust (Fig. 1(d)), $\langle 1, .95\rangle$-partially robust (Fig. 1(e)), and $\langle 1, .90\rangle$-partially robust (Fig. 1(f)). 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.

6.2 Empirical Results

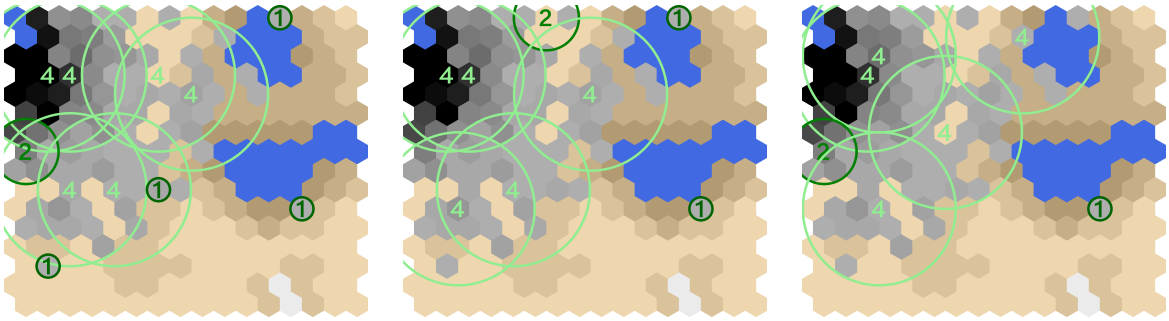
Let us start with a short analysis of the results in the map instance example provided in Fig. 1. Similarly to our introductive example, one can see on these figures the advantages of forming partially robust teams instead of efficient or (fully) robust teams. For instance, the optimal $\langle 1, .99\rangle$-robust team has only a deployment cost of 30 and a recovery cost of 4, whereas a 1-robust team has a deployment cost of 41: thus allowing 1% of coverage loss in the worst case during a disaster phase of the same scale (i.e., $k = 1$ in both cases) leads 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 ($f(T_2) = 41$, $rc(T_2, 1) = 0$, $pc(T_2, 1) = 1$).



(d) An optimal $\langle 1, .99\rangle$-partially robust team T_3 ($f(T_3) = 30$, $rc(T_3, 1) = 4$, $pc(T_3, 1) = .99$).

(e) An optimal $\langle 1, .95\rangle$-partially robust team T_4 ($f(T_4) = 24$, $rc(T_4, 1) = 10$, $pc(T_4, 1) = .95$).

(f) An optimal $\langle 1, .90\rangle$-partially robust team T_5 ($f(T_5) = 23$, $rc(T_5, 1) = 8$, $pc(T_5, 1) = .90$).

Figure 1: Optimal teams for different solution concepts (efficiency, robustness and partial robustness).

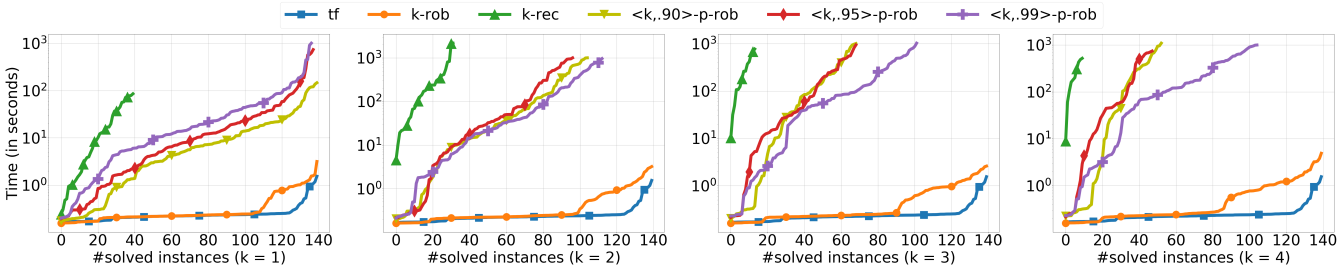


Figure 2: Time results on all instances for $k \in \{1, 2, 3, 4\}$.

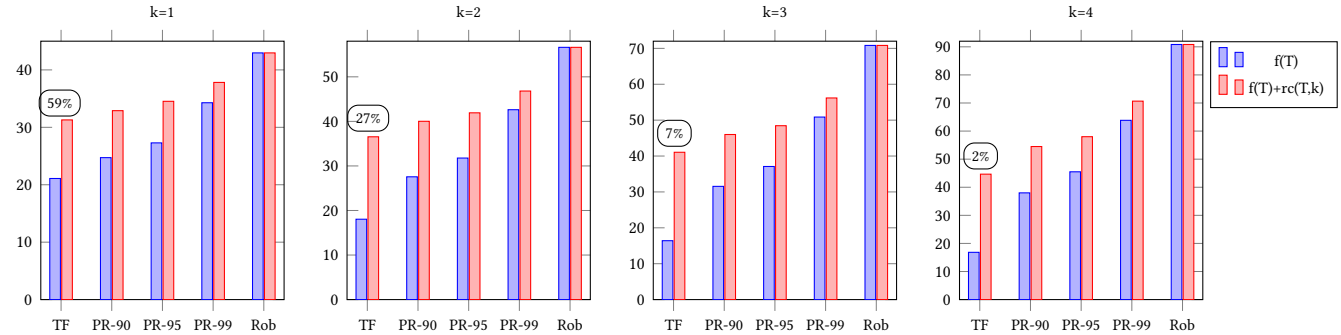


Figure 3: Deployment and overall cost in average for the generated facility location instances (maps). The percentage above each TF bar corresponds to the k-partial coverage value $pc(T)$ in average.

Type	#total	k	#solved					
			TF	RobTF	RecTF	PR-TF (w/o-w/ improved cut)		
						t = .90	t = .95	t = .99
small	10	1	⋆	⋆	⋆			
		2				8-⋆	2-⋆	2-⋆
		3				4-⋆	0-⋆	0-⋆
		4				0-⋆	0-⋆	0-⋆
medium	20	1				19-⋆	0-⋆	0-⋆
		2				0-⋆	0-⋆	0-⋆
		3			4	0-⋆	0-⋆	0-⋆
		4			0	0-⋆	0-⋆	0-⋆
large	10	1			⋆	⋆-⋆	⋆-⋆	0-⋆
		2			2	⋆-⋆	⋆-⋆	0-0
		3			0	0-0	0-9	0-0
		4			0	0-0	0-0	0-0
maps	100	1			0	0-⋆	0-98	0-97
		2			0	0-65	0-57	0-83
		3			0	0-29	0-30	0-72
		4			0	0-13	0-18	0-75

Table 2: Number of solved instances with a time out of 3600 seconds, for each set of instances and each solution concept. The symbol ⋆ means that all instances were solved. When two values p-q are given, p denotes the results without the improved cut (Sec 5.2), and q with the improved cut.

deployed solution to be $1-30/41 = 27\%$ cheaper in the initial phase, and $1-34/41 = 17\%$ cheaper overall (in the initial and recovery phases). On the other hand, the 1-partial coverage of the optimal efficient team (Fig. 1(b)) is only 20%, since a single, large facility is deployed to provide the necessary service for a densely populated area; its overall cost is, however, comparable to the overall cost of all three depicted partially robust teams.

In our experiments, the version of CPLEX used was IBM(R) ILOG(R) CPLEX(R) Interactive Optimizer 12.10 with the option set parallel 1. All experimentations have been conducted on Intel Xeon E52643 (3.30GHz) processors with 64Gb memory on Linux CentOS. Time-out was set to 3600 seconds for each run of the algorithm and for each instance; memory-out was set to 32 Gb for each such run.

Table 2 shows the number of instances solved within the time limit of 3600 seconds, for each method. It can be seen that computing an optimal $\langle k, t\rangle$-partially robust team is proved to be much more efficient than computing an optimal k-recoverable team: for example, no optimal k-recoverable team could be computed over all instances corresponding to our facility location problems (maps), whereas an optimal partially robust team could be found for a reasonable proportion of these instances. Table 2 also shows the comparison between two implementations of our algorithm: one with and one without the improved cut presented in the previous section. This shows that our improved cut plays a crucial role in the efficiency of the algorithm, as almost no instance was solved without exploiting it.

We also measured the time in seconds required for each notion (efficiency, robustness, recoverability and partial robustness) to compute optimal teams. Fig. 2 shows four cactus plots for different values of k in $\{1, 2, 3, 4\}$. Each plot gives for each notion[3] the number of instances solved in a given amount of time. As to $\langle k, t\rangle$-partial robustness, we see a slight change of behavior depending on the value of k. For higher values of k, the amount of skill coverage loss has a greater impact on the performance of our improved cut: the closer t is to 1, the more teams the improved cut prunes. Indeed, when the cost of an optimal $\langle k, t\rangle$-partially robust team is close to the cost of an optimal efficient team, then our algorithm only performs a smaller number of iterations. It is actually easy to see that if $t = 1$, one only needs two iterations to reach optimality: the constraints added by Equation 6 ensure that each skill is covered by at least $k + 1$ agents, i.e., that the team is 1-robust.

Lastly, Fig. 3 shows the deployment and overall cost in average for the generated facility location instances (maps). The percentage above each TF bar corresponds to the k-partial coverage value $pc(T, k)$ in average. It shows that the comparative behavior between optimal efficient teams, robust teams and partially robust teams described previously in Fig. 1 extends to the set of all facility location instances in average. On the one hand, the overall costs of $\langle k, t\rangle$-partially robust teams remain below the deployment of k-robust teams, even for high values of t. On the other hand, these costs remain arguably reasonable in comparaison with optimal efficient teams, while efficiency can only guarantee a very low k-partial coverage, e.g., below 10% in average for $k \geq 3$.

7 CONCLUSION

Our contribution is manifold: (i) we introduced the notion of partially robust team formation (PR-TF); (ii) we analyzed the computational complexity of PR-TF. The corresponding decision problem was shown to be $\Sigma_2^{\mathbf{P}}$-complete, which turns out to be "in-between" the robustness and recoverability counterparts in the polynomial hierarchy; (iii) we provided a complete algorithm for PR-TF *à la* CEGAR [7, 15, 16]; (iv) we implemented our algorithm and compared its performance with the existing complete algorithms for robust and recoverable TF on a number of existing and new benchmarks.s Our findings were that in practice, PR-TF is shown to be an interesting trade-off notion between (full) robustness and recoverability in terms of initial deployment cost, skill coverage in the disaster phase of a resilience scenario, and recovery cost in the third phase. In addition, PR-TF is much more efficient than recoverable TF, and thus more suited to be used for reasonably sized instances.

A first perspective is to seek for more efficient algorithms for PR-TF. One way would be to consider an anytime algorithm, computing sub-optimal teams and iteratively searching for cheaper ones while still guaranteeing their $\langle k, t\rangle$-partial robustness.

The facility location benchmarks we introduced in this paper can also be seen as a contribution by itself. Such benchmarks can be used to compare different SAT solvers for standard and robust TF, both **NP**-complete problems. In perspective, we plan to develop a software publicly available allowing for the generation of benchmarks of larger size. This can be done by tuning parameters related to the generation of the initial matrix based on Perlin noise [23], and by refining the granularity of the maps to be generated. Another perspective for benchmark generation will be to take advantage of real-world populated maps [9] and translate them into facility location problem TF instances.

ACKNOWLEDGMENTS

This work was partly supported by JSPS Kakenhi Grant Number JP20K11947 and JSPS Kakenhi Grant Number JP17H00763.

[3] tf, k-rob, k-rec and <k,t>-p-rob respectively stand for the notions of efficiency, robustness, recoverability and partial robustness in the figure.

REFERENCES

[1] Amir Ahmadi-Javid, Pardis Seyedi, and Siddhartha S. Syam. 2017. A Survey of Healthcare Facility Location. *Computers & Operations Research* 79 (2017), 223–263.

[2] John E. Beasley. 1990. OR-Library: Distributing Test Problems by Electronic Mail. *Journal of the Operational Research Society* 41, 11 (1990), 1069–1072.

[3] David Bergman and André Augusto Ciré. 2016. Multiobjective Optimization by Decision Diagrams. In *Proc. of the 22nd International Conference on Principles and Practice of Constraint Programming (CP'16)*. 86–95.

[4] Miquel Bofill, Dídac Busquets, Víctor Muñoz, and Mateu Villaret. 2013. Reformulation Based MaxSAT Robustness. *Constraints* 18, 2 (2013), 202–235.

[5] Michel Bruneau, Stephanie E. Chang, Ronald T. Eguchi, George C. Lee, Thomas D. O'Rourke, Andrei M. Reinhorn, Masanobu Shinozuka, Kathleen Tierney, William A. Wallace, and Detlof von Winterfeldt. 2003. A Framework to Quantitatively Assess and Enhance the Seismic Resilience of Communities. *Earthquake Spectra* 19, 4, 733–752.

[6] Danuta Sorina Chisca, Michele Lombardi, Michela Milano, and Barry O'Sullivan. 2019. Logic-Based Benders Decomposition for Super Solutions: An Application to the Kidney Exchange Problem. In *Proc. of the 25th International Conference on Principles and Practice of Constraint Programming (CP'19)*. 108–125.

[7] Edmund M. Clarke, Orna Grumberg, Somesh Jha, Yuan Lu, and Helmut Veith. 2000. Counterexample-Guided Abstraction Refinement. In *Proc. of the 12th International Conference on Computer Aided Verification (CAV'00)*. 154–169.

[8] Emir Demirović, Nicolas Schwind, Tenda Okimoto, and Katsumi Inoue. 2018. Recoverable Team Formation: Building Teams Resilient to Change. In *Proc. of the 17th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'18)*. 1362–1370.

[9] Pierre Deville, Catherine Linard, Samuel Martin, Marius Gilbert, Forrest R. Stevens, Andrea E. Gaughan, Vincent D. Blondel, and Andrew J. Tatem. 2014. Dynamic Population Mapping Using Mobile Phone Data. *Proc. of the National Academy of Sciences* 111, 45 (2014), 15888–15893.

[10] Chao Gao, Xin Yao, Thomas Weise, and Jinlong Li. 2015. An Efficient Local Search Heuristic with Row Weighting for the Unicost Set Covering Problem. *European Journal of Operational Research* 246, 3 (2015), 750–761.

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

[12] Alexandra Goultiaeva, Allen Van Gelder, and Fahiem Bacchus. 2011. A Uniform Approach for Generating Proofs and Strategies for Both True and False QBF Formulas. In *Proc. of the 22nd International Joint Conference on Artificial Intelligence (IJCAI'11)*. 546–553.

[13] Emmanuel Hebrard, Brahim Hnich, and Toby Walsh. 2004. Super Solutions in Constraint Programming. In *Proc. of the 1st International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR'04)*. 157–172.

[14] John N. Hooker and Greger Ottosson. 2003. Logic-Based Benders Decomposition. *Mathematical Programming* 96, 1 (2003), 33–60.

[15] Mikolás Janota, William Klieber, João Marques-Silva, and Edmund M. Clarke. 2016. Solving QBF with Counterexample Guided Refinement. *Artificial Intelligence* 234 (2016), 1–25.

[16] Mikolás Janota and João P. Marques Silva. 2011. Abstraction-Based Algorithm for 2QBF. In *Proc. of the 14th International Conference on Theory and Applications of Satisfiability Testing (SAT'11)*. 230–244.

[17] Serdar Kadioglu and Meinolf Sellmann. 2009. Dialectic Search. In *Proc. of the 15th International Conference on Principles and Practice of Constraint Programming (CP'09)*. 486–500.

[18] Hans Kleine Büning and Uwe Bubeck. 2009. Theory of Quantified Boolean Formulas. In *Handbook of Satisfiability*, Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh (Eds.). Frontiers in Artificial Intelligence and Applications, Vol. 185. IOS Press, 735–760.

[19] Nysret Musliu. 2006. Local Search Algorithm for Unicost Set Covering Problem. In *Proc. of the 19th International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, Advances in Applied Artificial Intelligence (IEA/AIE'06)*. 302–311.

[20] Tenda Okimoto, Nicolas Schwind, Maxime Clement, Tony Ribeiro, Katsumi Inoue, and Pierre Marquis. 2015. How to Form a Task-Oriented Robust Team. In *Proc. of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'15)*. 395–403.

[21] Tenda Okimoto, Nicolas Schwind, Emir Demirović, Katsumi Inoue, and Pierre Marquis. 2018. Robust Coalition Structure Generation. In *Proc. of the 21st International Conference on Principles and Practice of Multi-Agent Systems (PRIMA'18)*. 140–157.

[22] Christos H. Papadimitriou. 1994. *Computational complexity*. Addison-Wesley, Reading, Massachusets.

[23] Ken Perlin. 1985. An Image Synthesizer. In *Proc. of the 12th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH'85)*. 287–296.

[24] Talal Rahwan, Tomasz P. Michalak, Michael Wooldridge, and Nicholas R. Jennings. 2015. Coalition Structure Generation: A Survey. *Artificial Intelligence* 229 (2015), 139–174.

[25] Nicolas Schwind, Morgan Magnin, Katsumi Inoue, Tenda Okimoto, Taisuke Sato, Kazuhiro Minami, and Hiroshi Maruyama. 2016. Formalization of Resilience for Constraint-Based Dynamic Systems. *Journal of Reliable Intelligent Environments* 2, 1 (2016), 17–35.

[26] Nicolas Schwind, Tenda Okimoto, Katsumi Inoue, Katsutoshi Hirayama, Jean-Marie Lagniez, and Pierre Marquis. 2018. Probabilistic Coalition Structure Generation. In *Proc. of the 16th International Conference on Principles of Knowledge Representation and Reasoning (KR'18)*. 663–664.

[27] James C. Spall. 2005. *Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control*. Vol. 65. John Wiley & Sons.

[28] Thomas R. Stidsen, Kim A. Andersen, and Bernd Dammann. 2014. A Branch and Bound Algorithm for a Class of Biobjective Mixed Integer Programs. *Management Science* 60, 4 (2014), 1009–1032.