Algorithms for partially robust team formation

Schwind, Nicolas; Demirović, Emir; Inoue, Katsumi; Lagniez, Jean Marie

DOI
10.1007/s10458-023-09608-7

Publication date
2023

Document Version
Final published version

Published in
Autonomous Agents and Multi-Agent Systems

Autonomous Agents and Multi-Agent Systems (2023) 37:22
https://doi.org/10.1007/s10458-023-09608-7

Algorithms for partially robust team formation

Nicolas Schwind[1] · **Emir Demirović**[2] · **Katsumi Inoue**[3,4] · **Jean-Marie Lagniez**[5]

Accepted: 29 March 2023

Abstract
In one of its simplest forms, Team Formation involves deploying the least expensive team of agents while covering a set of skills. While current algorithms are reasonably successful in computing the best teams, the resilience to change of such solutions remains an important concern: Once a team has been formed, some of the agents considered at start may be finally defective and some skills may become uncovered. Two recently introduced solution concepts deal with this issue proactively: 1) form a team which is robust to changes so that after some agent losses, all skills remain covered, and 2) 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 two complete algorithms for it. We compare the performance of our algorithms with the existing methods for robust and recoverable team formation on several existing and newly introduced benchmarks. Our empirical study demonstrates that partial robustness offers an interesting trade-off between (full) robustness and recoverability in terms of computational efficiency, skill coverage guaranteed after agent losses and repairability. This paper is an extended and revised version of as reported by (Schwind et al., Proceedings of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'21), pp. 1154–1162, 2021).

Keywords Team formation · Robustness · Partial robustness · Resilience · Facility location · Computational complexity · Anytime algorithm

✉ Nicolas Schwind
nicolas-schwind@aist.go.jp

[1] National Institute of Advanced Industrial Science and Technology, Tokyo, Japan

[2] Delft University of Technology, Delft, The Netherlands

[3] National Institute of Informatics, Tokyo, Japan

[4] The Graduate University for Advanced Studies, SOKENDAI, Tokyo, Japan

[5] CRIL-CNRS UMR8188, Université d'Artois, rue Jean Souvraz, 62307 Lens, France

Published online: 25 April 2023

1 Introduction

In this paper, we focus on the Team Formation problem (TF). One of its simplest and most abstract forms consists in forming a team of agents with minimum cost that meets a given set of requirements. The problem is equivalent to the set covering and hitting set problems [1]. 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 [1, 2].

In realistic settings, there may be uncertainty about agents, in particularly after deployment: agents previously deemed capable may become unable to perform their duties, e.g., due to illness. Thus it is important to consider resilience properties in TF, i.e., proactively seek to form an efficient team that is robust to unexpected changes.

Robustness [2] and recoverability [3] are complementary notions in TF, both with their own advantages and disadvantages. A team is said to be *k-robust* if it remains efficient even in the event that any k agents are removed from it [2]. Robust TF consists in finding an optimal k-robust team of minimal deployment cost. Interestingly, computing an optimal robust team is not harder than computing an optimal efficient team without considering robustness [2]. However, the deployment cost of a robust team may be prohibitively high, as providing such strong guarantees is only possible by introducing a high degree of redundancy, i.e., each skill must be covered by at least $k + 1$ agents.

An alternative to robustness is *recoverability*: a team is considered *k-recoverable* if after losing any k agents, it can be "easily repaired" by hiring new agents at low cost. Demirović et al. [3] empirically showed that the overall deployment cost is effectively lower than the initial deployment cost of a robust team, providing a trade-off: the cost of the team may be lower at the expense of allowing that the team may be dysfunctional for some period after disruption. However, the problem of computing a recoverable team is $\mathbf{\Sigma_3^P}$-hard [3], making it difficult, if not inapplicable, in practice. Moreover, recoverability does not provide coverage guarantee during the disaster phase. As a result, a large number of critical skills may become uncovered for a certain amount of time during which the system loses most of its functionality. Depending on the application, this may be highly undesirable.

Let us introduce an example illustrating the problem and notions.

Example 1 The organizers of a special exhibition have a budget of 900 to hire professional guides. The attendees are expected to be from China (50%), Japan (40%), and France (10%). To enhance attendee experience, the organizers plan to hire proficient Chinese, Japanese and French speakers so that the exhibits could be explained to the attendees in their native language. A number of guides (called agents in the following) are available for hire. Type 1 agents are monolingual. It costs 100 to hire a type 1 Chinese-speaking or Japanese-speaking agent, and 150 for a French-speaking agent. Let us denote by C, J, and F a type 1 Chinese-, Japanese-, and French-speaking agent, respectively. Type 2 agents are bilingual, and are denoted respectively by CJ, CF, and FJ. It costs 180 to hire CJ, and 230 for CF or FJ. Agents are paid in advance and should be contacted at least one day ahead of the event, otherwise the agents may only attend the afternoon session whilst charging the same price.

A minimum-cost alternative (Plan I) is is to hire the team $\{C, FJ\}$. By doing so, agent C may help Chinese speakers, whereas agent FJ may help French and Japanese speakers. This plan corresponds to an optimal team for the standard TF problem and costs 330, i.e., it only considers covering the required skills (language proficiency) while minimizing cost,

without considering robustness. In the unfortunate case where both of these agents fall ill on the day of the event, two new agents may be hired as an emergency at the cost of 330.

However, under the same circumstances a slightly cheaper solution exists (Plan II), which consists in initially hiring 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 could ask for help from an FJ agent. Plan II is slightly more expensive than Plan I (350 vs. 330), but the recovery cost is 230 instead of 330, making this alternative arguably preferable over 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 in the last minute, then the event would be without support for two languages the whole morning (no support for any language in the case of Plan I).

Assuming we would like to be *robust* to a loss of *two* agents, an alternative plan would be to form the team $\{C, CJ, CF, FJ, FJ\}$ (Plan III): since each language is spoken by at least three agents, losing any two agents would still result in a team that can provide support in all three languages. Plan III corresponds to an optimal 2-robust team. However, Plan III is quite expensive: it costs 970, which exceeds the initial 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 speak only French, 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) at the cost of 690. Losing two agents from the team would still guarantee the Chinese and Japanese languages are covered by the remaining team members, while losing agent 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 team 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 $\langle 2, 0.9\rangle$-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 $\mathbf{\Sigma_2^P}$-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 may be computed significantly 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 may be cheaper than the initial deployment cost of a "fully" robust team in the general case.

We empirically study the benefits of our novel partial robustness notion on a number of existing benchmarks used in [3]. We also introduce a new set of benchmarks related to the facility location problem. This problem consists in deploying a set of facilities (e.g., health centers, antennas, schools, shelters) on a populated map to maximize a certain population coverage while minimizing the overall deployment cost [4]. The notion of partial robustness is of particular importance for these type of problems. When facilities are interpreted

as agents and population as (weighted) skills, the goal is to guarantee that a high percentage of the population still can access to a fully functional facility after some of the facilities break down.

This article is an extended and revised version of [5], a paper in which we introduced the notion of partial robustness, reported the computational complexity of the corresponding decision problem, presented a complete algorithm (PR) and reported some empirical results. This present article extends the results reported in [5] in several aspects:

- In addition to our initial algorithm PR, we introduce another complete algorithm (PR_A) that is anytime. While algorithm PR needs termination to return a partially robust solution, algorithm PR_A starts with a sub-optimal partially robust solution (sub-optimal in terms of deployment cost) and improves it over time. Interestingly, algorithm PR_A empirically outperforms algorithm PR in terms of runtime to reach optimality on a large set of instances. Our empirical evaluation shows that we may obtain near-optimal solutions after a very few number of iterations. Despite the high *theoretical* complexity of achieving partial robustness, algorithm PR_A allows one to compute satisfactory partially robust solutions in a very short amount of time on instances of reasonable size. This is particularly interesting for large instances, for which algorithm PR does not compute any solution within 3600 s. Both algorithms are presented and detailed in Sect. 5.
- We introduce two new "cuts" (named *cut* and *cut+* in this paper) that allow one to learn from counter-examples and prune the search space, improving the efficiency of both algorithms PR and PR_A (cf. Sect. 5.3). These cuts improve over the single cut reported in our conference paper.
- We provide software to generate facility location instances discussed above, whose structure and size can be tuned by a number of parameters. This was used to generate four sets of benchmarks of different sizes, described briefly in Sect. 6 and in more detail in Appendix B. We made our instance generator, together with the implementation of our algorithms, publicly available [6].
- We have extended our empirical analysis beyond the conference version.This includes both the evaluation of algorithms PR and PR_A, the comparison of our different cuts, and the additional set of benchmarks. These results further show the advantages of our novel anytime algorithm PR_A.
- The proofs of our formal propositions are available in Appendix A.

2 Related work

Team formation (TF) has received an ever-growing interest from practitioners and researchers over the past two decades, e.g., see [7] for a recent overview on applications, computational complexity, and resolution methods on TF. In particular, notions of resilience and robustness in multiagent systems, both separately and in combination, have recently attracted attention in a variety of contexts, including the design of new solution concepts, their formalization in the underlying multiagent frameworks, and the computational issues that follow.

From the conceptual perspective, *robustness* is the ability to withstand adversarial conditions without negative impact on performance [8, 2]. *Recoverability*, on the other hand, is the capability to restore the functionality of the system after disturbance [9]. Both notions

have been separately formalized in TF and analyzed from the computational viewpoint [3, 2]: the decision problems related to robust TF and recoverable TF have been shown to be respectively **NP**-complete and $\mathbf{\Sigma_3^P}$-complete. It turns out that recoverability can be seen as a generalization of robustness in TF, as pointed out in [3]. Alternative solution concepts, although derived from robustness and resilience, have been recently considered in TF, including team diversity [10], a key concept for the formation of synergistic teams; or when skills are replaced by (complex) tasks [11], team stabilizability [12]. The notion of robustness has also been imported in some variants of coalition formation including Hedonic games [13] and multi-team formation [14, 15]. Uncertainty issues have also been considered in the redundant multi-agent task allocation problem [16] that involves uncertain agents-task assignment costs, and in cooperative games with other failure models [17–21]. These frameworks depart from TF in both their structure and the underlying goals, which involve notions of stability and fairness, e.g., when some underlying team utility or value is distributed among the agents.

Closely related to TF is the framework of Coalition Structure Generation (CSG) [22], in which a number of solution concepts considering agent failure have also been formalized in the recent years. In CSG, we are given a set of agents and a characteristic function that associates every subset of agents with a value representing its utility. The goal is to form a *coalition structure*, i.e., a partition of the given agents into a set of teams such that the sum of the utilities of all teams is maximized. This problem is known to be **NP**-hard in the general case [23] and many algorithms have been proposed for solving, both in the cases when the characteristic function is provided extensively (i.e., as a table with 2^n entries, n being the number of agents) [24–27], or when concise representation languages for characteristic functions are considered, such as marginal contribution networks (MC-nets) [28], synergy coalition groups (SCGs) [29], skilled-based representations [30], agent-type representations [31], and coalition resource games [32], among others. Robustness and stochastic notions have recently been investigated in CSG. Robustness in CSG was first introduced in [33] and is a notion similar to robust TF: a coalition structure is k-robust if its global performance is kept above a certain threshold value in any case where at most k agents are removed from it. It has been proved that the underlying decision problem for robust CSG is $\mathbf{\Sigma_2^P}$-complete in the general case [33], and that $\mathbf{\Sigma_2^P}$-harness still holds when the characteristic function is represented by means of a coalition resource game [34]. Forming coalition structures under uncertainty has also been considered in the so-called *probabilistic* CSG framework [35, 36], where the attendance of agents is of stochastic nature. In [36] the problem was shown to be $\mathbf{NP^{PP}}$-hard in the general case, but remains in **NP** for some natural classes of problems where the characteristic function is represented by means of an MC-net. CSG and TF share obvious similarities in that in both cases the goal is to form teams (in the case of TF, a single team) to accomplish a certain (set of) task(s). However, CSG and TF depart from each other in their core structure: in TF, a single team is formed out of a pool of agents and the selection of each agent involves a cost, whereas in CSG a set of agents is already preselected and must be split into different teams, each of which is implicitly associated with a certain task to perform. Indeed, while the standard CSG and TF decision problems can be viewed as equivalent from the theoretically computational viewpoint (they are both **NP**-complete), the same cannot be stated for their robust counterpart: robust TF is **NP**-complete [2], whereas robust CSG is $\mathbf{\Sigma_2^P}$-complete [33]. One of the direct consequence of this observation is that results in robust CSG cannot be directly adapted to TF.

Noteworthy, TF is equivalent to the well-known set cover problem (SCP) [37], which is intrinsically connected to the facility location problem (FLP) [38]. There is a vast literature

investigating extensions and variants of this problem (see [39] for a survey), a number of which deal with uncertainties [40–47]. In all of these works, different uncertainty aspects are considered such as the reliability of the service provided by a facility, or when reliability decays with the distance from a facility to a customer. All of these considerations are of stochastic nature, and the main goal is to find a input set / facilities that maximize some expected coverage. Thus, these notions depart from the robustness notions focused in this paper, that are based on worst-case scenarios given k agent losses. In [48] the authors consider a variant of the FLP called maximal covering location problem (MCLP). In the MCLP, the goal is to maximize the amount of demand covered within an acceptable service distance by locating a given fixed number of facilities. Similar to our considerations of skill coverage, in MCLP the key assumption is that coverage is binary: each customer is either fully covered if there is a facility within acceptable distance, otherwise it is not covered at all. Such an assumption has been released in subsequent works, leading to the alternative notion of partial covering problem [49–52]. These notions are intended to maximize a coverage degree in the initial deployment step, and thus do not account for potential damaging events after deployment. More closely related to our notion of partial robustness, are similar notions introduced for FLPs in [53, 54]. In [54], the authors introduced the r-interdiction covering problem, in which facilities and services can be lost due to natural or man-made disasters. The problem to be solved takes place from the viewpoint of the attacker: from some predefined facility deployment, the goal is to find a subset of r facilities, which when removed, maximizes the resulting drop in coverage. The notion can be viewed as some inverse problem of the standard FLP/SCP. Indeed, the problem still lies in **NP**. In [53] the defensive maximal covering problem is introduced. The notion is defined for a specific type of facility location problems where the coverage of facilities is induced by "weighted links", connecting the candidate facility nodes in a network. The authors then consider a leader-follower formulation of the problem: the leader wants to locate facilities to maximize demand coverage while the follower's goal is to disconnect the most damaging links once the facilities are deployed. While the notion bears similarity with our notion of partial robustness, in [53] the potential loss is not considered in the facilities but on the links between the candidate facility locations. Moreover, the underlying leader-follower model is formulated as a pair of independent problems, which are designed to be solved sequentially; this makes both problems in **NP** and thus departs from our work.

From the algorithmic viewpoint, our approach to compute optimal partial robust teams is similar to Counter-Example-Guided Abstraction Refinement (CEGAR) [55–57], one of the most successful approaches for QBF (Quantified Boolean Formulae) solving. Indeed, as will be shown in Sect. 4, the decision problem related to our notion of partial robustness is in $\mathbf{\Sigma_2^P}$ (cf. Prop. 4), so it is similar in structure to a 2QBF problem $\exists X \forall Y \varphi$, where $Var(\varphi) = X \cup Y$ [58]. Such problems have a natural interpretation as a two-person game between an "existential" player and a "universal" player [59]: the goal of the existential player is to find out a valuation of X that cannot be refuted by the universal player through a valuation of Y. In a CEGAR-based algorithm, computing a solution ω_X 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 certificate ω_Y showing that ω_X is a counter-example. If no such certificate ω_Y is found for the abstract solution ω_X, then ω_X is a solution of the original problem. Otherwise, ω_X is blocked by taking advantage of the certificate ω_Y to refine the abstracted problem. Our algorithms for computing optimal partially robust teams are similar in essence. Initially a simplified problem is considered, and the algorithm iteratively computes a new candidate team. If a certificate is found showing that the candidate team

is a counter-example, i.e., it is does not meet the partial robustness constraints, then a new constraint is generated to prune the search space, and the process iterates until an optimal partially robust team is found. In this context, a certificate consists in a removal of some agents of the candidate team resulting in a violation of the partial robustness constraints.

3 Preliminaries

This section recalls some preliminaries about basic notions of computational complexity, the Team Formation (TF) problem, and its extensions to Robust TF [2] and Recoverable TF [3].

3.1 Computational complexity

We assume that the reader is familiar with the complexity class **NP** (see [60] for more details). Higher complexity classes are defined using oracles. In particular, $\mathbf{\Sigma_2^P} = \mathbf{NP^{NP}}$ (resp. $\mathbf{\Sigma_3^P}$) corresponds to the class of decision problems that are solved in polynomial time by non-deterministic Turing machines using an oracle for **NP** (resp. $\mathbf{\Sigma_2^P}$).

3.2 Team formation

Let us formalize the (standard) TF problem [2].

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 $T \subseteq A$. It is possible to extend 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 as input a TF problem description $\langle A, S, f, \alpha \rangle$ where f and α are computed in polynomial time and a threshold $c \in \mathbb{N}$, whether there exists an efficient team T such that $f(T) \leq c$. This problem is equivalent to the well-known set cover problem [1]:

Proposition 1 ([2]) DP-TF *is* **NP**-*complete*.

3.3 Robust TF

Definition 2 (Robust Team [2]) 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 as input a TF problem description $\langle A, S, f, \alpha \rangle$ where f and α are computed in polynomial time and c, k in $\mathbb{N}$, if there exists a k-robust team $T \subseteq A$ such that $f(T) \leq c$. Then:

Proposition 2 ([2]) 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. 1):

Definition 3 (RecTF Problem Description [3]) 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.

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 ($h(a_i) = +\infty$). Similar 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 $(T \backslash T') \cup T_{rec}$ is efficient:

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

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 4 (Recoverable Team [3]) 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 as input a TF problem description $\langle A, S, f, \alpha, h \rangle$ where f, α and h are computed in polynomial time 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 ([3]) DP-RecTF *is* $\mathbf{\Sigma_3^P}$*-complete*.

The goal of RecTF is, given a fixed non-negative integer k, to compute an *optimal* k-recoverable team. As opposed to RobTF, optimality in RecTF can be defined in several ways. Indeed, after fixing the robustness parameter k, one is left with two functions to minimize for a team T: its deployment cost $f(T)$ and its k-recovery cost $rc(T, k)$, bounded by the input parameters c and r, respectively, in the decision problem DP-RecTF. For instance, in [3] the authors defined optimality as a single objective problem: an optimal k-recoverable team is therein defined as a team T that is $\langle k, r\rangle$-recoverable and whose *overall cost* $f(T) + rc(T, k)$ is minimal.

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 5 (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. 5 and we used it in all benchmarks presented in Sect. 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 6 (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 $pc(T, k) \geq t$.

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

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

	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

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 weighted TF problem description $\langle A, S, f, w, \alpha\rangle$ where f, w and α are computed in polynomial time, 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[1]:

Proposition 4 DP-PR-TF *is* $\mathbf{\Sigma_2^P}$*-complete*.

Optimality is defined similar to the efficient 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.

Before concluding this section, let us illustrate the notions introduced so far on our introductory example of organizing a special exhibition and hiring of professional guides (cf. Sect. 1):

Example (continued) Table 1 summarizes the deployment cost, recovery cost, overall cost and coverage of the teams $T_1, \ldots, T_4$, which correspond to plans I, ..., IV, respectively, 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 team 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 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 two 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 ($f(T_3)$).

[1] One does not need to make any assumption about the way f, w, and α are represented for our result to hold. However, one must assume that the corresponding mappings are computed in polynomial time.

[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.

5 Algorithms

This section provides two procedures 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 $t \in [0, 1]$. Both approaches follow a similar idea as Counter-Example-Guided Abstraction Refinement (CEGAR) [55–57], briefly described in the related work section (Sect. 2).

The first method, referred to as PR in the following text, iterates over the set of *optimal* efficient teams (the "abstract" candidate solutions chosen by an existential player). For each such candidate team T, one tries to "break" it by removing k agents from it so that the coverage of the residual team is strictly lower than t. If T cannot be broken, then it is $\langle k, t\rangle$ -partially robust. Otherwise, one generates a constraint that safely removes a set of teams from the search space (including T), and one proceeds to the next iteration step. Stated otherwise, all optimal efficient teams that are potentially $\langle k, t\rangle$-partially robust are computed and tested for partial robustness in an *increasing* deployment cost order. The process is iterated until a candidate team that cannot be broken is found. This first procedure provides a guarantee to find an optimal partially robust team in a finite amount of steps.

With our first method, no $\langle k, t\rangle$-partially robust team is provided before the procedure terminates. However, because of time limitations one may be interested in being given a sub-optimal $\langle k, t\rangle$-partially robust team in a short amount of time (a team with a non-minimal deployment cost), that improves over time. This motivates our second method, an anytime algorithm named PR_A. It first starts with a k-robust team, which can be computed in a single call to an **NP** oracle (cf. Proposition 2). Accordingly, a k-robust team is $\langle k, t\rangle$-partially robust for any value $t \in [0, 1]$, and thus can be considered as a sub-optimal output. The procedure then searches for a sub-optimal efficient team T, i.e., such that $f(T) < c$, where c is the deployment cost of the team computed at the previous step. It tries to "break" the team similarly to the first method, updates c when the break attempt fails, and in such a case proceeds to search for another efficient team with the updated threshold c. While PR_A is anytime, it also provides a guarantee to eventually find an optimal partially robust team.

5.1 Algorithm PR

The outline of our first method (PR) is given in Algorithm 1. Let us first explain the core of the algorithm (the details of the procedures *initConstraints* in line 2, *solveTF* in lines 3 and 9, *breakTeam* in line 5 and *generateConstraint* in line 8 are detailed 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 constraint is generated based on T_{cur} and T' to prune a set of non $\langle k, t\rangle$-partially robust teams

(including T_{cur}) from future consideration. The procedure then searches for another efficient team of minimal cost under the new set of constraints. 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 $\langle k, t\rangle$-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: PR

```
   input: A weighted TF problem description ⟨A, S, f, w, α⟩,
          a non-negative integer k,
          a rational number t ∈ [0, 1]
   ouput: An optimal ⟨k, t⟩-partially robust team
 1 begin
 2     C ⟵ initConstraints(⟨A, S, f, w, α⟩, k, t)
 3     T_cur ⟵ solveTF(C)
 4     while T_cur ≠ null do
           // Search T' ⊆ T_cur s.t. |T'| ≤ k, cov(T_cur \ T') < t
 5         T' ⟵ breakTeam(T_cur, k, t)
 6         if T' = null then
               // T_cur is an optimal ⟨k, t⟩-partially robust team
 7             return T_cur
           // T_cur is not ⟨k, t⟩-partially robust
 8         C ⟵ C ∪ generateConstraint(T_cur, T')
 9         T_cur ⟵ solveTF(C)
       // There is no ⟨k, t⟩-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:

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

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:

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

breakTeam$(T,\ k,\ t)$ This procedure searches for a set $T' \subseteq T$, $|T'| \leq k$, such that $cov(T \backslash T') < t$. This is done by generating a new model on a set X' of $|T|$ binary variables $X' = \{x'_i \mid a_i \in T\}$, so that an assignment of values to the variables from X' represents a subset of agents $T' \subseteq T$ to be removed from T, 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:

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

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

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

The procedures *solveTF* and *breakTeam* both involve solving optimization problems that are **NP**-hard in the general case, and thus rely on an underlying integer programming optimization solver. Notably, these procedures can be implemented using any state-of-the-art solver, the choice of which may only have an effect the computational efficiency of the whole procedure.

generateConstraint(T, T'). When this procedure is called, its input teams T, T' are such that T is efficient, $|T'| \leq k$ and $cov(T \setminus T') < t$. That is, T is an efficient team and T' can be viewed as a certificate proving that T is not $\langle k, t \rangle$-partially robust. The procedure aims to generate a constraint based on T and T' to discard a set of teams from the search space, such that the two following conditions are satisfied: (i) at least team T must be discarded; and (ii) all teams from the discarded set must be non $\langle k, t \rangle$-partially robust teams. Condition (i) ensures that Algorithm 1 terminates, and condition (ii) ensures that no potentially $\langle k, t \rangle$-partially robust team is discarded, so that the $\langle k, t \rangle$-partially robust team eventually returned by the algorithm is optimal. Such a constraint can be specified in a number of ways. We first present here its simplest alternative, which consists in discarding T only.

Doing so, one forbids the previously computed candidate team T to be selected again in any future search iteration. The corresponding constraint states that a candidate team must contain at least one agent that does not belong to T:

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

Since the constraint formalized in Eq. 5 discards T only, it does not take advantage of potentially additional information provided by the certificate T'. However, in the next section (Sect. 5.3), we will introduce two cuts, i.e., two alternatives that exploit team T' and improve computational efficiency.

To summarize, Algorithm 1 explores a set of 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 Algorithm PR_A

While PR (cf. Algorithm 1) has the guarantee to compute an optimal partially robust team after a finite number of steps, it does not provide any output before the procedure terminates. However, one may be interested in a procedure that provides one with a sub-optimal partially robust team first, and that iteratively searches for another "better" partially robust candidate over time, with a lower deployment cost. This motivates our second method PR_A, an anytime algorithm whose outline is given in Algorithm 2.

Algorithm 2: PR_A

input: A weighted TF problem description $\langle A, S, f, w, \alpha \rangle$,
a non-negative integer k,
a rational number $t \in [0, 1]$
ouput: An optimal $\langle k, t \rangle$-partially robust team

1 **begin**
2 $T_{cur} \longleftarrow A$
// Search $T' \subseteq T_{cur}$ s.t. $|T'| \leq k$, $cov(T_{cur} \setminus T') < t$
3 $T' \longleftarrow breakTeam(T_{cur}, k, t)$
4 **if** $T' \neq null$ **then**
// No $\langle k, t \rangle$-partially robust team exists
5 **return** *null*
6 $T_{new} \longleftarrow solveRobTF(\langle A, S, f, w, \alpha \rangle, k)$
7 **if** $T_{new} \notin \{T_{cur}, null\}$ **then**
// T_{new} is an improved $\langle k, t \rangle$-partially robust team ($f(T_{new}) < f(T_{cur})$)
8 $T_{cur} \longleftarrow T_{new}$
9 $c \longleftarrow f(T_{cur})$
10 $C \longleftarrow initConstraints(\langle A, S, f, w, \alpha \rangle, k, t)$
11 $T_{new} \longleftarrow solveTF\text{-}thr(C, c)$
12 **while** $T_{new} \neq null$ **do**
// Search $T' \subseteq T_{new}$ s.t. $|T'| \leq k$, $cov(T_{new} \setminus T') < t$
13 $T' \longleftarrow breakTeam(T_{new}, k, t)$
14 **if** $T' = null$ **then**
// T_{new} is an improved $\langle k, t \rangle$-partially robust team ($f(T_{new}) < f(T_{cur})$)
15 $c \longleftarrow f(T_{new})$
16 $T_{cur} \longleftarrow T_{new}$
17 $C \longleftarrow C \cup generateConstraint(T_{new}, T')$
18 $T_{new} \longleftarrow solveTF\text{-}thr(C, c)$
// T_{cur} is an optimal $\langle k, t \rangle$-partially robust team
19 **return** T_{cur}

This algorithm takes advantage of the procedures *initConstraints*, *breakTeam* and *generateConstraint* also used in Algorithm 1, and uses two new procedures *solveRobTF* and *solveTF-thr* whose details are given later in this section.

The initialization phase goes from lines 2 to 8 and is divided into two sub-steps. First, one checks if the team formed of all available agents ($T_{cur} = A$, line 2) is $\langle k, t \rangle$-partially robust (line 3). If not, then obviously enough no $\langle k, t \rangle$-partially robust team can exist, and *null* is returned in line 5. Otherwise, T_{cur} is a first $\langle k, t \rangle$-partially robust candidate that one seeks to improve in terms of deployment cost. Second, as another potential, better candidate, one computes an optimal k-robust team T_{new} (line 6). If such a team is found and is different from the initial team $T_{cur} = A$, this team is set to be the new improved candidate (line 8): since it is k-robust, it is necessarily $\langle k, t \rangle$-partially robust as well; and since $T_{new} \neq A$ (cf. line 7), this team is necessarily strictly improved compared to T_{cur}, i.e., $f(T_{new}) < f(T_{cur})$. Starting from a k-robust team instead of the set of all agents serves as a search space pruning: if a k-robust team exists, there is no need to search for teams with a higher deployment cost. This step is also "almost free" computationally speaking, since computing an optimal k-robust team is not harder than computing an optimal efficient team (cf. Proposition 2).

At this point, one proceeds to iteratively improve the current candidate T_{cur}, with the procedure *solveTF-thr* being now the main tool. Similar to Algorithm 1, one starts with a set of constraints C that encodes the conditions of team efficiency (line 10). One then

searches using *solveTF-thr* (line 11) for any efficient team T_{new} whose deployment cost is strictly lower than the one of the current candidate T_{cur}, i.e., such that $f(T_{new}) < c$ with $c = f(T_{cur})$. If no such team exists, then no partially $\langle k, t\rangle$-partially robust team can be found with deployment cost strictly lower than c, i.e., T_{cur} is an optimal $\langle k, t\rangle$-partially robust team and is returned in line 19. Otherwise, similar to line 3 one checks whether T_{new} is partially robust, by seeking to "break" it (line 13). If T_{new} is not found to be a counter-example, then it is a new $\langle k, t\rangle$-partially robust candidate with a deployment cost improved compared to the one of T_{cur}. In this case, T_{cur} and the new cost threshold are updated accordingly (lines 15 and 16). Otherwise, T' serves as a certificate that T_{new} is not $\langle k, t\rangle$-partially robust, and a constraint is generated based on T_{new} and T' (line 17). In both cases, T_{new} is removed from the search space and a new efficient candidate is sought for (line 18).

The procedures *solveRobTF* and *solveTF-thr* are characterized as follows:

solveRobTF$(\langle A, S, f, w, \alpha\rangle, k)$. This procedure is called in line 6 and computes an optimal k-robust team. It searches for an assignment of X that minimizes the cost of the corresponding team (cf. Eq. 2) under the following set of constraints:

$$\forall s_j \in S \sum_{x_i \in X, s_j \in \alpha(a_i)} x_i \geq k + 1 \tag{6}$$

That is, *solveRobTF* consists in calling the procedure *solveTF* used in Algorithm 1, but under a set of constraints that strengthen the ones given in Eq. 1. This set of constraints precisely encodes the conditions of team robustness, i.e., a team is k-robust if and only if each skill is covered by least $k + 1$ agents from the team [2].

solveTF-thr(C, c). This procedure is called in lines 11 and 18. It is a modified version of the procedure *solveTF*: instead of searching for an efficient team of minimal deployment cost, *solveTF-thr*(C, c) consists in restricting the search space (currently characterized by the set of constraints C) to teams whose deployment cost is strictly lower than a threshold c. The set of corresponding constraints is characterized by:

$$\sum_{x_i \in X} f(a_i) \cdot x_i < c \tag{7}$$

PR_A is anytime, since one is always given a partially robust team T_{cur} that is updated over time if another better candidate is found. It is also complete, since it is guaranteed to return an optimal team after a finite number of steps.

5.3 Cut generation

The constraint added by the procedure *generateConstraint*(T, T') used in both algorithms can be specified in various ways, as long as it safely discards from the search space a set of teams that are guaranteed not to be $\langle k, t\rangle$-partially robust, including T. In the previous section, we have introduced one of its simplest forms that simply removes T from the search space, and thus does not exploit any potentially useful information provided by the certificate T' (cf. Eq. 5). To fill the gap, we now present two cuts, i.e., two alternative implementations of this procedure that exploit T' to prune from the search space a larger set of spurious teams.

These cuts are based on the following useful result:

Proposition 5 *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* $T \subseteq A$ *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* $|\{a_i \in T \mid \alpha(a_i) \cap S' \neq \emptyset\}| \geq k + 1$
.

Proposition 5 above states that a necessary and sufficient condition for any team T to be $\langle k, t \rangle$-partially robust is that for any subset of skills S' such that the weight of $S \setminus S'$ is strictly lower than t, T must contain at least $k + 1$ agents that possess at least one of the skills from S'. Our cuts, simply named cut and cut+ in the following, are based on this result: they both consist in generating a (set of) constraint(s) that exclude(s) teams which do not satisfy this condition, based on the counter-example T' found in line 5 of Algorithm 1 and line 13 of Algorithm 3.

cut (first version). Taking advantage of a certificate T', one seeks for the *smallest* (in terms of cardinality) subset of skills $\mathit{filter}(\alpha(T'))$ (simply denoted by S') covered by T' such that $w(S \setminus S') < t$. To this end, one sorts the skills $\alpha(T')$ in a non-increasing order w.r.t. w, and from the resulting ordered list $(s_1, s_2, \ldots)$ with $w(s_i) \geq w(s_{i+1})$ for each i, one selects the sublist $S' = (s_1, \ldots, s_j)$ such that $w(S \setminus S') < t$ and $w((S \setminus S') \cup \{s_j\}) \geq t$. Obviously enough, the set S' is computed in polynomial time in the number of agents in T'.

This cut is implemented in the procedure *generateConstraint*(T, T') in line 8 of PR (Algorithm 1) and line 17 of PR_A (Algorithm 3), that computes S' as described above and generates a constraint characterized by the following equation:

$$\sum_{x_i \in X, \alpha(a_i) \cap S' \neq \emptyset} x_i \geq k + 1 \tag{8}$$

Doing so, one prunes only teams that are not $\langle k, t \rangle$-partially robust, which also includes the team T last computed. The idea is an improved version of the cut considered in [3] 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).

Notably, in Eq. 8 one could simply choose $\alpha(T')$ instead of its preprocessed subset of skills $S' = \mathit{filter}(\alpha(T'))$. However, the equation using S' generates a constraint strictly stronger than using $\alpha(T')$ (when $S' \subsetneq \alpha(T')$), resulting in an improved pruning of the search space.

cut+. We also considered a slightly modified version of the cut introduced above, by generating a *set of* constraints based on the counter-example T'. The idea is similar to the first version of the cut, i.e., by sorting the skills $\alpha(T')$ in a non-increasing order w.r.t. w. However, instead of focusing only on its smallest subset $S' = \mathit{filter}(\alpha(T')) \subseteq \alpha(T')$ one considers a set of such subsets $\mathcal{S} = \{S'_1, \ldots\}$, recursively computed as:

- $S'_1 = \mathit{filter}(\alpha(T'))$
- for each $S'_i \in \mathcal{S}$, $i > 1$, $S'_i = \mathit{filter}(\alpha(T') \setminus (S'_1 \cup \ldots \cup S'_{i-1}))$,

and such that for each $S'_i \in \mathcal{S}$, $w(S \setminus S'_i) < t$.

This modified cut is implemented in the procedure *generateConstraint*(T, T') that now computes $\mathcal{S}$ as described above and generates a set of constraints characterized by the following set of equations:

$$\forall S'_i \in \mathcal{S} \quad \sum_{x_i \in X, \alpha(a_i) \cap S'_i \neq \emptyset} x_i \geq k+1 \tag{9}$$

The choice of this modified cut is motivated by the observation that the constraints from the generated set all bear on disjoint set of skills S'_i. Indeed, this potentially increases the diversity of the set of generated contraints given a single certificate T', i.e., it intends to increase the pairwise symmetrical difference between the set of agents appearing in these constraints, thus pruning different areas of the search space at once.

In the empirical results section, these two cuts (cut and cut+) will be compared and shown to significantly reduce the runtime of the base procedures.

6 Benchmarks

In this section, we empirically compare our algorithm for finding optimal partially robust teams to other solution concepts, namely efficiency, robustness and recoverability, on a wide range of instances. This section also describes the generation protocol of these instances, which can be divided into two sets. The first set consists of existing instances found in previous works on TF and set covering [61, 62, 2, 63]. These instances were precisely the ones considered in [3] to compare the computational efficiency of TF, RobTF and RecTF. Since none of these instances considered weighted skills (the notion being relevant only to partially robust TF), we have artifically extended them to weighted TF problem descriptions (see Sect. 6.1 below). The second set consists of original, structured weighted TF instances generated following a protocol we introduce in Sect. 6.2.

6.1 Existing benchmarks (rtf30, cire100, cire150, scp1000)

First, we have considered all sets of instances (40 instances in total) that have been considered in [3]:

- *rtf30* a set of ten small instances (30 agents / 11 skills) which correspond to the robust TF instances given in [2];
- *cire100 / cire150* two sets of ten medium-sized instances (100 agents / 20 skills for the cire100 set and 150 agents / 30 skills for the cire150 set), that are set covering instances used in [62, 63];
- *scp1000* a set of ten large instances (1000 agents / 200 skills), that are classical instances found in the *OR Library* [61] under the *scp4x* package, used in set covering works (e.g., [64–66]).

All of these instances fully specify a TF problem description as in Definition 1, i.e., no weight was initially associated with the skills. Therefore, for all of these instances we set by default a weight of $w(s_j) = 1/|S|$ for each skill, and we used a normalized weighted sum function $w = w_\Sigma$ (cf. Sect. 4), i.e., every $S' \subseteq S$, $w(S') = \sum_{s_j \in S'} w(s_j)$.

These instances were randomly generated following different and simple protocols. For example, the protocol used to generate each rtf30 instance [2] consisted in associating with each agent $a_i \in A$ a random cost $f(a_i)$ within range [1000, 2000], and a random set of skills with set size randomly chosen within range [1, 10]. Similarly, the cire100 and cire150 instances consisted of agents with a random cost and a random set of skills. As to

Table 2 Description of instances for each type

Type	#instances	#agents	#skills	Sum of skills weights	max #skills per agent	avg #skills per agent
rtf30	10	30	11	11	10	4.63
cire100	10	100	20	20	8	2.00
cire150	10	150	30	30	10	2.00
scp1000	10	1000	200	200	12	3.98
map-r1	100	15.78	5.51	17	7	2.09
map-r2	100	100.62	19.88	65.94	16	3.99
map-r3	100	537.05	75.32	259.89	37	7.19
map-r4	100	1534.84	284.91	1030.97	61	12.54

In order: type of instance, number of instances, number of agents per instance on average, number of skills per instance on average, sum of skill weights ($w_\Sigma(A)$) per instance on average, maximum number of skills per agent per instance on average, average number of skills per agent per instance on average

the scp1000 instances, for each cost value c ranging over [1, 100], ten agents were associated with c, and a small randomly chosen set of skills was associated with each agent (four skills per agent in average). Table 2 summarizes some characteristics of these instances.

6.2 New benchmarks (map-r1, map-r2, map-r3, map-r4)

One of the drawbacks of the instances presented in the previous section is that they lack structure and may not reflect real-world situations. In particular, in all of these instances the cost of each agent has no link with the amount or rarity of skills it is associated with. As a consequence, these instances are not prevented from the presence of agents a_i dominated by some other agent a_j, i.e., such that $f(a_j) < f(a_i)$ and $\alpha(a_i) \subseteq \alpha(a_j)$. In addition, all skills have the same importance since we assigned by default the same weight value for each skill; doing otherwise, i.e., associating each skill with a random value, would not solve the issue about the lack of structure.

To fill the gap, we have developed a software that allows one to generate structured instances modeling facility deployment problem instances[3]. Our software is publicly available [6] and allows for the generation of various types of instances according to a set of parameters. The facility deployment 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 [4]. 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. In this context, finding an optimal efficient team corresponds to finding a facility deployment solution of minimal cost while providing a service coverage over the whole population: facilities correspond to agents and the population to be covered in a certain area corresponds to a weighted skill, where the weight depends on

[3] Publicly available benchmark generation protocols related to facility deployment can be found at http://old.math.nsc.ru/AP/benchmarks/english.html. However, to the best of our knowledge, none of these protocols includes the generation of skill weights. Adapting any of these protocols to fit our setting would be necessary, but this falls outside the scope of this paper.

the density of the population at that specific location. Indeed, this type of problems is perfectly suitable from the partial robustness analysis viewpoint: when a number of cell towers suddenly becomes unfunctional, one wants to ensure that a certain high percentage (if not 100%) of the population is still provided with an access to the network before recovery.

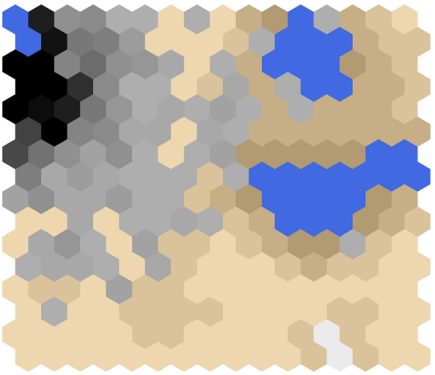
Each populated map was synthesized according to a number of parameters, e.g., size, variation in terms of elevation, and total population. We only provide below a rough description of the generation protocol, but the detailed list of the parameters and their role are available in Appendix B.

First, one generates an elevation map made of water parts, lands and mountains. A grid of numbers is created using Perlin noise [67], 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).

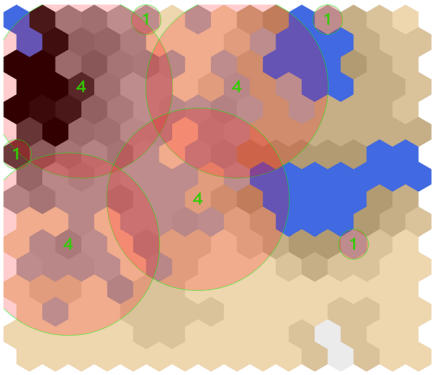
Second, the map is populated by iteratively adding an individual on the grid. Initially, a few individuals are added in different land cells randomly chosen, provided that the cell is next to a water cell. Then, a new individual is added at random following a probability distribution. The closer to an already populated cell, the higher its probability to welcome a new individual. The water cells and the cells that already host a maximum number of individuals cannot host a new individual. The process is repeated a number of times which at last corresponds to the total population in the map. Figure 1a 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 a maximum number of 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 different types of agents $type_1, type_2, \ldots$. 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 C that is not of water type, one considers an agent a_i^C of cost $f(a_i^C) = i$ which corresponds to a facility of type $type_i$ to be potentially deployed in the cell C. 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. Then, the agent-to-skill mapping α is defined as follows. An agent a_i^C has the skill s_P if the grid cell P is within the reach of the facility a_i^C, i.e., if the distance between the grid cells C and P is less than or equal to $i-1$. Lastly, we have added for each instance a set of constraints forbidding the joint deployment of two facilities a_i^C, a_j^C located at the same cell C. Thus, 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.

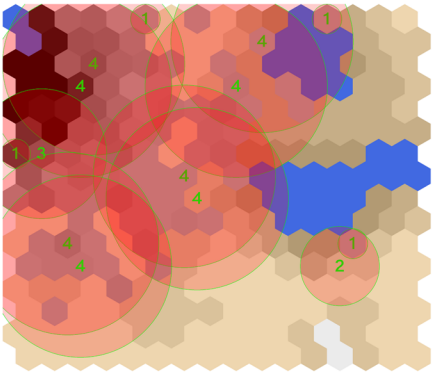
We have focused on four sets of 100 map instances of different sizes by varying their "resolution" r from 1 to 4, and with default values for the remaining parameters. These resulted in the four sets of instances map-r1, map-r2, map-r3 and map-r4. The map depicted



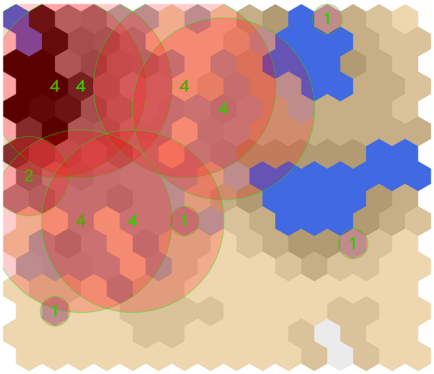
(a) A populated map generated by our procedure ($r = 3$, $c = 1$).



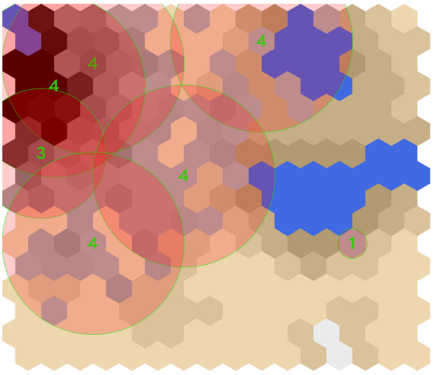
(b) An optimal efficient team T_1 ($f(T_1) = 20$, $rc(T_1, 1) = 13$, $pc(T_1, 1) = .20$).



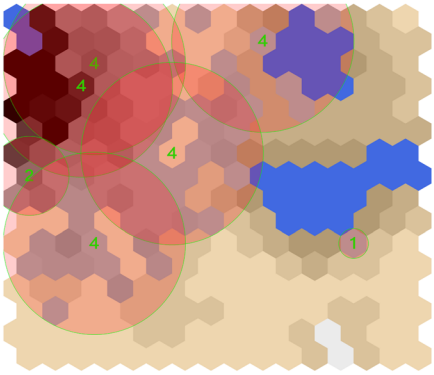
(c) An optimal 1-robust team T_2 ($f(T_2) = 41$, $rc(T_2, 1) = 0$, $pc(T_2, 1) = 1$).



(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 ($f(T_5) = 23$, $rc(T_5, 1) = 8$, $pc(T_5, 1) = .90$).

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

in Fig. 1a represents an instance of the set map-r3. Considering four such sets allowed us to focus on weighted TF instances on different sizes in terms of number of agents and skills.

For instance, the map-r3 set was formed of instances with an average of 537 agents / 75 skills, while the map-r4 set consisted of instances with an average of 1535 agents / 285 skills. Table 2 summarizes the characteristics of each set.

Figure 1 depicts for a given map instance from the set map-r3 an optimal team, that is respectively efficient (Fig. 1b), 1-robust (Fig. 1c), $\langle 1, .99\rangle$-partially robust (Fig. 1d), $\langle 1, .95\rangle$ -partially robust (Fig. 1e), and $\langle 1, .90\rangle$-partially robust (Fig. 1f). For instance, the optimal efficient team (Fig. 1b) 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, 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.

7 Empirical results

We have implemented our algorithms PR and PR_A (cf. Algorithms 1 and 2) to compute partially robust solutions, as well as algorithms to compute solutions related to the existing solution concepts of optimal efficiency (denoted by TF for short), (full) robustness (RobTF), and recoverability (RecTF). The notion of optimality for RecTF (i.e., given k, one seeks for a $\langle k, r\rangle$-recoverable team T whose overall cost $f(T) + rc(T, k)$ is minimal) and the corresponding encodings were the same ones as described and used in [3]; TF was computed using Eq. 2 under the set of constraints given in Eq. 1; and RobTF through Eqs. 2 and 6.

All the solution concepts and algorithms were empirically evaluated and compared on the benchmarks described in the previous section. The evaluation was performed from the viewpoint of computational efficiency (runtime), but also in terms of solution quality: more precisely, we compared for each instance the *deployment cost* of the computed solution, its *skill coverage* when k agents are lost, and its *recovery cost* afterwise. We also analyzed how the parameter t in $\langle k, t\rangle$-partially robustness impacts those results, and focused on values $t \in \{0.99, 0.95, 0.90\}$.

We used CPLEX as the constraint solver in all 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 s for each run of the algorithm and for each instance; memory-out was set to 32 Gb for each such run.

7.1 Computational efficiency

Table 3 shows the number of instances solved within the time limit of 3600 s for each method, except for TF and RobTF for which all instances were solved. In the particular case of the anytime algorithm PR_A, an instance is considered to be solved when an optimal team is found, i.e., the algorithm terminates without being interrupted. We also measured the time in seconds required for each solution concept and algorithm. Figure 2 shows four cactus plots for different values of k in $\{1, 2, 3, 4\}$ (the results for PR and PR_A are shown using cut+). Each plot gives for each notion the number of instances solved in a given amount of time. Figures 3, 4 and 5 provide through scatter plots some additional insights on the relative efficiency between (i) an implementation of PR with (our first

Table 3 Number of solved instances with a time out of 3600 s, for each set of instances and each solution concept

Type	#instances	k	#solved			
			RecTF	PR · PR_A (no_cut-cut-cut+)		
				$t = .90$	$t = .95$	$t = .99$
rtf30	10	1	⋆	⋆-⋆-⋆ · ⋆-⋆-⋆	⋆-⋆-⋆ · ⋆-⋆-⋆	⋆-⋆-⋆ · ⋆-⋆-⋆
		2		8-⋆-⋆ · ⋆-⋆-⋆	3-⋆-⋆ · 6-⋆-⋆	3-⋆-⋆ · 6-⋆-⋆
		3		4-⋆-⋆ · 6-⋆-⋆	0-⋆-⋆ · 2-⋆-⋆	0-⋆-⋆ · 2-⋆-⋆
		4		0-⋆-⋆ · 3-⋆-⋆		
cire100	10	1		9-⋆-⋆ · ⋆-⋆-⋆		
		2				
		3	4			
		4	0			
cire150	10	1	⋆	⋆-⋆-⋆ · ⋆-⋆-⋆	0-⋆-⋆ · 0-⋆-⋆	
		2				
		3	0			
		4		0-⋆-⋆ · 0-9-9		
scp1000	10	1	⋆	⋆-⋆-⋆ · ⋆-⋆-⋆		
		2	3			
		3	0		1-9-9 · 0-6-6	0-0-0 · 0-0-0
		4				
map-r1	100	1	⋆	99-⋆-⋆ · ⋆-⋆-⋆	86-⋆-⋆ · 99-⋆-⋆	86-⋆-⋆ · 99-⋆-⋆
		2		41-⋆-⋆ · 84-⋆-⋆	33-⋆-⋆ · 75-⋆-⋆	33-⋆-⋆ · 75-⋆-⋆
		3		11-⋆-⋆ · 89-⋆-⋆	11-⋆-⋆ · 89-⋆-⋆	11-⋆-⋆ · 89-⋆-⋆
		4		11-⋆-⋆ · 99-⋆-⋆	11-⋆-⋆ · ⋆-⋆-⋆	11-⋆-⋆ · ⋆-⋆-⋆
map-r2	100	1	99	39-⋆-⋆ · 42-⋆-⋆	14-⋆-⋆ · 15-⋆-⋆	
		2	1			
		3	0		0-⋆-⋆ · 0-⋆-⋆	
		4				
map-r3	100	1				
		2		0-61-70 · 0-83-84	0-66-69 · 0-83-82	
		3		0-32-28 · 0-41-43	0-29-31 · 0-46-49	
		4		0-14-15 · 0-22-21	0-22-22 · 0-31-29	
map-r4	100	1	0	0-91-99 · 1-⋆-⋆	0-39-48 · 0-56-58	0-12-17 · 0-28-32
		2		0-0-1 · 0-1-2		0-0-1 · 0-1-1
		3			0-0-0 · 0-0-0	
		4				

The symbol ⋆ means that all instances were solved. When three values p-q-r are given, p denotes the results without cut, q with the first version of the cut, and r with the second version of the cut (cut+)

version of) the cut and without the cut (Fig. 3); (ii) an implementation of PR with our first version of the cut and cut+ (Fig. 4); (iii) PR and PR_A, both using cut+ (Fig. 5).

First, it can be seen in Table 3 that both PR and PR_A are proved to be much more efficient than RecTF: for example, no RecTF solution could be found for any of the map-r3 instances, whereas PR and PR_A could find one for a reasonable proportion of these instances, using any of our cuts (cut / cut+) presented in Sect. 5. Note that all map-r2 instances were solved using PR or PR_A for $k \in \{3, 4\}$, whereas none of them was solved for RecTF.

Table 3 also shows how our cuts play a crucial role in the efficiency of both algorithms PR and PR_A, as few instances overall were solved without exploiting them. The impact of the cut even on instances that were solved without using it is also seen in Fig. 3. The reason why the cut did not impact the runtime efficiency of scp1000 instances for $t = .90$ is because for these instances, optimal efficient teams were all optimal $\langle k, .90\rangle$-partially robust teams (see Fig. 8 which we discuss later): an optimal team was always found at the first iteration step (line 3 of Algorithm 1), so the cut was simply not used.

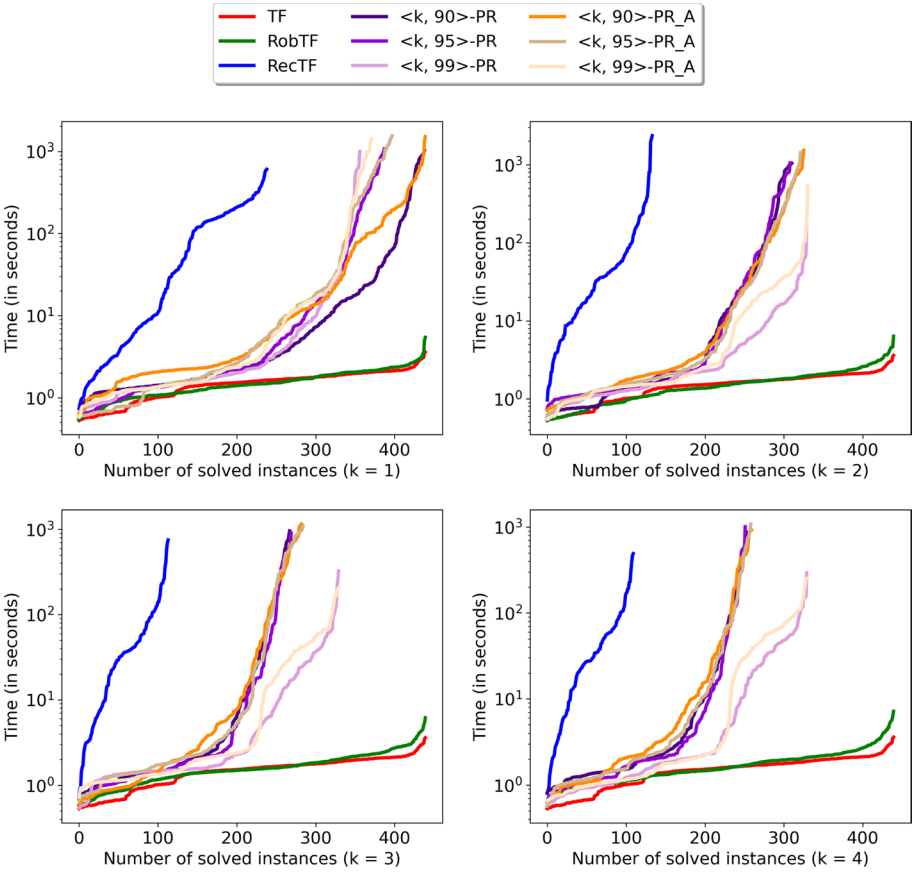


Fig. 2 Time results on all instances for $k \in \{1, 2, 3, 4\}$

One can also see from Table 3 that in the majority of cases, the use of cut+ instead of the first version of our cut allows one to solve slightly more instances. As shown in Fig. 4, when considering the instances that were solved in both cases, using cut+ in PR was more efficient for $t = .99$ compared to $t = .90$. For $t = .90$, the impact was lower, as the solver required more time to preprocess the additional constraints provided by our cut+ without gaining significant efficiency in the solving phase. This can be attributed to the fact that these instances were relatively small, and our cut+ did not provide much benefit. For instance, instances with approximately 100 agents could be solved almost instantly even without using our cut+. However, it can be seen that adding more constraints learned from a single counter-example is more beneficial for higher values of t. Indeed, the closer t is to 1, the more teams a cut prunes. It is actually easy to see that if $t = 1$, PR only performs two iterations to reach optimality: Eqs. 6, 9 and 8 coincide, i.e., they all ensure that each skill is covered by at least $k + 1$ agents, i.e., that the team is k-robust.

The phenomenon described above can also be seen in the cactus plots in Fig. 2. For higher values of k, the optimal partially robust teams could be found more efficiently for $t = .99$ than for lower values of t: indeed, in those settings optimal partially robust teams are typically "far" from optimal efficient ones in terms of deployment cost, suggesting an

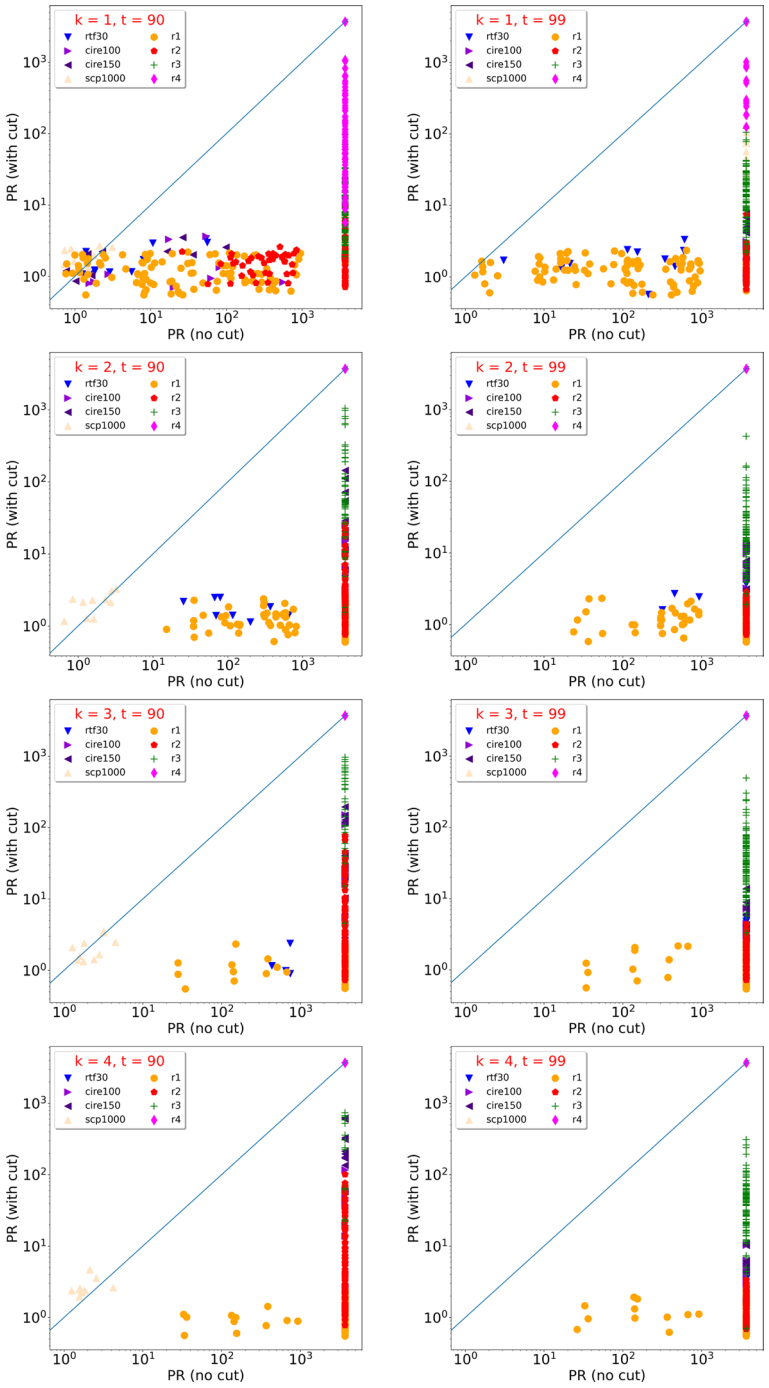


Fig. 3 Comparison between the implementation of PR with the first version of the cut, and without

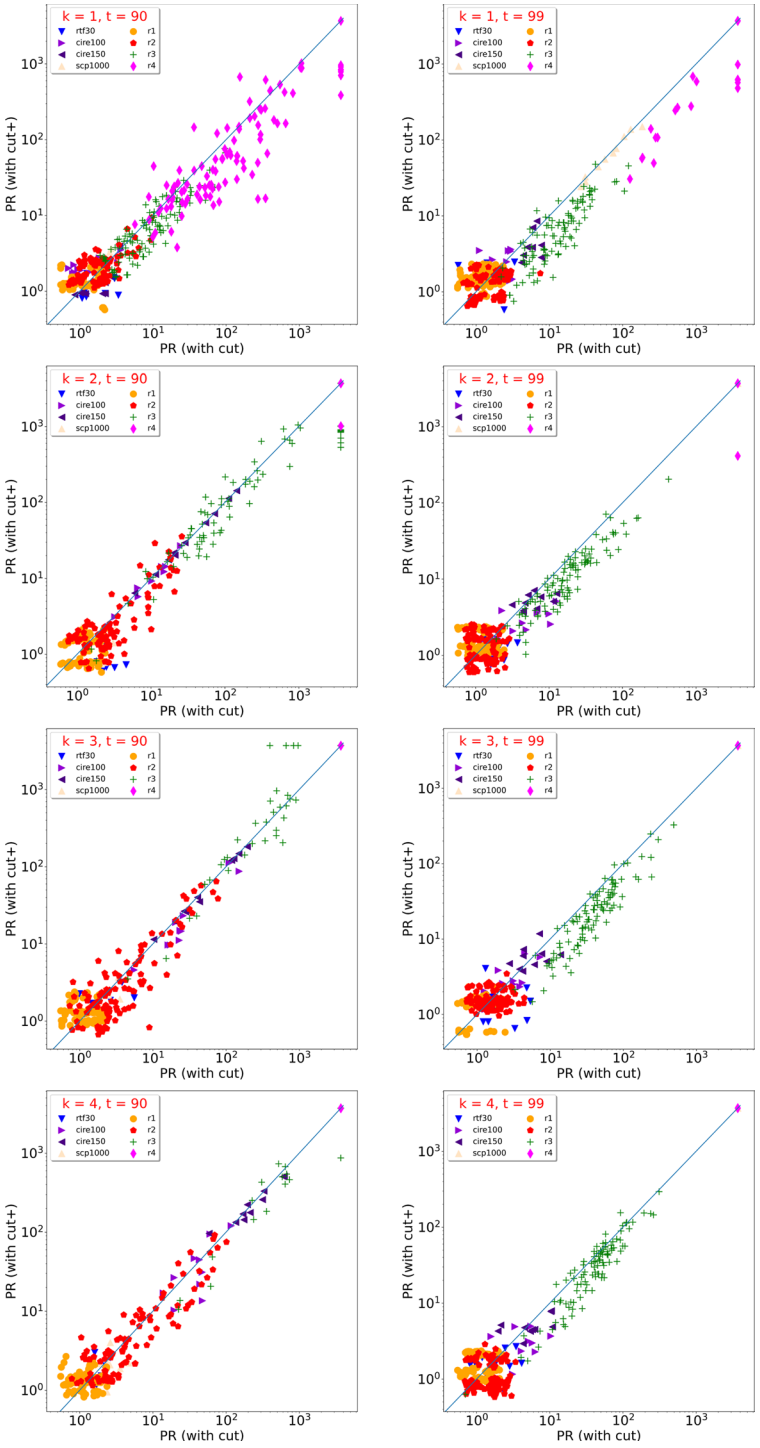


Fig. 4 Comparison between the use of (the first version of the) cut and cut+ for PR

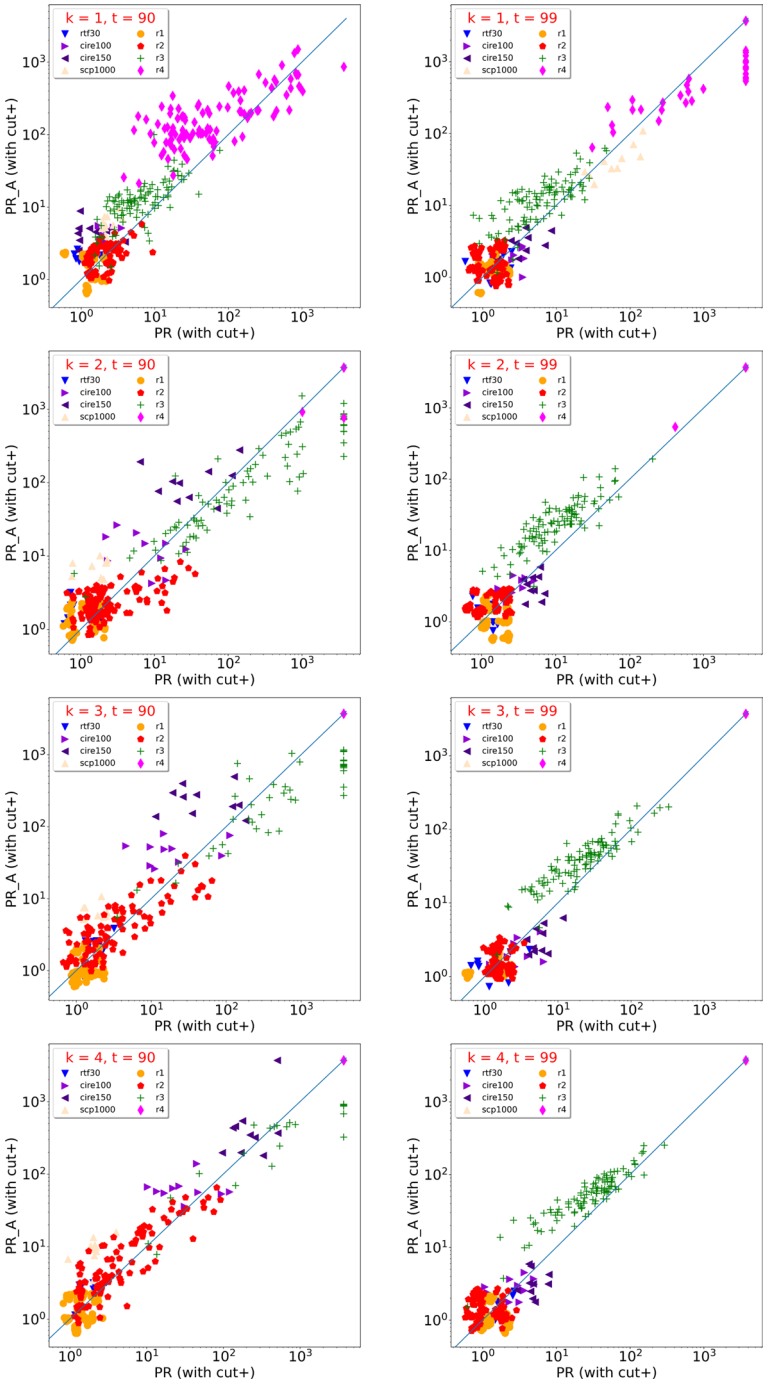


Fig. 5 Comparison between PR and PR_A in terms of solving time (in seconds), both implemented with cut+

increasing impact of pruning the search space through our cut+. When $k = 1$, optimal partially robust teams could be found slightly more efficiently for lower values of t: indeed, those teams where close to the optimal efficient ones, and only few iterations were necessary to compute them without the need for pruning the search space. In contrast, for higher values of k the optimal partially robust teams could be found more efficiently when $t = .99$ compared to lower values of t: indeed, for higher values of k and t, optimal partially robust teams are typically far from optimal efficient ones, suggesting that more iterations are typically required to compute them, and thus increasing the impact of an efficient pruning of the search space implemented through our cut+.

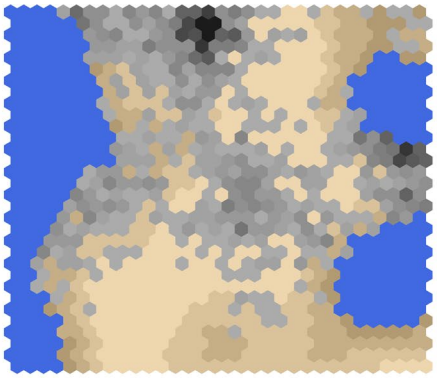
As to the comparison between PR and PR_A, Table 3 shows that in most of the benchmarks, PR_A could solve more instances than PR. However, through Fig. 5, PR is shown to be generally more efficient than PR_A for instances solved by both of them. This is explained by the fact that PR, as opposed to PR_A, iteratively computes the candidate efficient teams in an increasing order. Hence, for those partially robust teams whose cost is close to an optimal efficient team, PR performs better than PR_A. However, as the number of iterations increases, the chances of PR to fail to compute a solution within the time limit increase as well, and in these cases the success rate of PR_A is higher.

7.2 Solution Quality

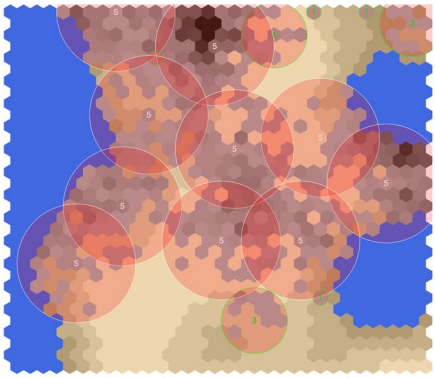
Let us start with a short analysis of the results in the map instance example provided in Fig. 1. Similar to our introductory example, one can see on these figures the advantages of forming partially robust teams instead of efficient or (fully) robust teams. For instance, the optimal $\langle 1, .99\rangle$-robust 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 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). The 1-partial coverage of the optimal efficient team (Fig. 1b) 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.

Figure 6 shows an instance of the map-r4 set that was not solved by PR and for which a sub-optimal solution was found using PR_A. The difference in terms of deployment cost between an optimal 2-robust team ($f(T'_2) = 194$, cf. Fig. 6c) and a sub-optimal $\langle 2, .90\rangle$-partially robust team found by PR_A ($f(T'_3) = 95$, cf. Fig. 6d) clearly shows the advantage of our anytime algorithm: even when an optimal solution is not found, its deployment cost appears to be quite satisfactory compared to the (fully) robust deployment solution, while providing a reasonable guarantee of population coverage in the disaster phase.

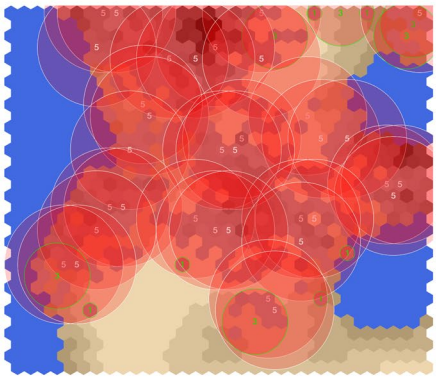
The advantages of the concept of partial robustness are made even clearer through Fig. 7 and 8, which show the deployment and overall cost on average for the generated facility location instances (map-r2 to map-r4 sets), and for the cire100, cire150 and scp1000 instances, respectively. The results for partially robust teams were the ones computed by PR_A after termination or currently found within the time limit, and thus some of the reported results may include sub-optimal teams. The percentage above each TF bar corresponds to the k-partial coverage value $pc(T, k)$ on average. It shows that the comparative behavior between optimal efficient teams, robust teams and partially robust teams described previously in Figs. 1 and 6 extends to all benchmarks on average. On the one hand, the overall costs of $\langle k, t\rangle$-partially robust teams remain below the deployment



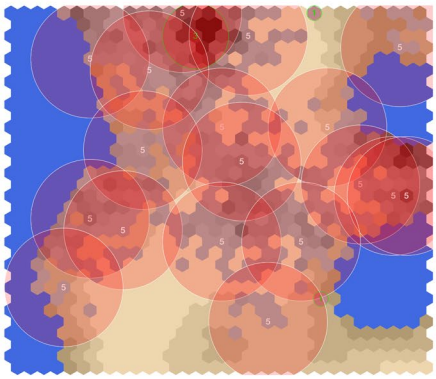
(a) A populated map generated by our procedure ($r = 4$, $c = 1$).



(b) An optimal efficient team T_1' with $f(T_1') = 63$



(c) An optimal 2-robust team with $f(T_2') = 194$



(d) A sub-optimal $\langle 2, .90\rangle$-partially robust team T_3' with $f(T_3') = 95$

Fig. 6 Optimal efficient and 2-robust teams and sub-optimal $\langle 2, .90\rangle$-partially robust team

of k-robust teams, even for high values of t. As an example, one can see that the overall deployment cost of $\langle 4, .95\rangle$-partially robust teams for the map-r4 set (cf. Fig. 7) is about only 60% of the deployment cost of 4-robust teams on average; note that for this benchmark and parameters, all of the solutions found by PR_A are sub-optimal (see Table 3). On the other hand, these costs remain arguably reasonable in comparison with optimal efficient teams, while efficiency can only guarantee a very low k-partial coverage, especially for the more structured set of facility location instances (e.g., below 50% on average for $k \geq 2$).

Figure 9 also clearly shows the advantage of sub-optimal partially robust teams found with PR_A compared to robust ones in terms of deployment cost. This time, the results are shown instance-wise through a scatter plot, and only for those instances for which PR could not find a solution within the time limit. Lastly, Fig. 10 shows the evolution with time execution of the deployment cost of the currently found solution using PR_A, for some randomly selected instances and parameters. The focus was given for instances for which an optimal solution was found in more than 100 s, after more than 3 iterations (lines 12 to 18 in Algorithm 2). This clearly shows how on these instances a near-optimal solution can be quickly found in the first iteration steps, and suggests that one can fairly extend this conclusion to the cases when an optimal solution is not found within a certain time limit.

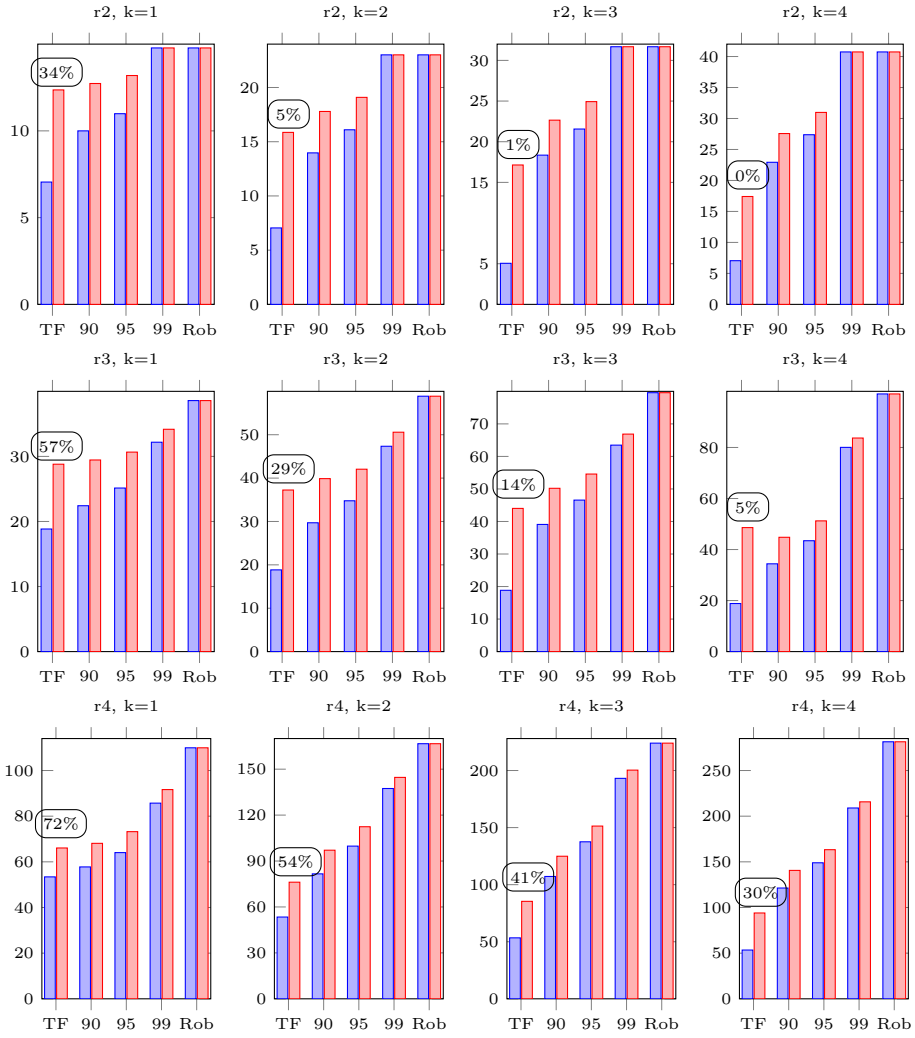


Fig. 7 Deployment and overall cost on average for the generated facility location instances. The percentage above each TF bar corresponds to the k-partial coverage value $pc(T)$ on average

8 Conclusion

We introduced the notion of partial robustness in Team Formation (PR-TF): given an integer k and a rational number $t \in [0, 1]$, a deployed team is $\langle k, t \rangle$-partially robust if whenever k agents are removed from it, the "ratio" of skills covered by the team formed of the remaining agents is kept higher than t. We formalized PR-TF and analyzed its computational complexity. We then provided two complete algorithms *à la* CEGAR [55–57], named PR and PR_A, to compute optimal partially robust teams, i.e., partially robust teams of lowest deployment cost. While PR searches for candidate teams in an increasing deployment cost order, PR_A is an anytime algorithm: starting from a sub-optimal candidate, it iteratively seeks for subsequent partially robust candidate teams with lower deployment costs. To empirically evaluate this new notion and our

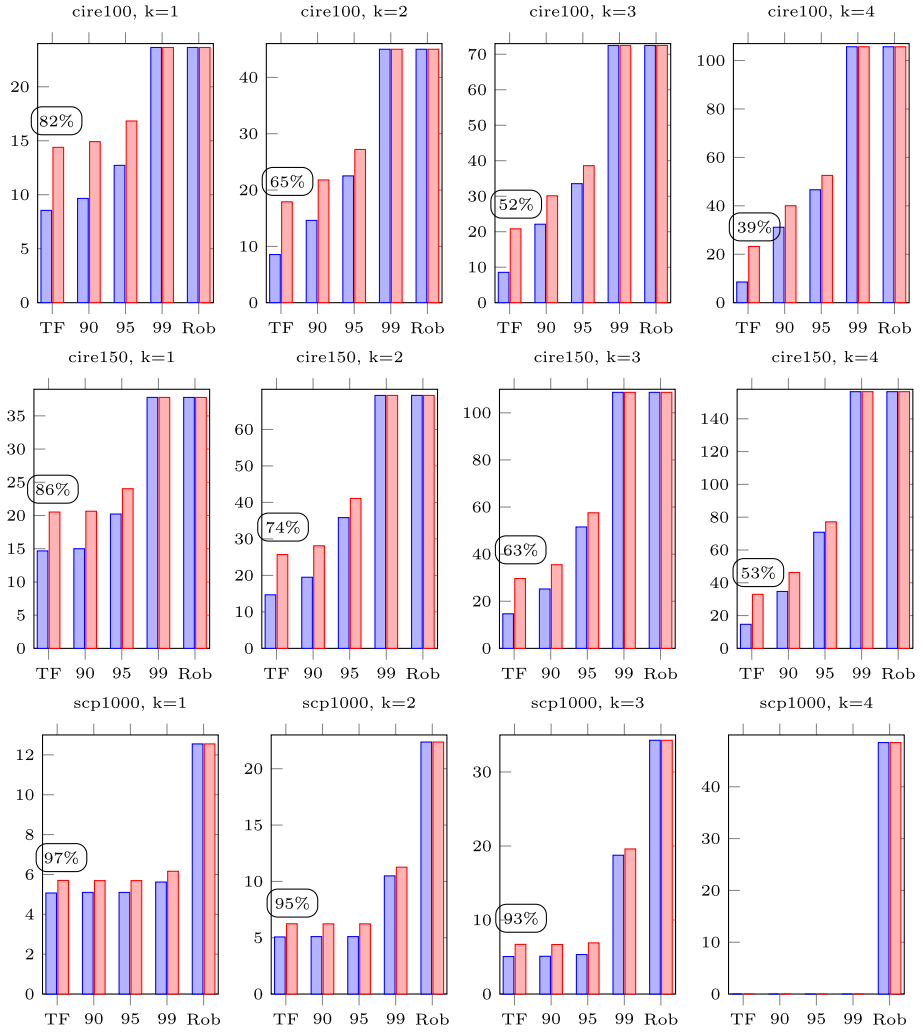


Fig. 8 Deployment and overall cost on average for the cire100, cire150 and scp1000 instances (values are divided by 100). The percentage above each TF bar corresponds to the k-partial coverage value $pc(T)$ on average

algorithms PR and PR_A, we compared with existing solution concepts and encodings for team efficiency (TF), robustness (RobTF) and recoverability (RecTF) on existing benchmarks, but also on new structured instances. For this purpose, we have developed a benchmark generation software publicly available [6]. Our software allows one to create instances modeling facility location problems on populated elevation maps. The structure of these maps can be adjusted through various parameters related to for example granularity and Perlin noise [67]. This type of instances is more structured than some existing benchmarks previously used in Team Formation [3, 2], since they are closer to real-world situations, and thus are arguably suitable to evaluate teams from the (partial) robustness viewpoint: a deployed team consists of a set of facilities, and the

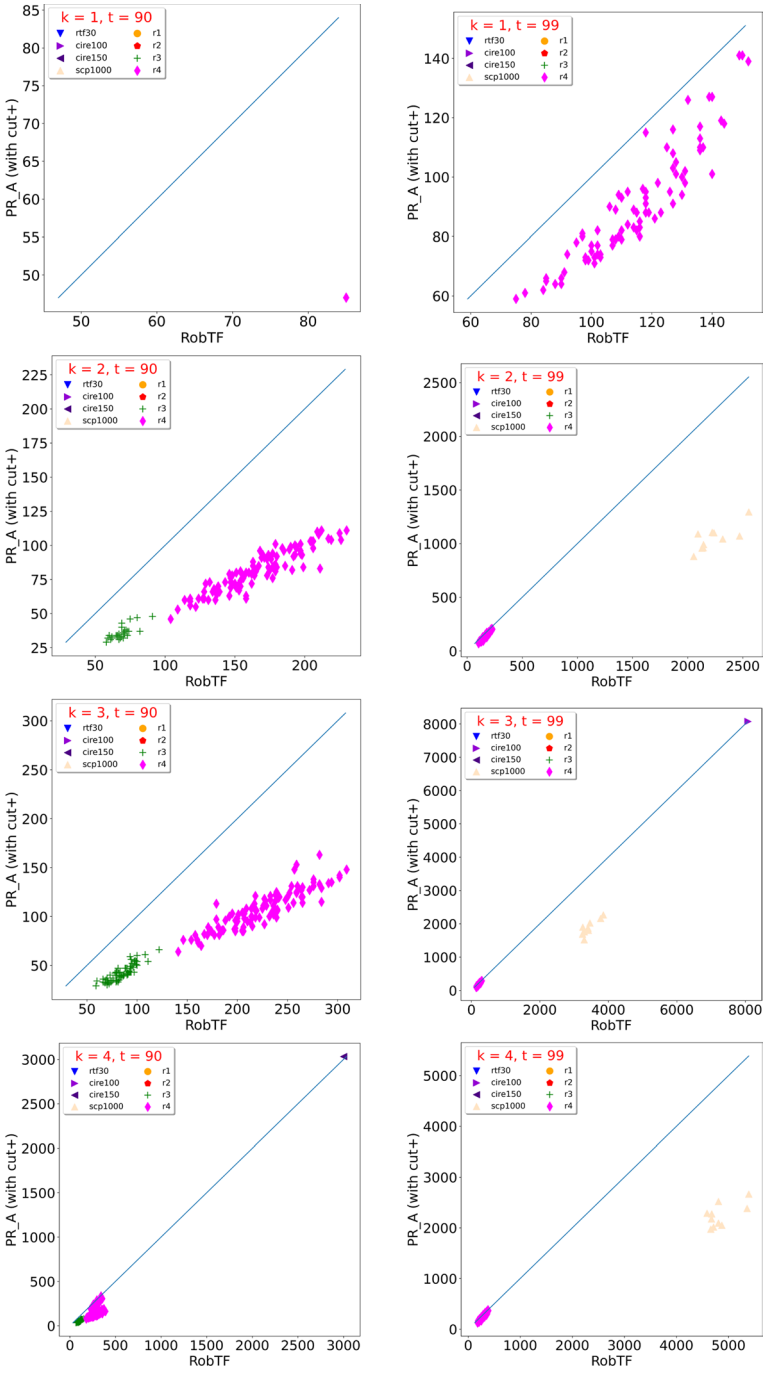


Fig. 9 Deployment cost of sub-optimal partially robust teams computed using PR_A compared with k-robust teams. Only instances not solved by PR are shown

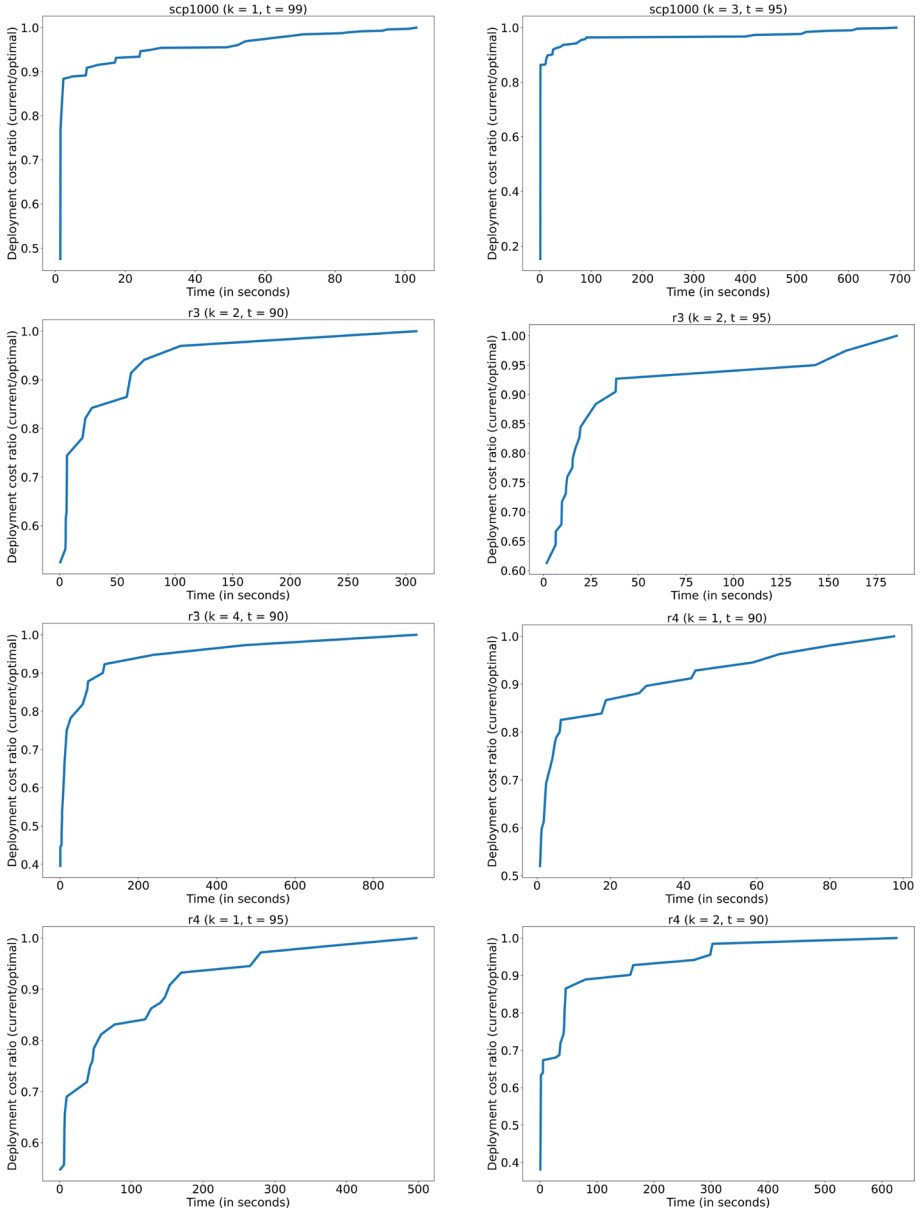


Fig. 10 Evolution in time execution of the deployment cost for some randomly selected instances (cire150, scp1000, r3 and r4 instances), using PR_A

goal is to find the least expensive set of facilities that provides an acceptable degree of population coverage, even in the case when some of them become out of service.

Our contribution reveals that PR-TF exhibits a number of interesting properties from several perspectives, when compared to the existing notions. First, we proved that the decision problem corresponding to PR-TF is $\mathbf{\Sigma_2^P}$-complete. On the one hand,

the counterpart problems for TF and RobTF are both **NP**-complete, which makes the partial robustness issue harder, computationally speaking. However, PR-TF is easier than RecTF since the latter was proved to be $\mathbf{\Sigma_3^P}$-complete [3]. Given the ever-increasing efficiency of modern constraint-based solvers, and since our algorithms are based on a approach that makes iterative use of such solvers, computing optimal partially robust teams could be done quite efficiently for instances of reasonable size: for instance, for one of our facility deployment benchmark consisting of instances with more than 500 agents and 75 skills in average (map-r3), our algorithms could compute an optimal solution for a majority of instances within one hour whereas the existing encodings for RecTF could not solve a single instance. Second, PR-TF provides by definition a guarantee of skill coverage in case of some agent losses. In comparison, the notions of TF and RecTF do not provide any such guarantee. In particular, we have empirically shown that the coverage loss in a disaster phase can be substantial for optimal efficient teams, even after losing a few agents from the team.

Third, we have evaluated the deployment cost of optimal partially robust teams and compared it with the other solution concepts. For most of the benchmarks, we found that there is a substantial gap between the deployment cost of $\langle k, t \rangle$-partially robust teams and the one of "fully" robust teams (RobTF), even for high values of t.

PR-TF is thus a new solution concept that can reasonably seen as an interesting trade-off between the existing notions of team efficiency, robustness and recoverability, both in terms of computational efficiency, skill coverage in the case of agent losses, and overall deployment cost.

This work opens a number of perspectives for further research. Considering other heuristics to quickly compute interesting cuts and determining how many can be added depending on the instance at hand will deserve to be investigated.

It could be also interesting to evaluate the practical benefits of considering the nature of the graph representing the problems. Because the TF problem is equivalent to the hitting set problem, all the parameterized complexity results that stand for the hitting set problem also stand for the team formation problem. In particular, it is well known that the hitting set problem parameterized by the size of the solution is a W[2]-complete problem in parameterized complexity theory [68], which shows the difficulty of computing one TF solution. Nevertheless, as shown in [69], it could be possible to investigate other parameters such as the cyclomatic number of the graph to obtain efficient algorithms. Because all our algorithms are based on the elementary brick that consists in computing TF solutions, improving this basic brick will lead to a reduction of the required time needed to obtain solutions for all the other tasks.

To deal with the particular structure of our map instances, we could also consider coarse-grained methods that would take into account regions of different sizes. More precisely, it could be possible to divide the maps into squares that can be handled efficiently. The challenging part is then to combine local solutions in order to get a global one.

Another perspective for further research consists in evaluating other solving paradigms such as SAT [58] or CSP [70]. In this respect, we plan to analyse the results obtained by the state-of-the-art SAT solvers during the last SAT competition on our TF benchmarks [71].

From an application point of view, we also plan to investigate if our framework can be used in combination with smart city energy models. For instance, we could consider the simulator proposed in [72] to evaluate the different team formation notions presented in this paper in a well controlled environment. Such experiments could be helpful in order to

estimate what is the good compromise between threshold values and the number of possibly defective agents.

Appendix A: Proofs of propositions

Proposition 4 DP-PR-TF is $\mathbf{\Sigma_2^P}$-complete.

Let us first prove that DP-PR-TF is in $\mathbf{\Sigma_2^P}$:

Proof (membership to $\mathbf{\Sigma_2^P}$) Consider the following algorithm:

1. Guess a set of agents $T \subseteq A$;
2. Check that T is efficient and $f(T) \leq c$;
3. Check using an NP-oracle that there does not exist a team $T' \subseteq T$ such that $|T'| \leq k$ and $cov(T \setminus T') < t$;

Obviously enough, this non-deterministic algorithm with a NP oracle runs in polynomial time and decides DP-PR-TF, which shows that DP-PR-TF is in $\mathbf{\Sigma_2^P}$. □

Before proving that DP-PR-TF is $\mathbf{\Sigma_2^P}$-hard, let us first consider the following intermediate decision problem, MINMAX-SAT [73]:

Definition 7 (MINMAX-SAT)

- *Input* A tuple $\langle X, Y, \varphi, p\rangle$, where $X = \{x_1, \ldots, x_n\}$ and $Y = \{y_1, \ldots, y_m\}$ are two disjoint sets of propositional atoms, φ is a 3-CNF propositional formula such that $Var(\varphi) = X \cup Y$, and p is a non-negative integer.
- *Question* For every truth-assignment to X, is there a truth-assignment to Y making at least p clauses in φ true?

MINMAX-SAT has been proven to be $\mathbf{\Pi_2^P}$-hard in [73], where $\mathbf{\Pi_2^P} = co\mathbf{\Sigma_2^P}$.

We now consider a variant of the MINMAX-SAT problem, which we call MAXMIN-SAT:

Definition 8 (MAXMIN-SAT)

- *Input* A tuple $\langle X, Y, \varphi, p\rangle$, where $X = \{x_1, \ldots, x_n\}$ and $Y = \{y_1, \ldots, y_m\}$ are two disjoint sets of propositional atoms, φ is a 3-CNF propositional formula such that $Var(\varphi) = X \cup Y$, and p is a non-negative integer.
- *Question* For every truth-assignment to X, is there a truth-assignment to Y making **at most** p clauses in φ true?

Let us prove that MAXMIN-SAT is $\mathbf{\Pi_2^P}$-hard:

Proof (MAXMIN-SAT is $\mathbf{\Pi_2^P}$-hard) We provide a polynomial-time reduction to MAXMIN-SAT from MINMAX-SAT. The reduction is defined as follows. Let $\langle X, Y, \varphi, p\rangle$ be an instance of MINMAX-SAT, i.e., $X = \{x_1, \ldots, x_n\}$ and $Y = \{y_1, \ldots, y_m\}$ are two disjoint sets of propositional atoms, φ is a 3-CNF formula consisting of q clauses such that

$Var(\varphi) = X \cup Y$, and p is a non-negative integer. The formula φ can be viewed as a set of clauses written as (l_i, l_j, l_k), where l_i, l_j, l_k are literals from $X \cup Y$.

With each clause $c_r \in \varphi$ we associate two fresh propositional atoms z_1^r, z_2^r and define the set $Z = \{z_1^r, z_2^r \mid c_r \in \varphi\}$ (note that Z is disjoint from X and Y). Now, for each clause $c_r = (l_i, l_j, l_k)$ from φ we associate the set of three clauses $C_r = \{(\overline{l_i}, z_1^r, z_2^r), (\overline{l_j}, z_1^r, \overline{z_2^r}), (\overline{l_k}, \overline{z_1^r}, z_2^r)\}$. Lastly, let us define the 3-CNF formula α made of the set of clauses $\bigcup_{c_r \in \varphi} C_r$. Note that $Var(\alpha) = X \cup Y \cup Z$.

Let us show that $\langle X, Y, \varphi, p\rangle$ is a "yes" instance for MINMAX-SAT if and only if $\langle X, Y \cup Z, \alpha, |\alpha| - p\rangle$ is a "yes" instance for MAXMIN-SAT, where $|\alpha|$ is the number of clauses in α.

(Only if part) Assume that $\langle X, Y, \varphi, p\rangle$ is a "yes" instance for MINMAX-SAT. Let ω_X be any assignment of X. Since $\langle X, Y, \varphi, p\rangle$ is a "yes" instance for MAXMIN-SAT, this means that there exists an assignment ω_Y of Y such that the assignment $\omega_X \cup \omega_Y$ makes at least p clauses in φ true. Now, for each clause $c_r = (l_i, l_j, l_k)$ from φ that is made true by the assignment $\omega_X \cup \omega_Y$, let us define the assignment ω_Z^r of the two variables z_1^c, z_2^c as follows. Since at least one of the literals l_i, l_j, l_k is true in c_r, if l_i is true in c_r, one sets $z_1^c = z_2^c = 0$; otherwise if l_j is true in c_r, one sets $z_1^c = 0$ and $z_2^c = 1$; and otherwise, if l_k is true in c_r, one sets $z_1^c = 1$ and $z_2^c = 0$. Doing so, one can verify that the assignment $\omega_X \cup \omega_Y \cup \omega_Z^r$ makes at least one clause from C_r false. Thus for each clause $c_r \in \varphi$ that is made true by the assignment $\omega_X \cup \omega_Y$, one can find an assignment ω_Z of Z so that the assignment $\omega_X \cup \omega_Y \cup \omega_Z$ makes one clause from C_r false.[4] Yet we know that the assignment $\omega_X \cup \omega_Y$ makes at least p clauses in φ true. Thus the assignment $\omega_X \cup \omega_Y \cup \omega_Z$ makes at least p clauses from C_r false, or equivalently it makes at most $|\alpha| - p$ clauses from C_r true. This means that $\langle X, Y \cup Z, \alpha, |\alpha| - p\rangle$ is a "yes" instance for MAXMIN-SAT.

(If part) Assume now that $\langle X, Y, \varphi, p\rangle$ is a "no" instance for MINMAX-SAT. So let ω_X be an assignment of X, then we know that for any assignment ω_Y of Y, the assignment $\omega_X \cup \omega_Y$ makes at most $p - 1$ clauses true in φ.

Now, let c_r be any clause from φ, and let ω_Y be any assignment of Y. One can easily see that for any assignment ω_Z^r of the two variables z_1^c, z_2^c, the assignment $\omega_X \cup \omega_Y \cup \omega_Z^r$ (i) makes at most one clause from C_r false if c_r is made true by $\omega_X \cup \omega_Y$, (ii) makes no clause from C_r false if c_r is made false by $\omega_X \cup \omega_Y$. But since we know that for any assignment ω_Y of Y, the assignment $\omega_X \cup \omega_Y$ makes at most $p - 1$ clauses true in φ, this means that for any assignment ω_Y of Y and for any assignment ω_Z of Z, the assignment $\omega_X \cup \omega_Y \cup \omega_Z$ makes at most $p - 1$ clauses from C_r false, or equivalently it makes at least $|\alpha| - p + 1$ clauses from C_r true. This means that $\langle X, Y \cup Z, \alpha, |\alpha| - p\rangle$ is a "no" instance for MAXMIN-SAT.

We have shown that $\langle X, Y, \varphi, p\rangle$ is a "yes" instance for MINMAX-SAT if and only if $\langle X, Y \cup Z, \alpha, |\alpha| - p\rangle$ is a "yes" instance for MAXMIN-SAT. Since MINMAX-SAT is $\mathbf{\Pi_2^P}$-hard, this proves that MAXMIN-SAT is $\mathbf{\Pi_2^P}$-hard. □

We are now ready to provide the proof of Proposition 4:

Proof (of Proposition 4) We intend to show that DP-PR-TF is $\mathbf{\Sigma_2^P}$-hard, by providing a polynomial-time reduction to its complementary problem from MAXMIN-SAT.

[4] Note that all pairs of sets $\{z_1^r, z_2^r\}$ and $\{z_1^{r'}, z_2^{r'}\}$ are pairwise disjoint when $r \neq r'$, so that all assignments $\{\omega_Z^r \mid c_r \in \varphi\}$ can be defined independently of each other.

Let $\langle X, Y, \varphi, p\rangle$ be an instance of MAXMIN-SAT, i.e., $X = \{x_1, \ldots, x_n\}$ and $Y = \{y_1, \ldots, y_n\}$ are two disjoint sets of propositional atoms, φ is a 3-CNF formula consisting of q clauses such that $Var(\varphi) = X \cup Y$, and p is a non-negative integer. Note that without loss of generality, we have here $|X| = |Y| = n$. Assume also without loss of generality that $p < |\varphi|$ (the case where $p = |\varphi|$ makes the instance trivially a "yes" one).

Let us associate with it a set of agents A, a set of skills S, a deployment cost function $f : A \mapsto \mathbb{N}$, a skill weight function $w : 2^S \mapsto [0, 1]$, and a skill-to-agent function $\beta : S \mapsto 2^A$. Note that these objects are not exactly the components of a weighted TF problem description, since one considers a skill-to-agent function $\beta : S \mapsto 2^A$ instead of an agent-to-skill function $\alpha : A \mapsto 2^S$. Intuitively, the function β associates with every skill from S the set of agents that possess the skill. This is made for simplicity in the reduction, however given A and S, an agent-to-skill function α can simply be derived from β as $\alpha(a) = \{s \in S \mid a \in \beta(s)\}$ for every agent $a \in A$. Then for instance, a skill $s \in S$ is covered by a team $T \subseteq A$ if and only if there is an agent $a \in T$ such that $a \in \beta(s)$; and a team is efficient if for all skills $s \in S$, $\beta(s) \cap T \neq \emptyset$.

Let us now define these objects in detail.

We define the set A of $4n + 1$ agents as $A = \{a_0\} \cup \{a_i, \overline{a_i}, b_i, \overline{b_i} \mid i \in \{1, \ldots, n\}\}$.

The cost function f is defined as $f(\{a_0\}) = 0$, and for every agent $a \in A \backslash \{a_0\}$, $f(a) = 1$.

The set S is formed of $4n + |\varphi|$ skills, where $|\varphi|$ is the number of clauses in φ, and is divided in two parts $S = S^* \cup S^\varphi$, with $|S^*| = 4n$ and $|S^\varphi| = \varphi$: the set S^φ depends on the clauses of φ, as opposite to the set S^* which only depends on A. So S^* is defined as $S^* = \{s_i^I, s_i^{II}, s_i^{III}, s_i^{IV} \mid i \in \{1, \ldots, n\}\}$, and S^φ is defined as $S^\varphi = \{s_1^\varphi, \ldots, s_q^\varphi\}$, where $q = |\varphi|$.

The skill weight function $w : 2^S \mapsto [0, 1]$ is characterized as follows. For every skill $s \in S$, one sets $w(s) = 1/|S|$. In addition, for every subset of skills $S' \subseteq S$, one defines $w(S') = 1$ if there exists $i \in \{1, \ldots, n\}$ such that $\{a_i, b_i\} \subseteq S'$ or $\{\overline{a_i}, \overline{b_i}\} \subseteq S'$, or if $S^\varphi \subseteq S'$; otherwise $w(S') = \sum_{s \in S'} w(s)$. Note that while the domain of w is in size exponential in $|S|$, the *representation* of w is in size polynomial in $n + |\varphi|$, i.e., in $|S|$.

Lastly, the skill-to-agent function $\beta : S \mapsto 2^A$ is defined as follows. For each $i \in \{1, \ldots, n\}$:

- $\beta(s_i^I) = \{a_i, \overline{a_i}\}$
- $\beta(s_i^{II}) = \{b_i, \overline{b_i}\}$
- $\beta(s_i^{III}) = \{a_i, \overline{b_i}\}$
- $\beta(s_i^{IV}) = \{\overline{a_i}, b_i\}$.

And for each skill $s_r^\varphi \in S^\varphi$, one identifies $\beta(s_r^\varphi)$ depending on the clause $c_r = (l_i, l_j, l_k)$ from φ. Beforehand, let us first consider the mapping γ associating any literal over $X \cup Y$ with a pair of elements of A, defined for every (possibly negated) literal l_i as

$$\gamma(l_i) = \begin{cases} \{a_i, b_i\} & \text{if } l_i \text{ is a positive literal over } X, \\ \{\overline{a_i}, \overline{b_i}\} & \text{if } l_i \text{ is a negative literal over } X, \\ \{a_i, \overline{a_i}\} & \text{if } l_i \text{ is a positive literal over } Y, \\ \{b_i, \overline{b_i}\} & \text{if } l_i \text{ is a negative literal over } Y. \end{cases}$$

Now, for each clause $c_r = (l_i, l_j, l_k)$ from φ, we define $\beta(s_r^\varphi)$ as $\beta(s_r^\varphi) = \{a_0\} \cup \gamma(l_i) \cup \gamma(l_j) \cup \gamma(l_k)$.

Example 2 For the sake of illustration, let us give an example of how the skill-to-agent function $\beta : S \mapsto 2^A$ is constructed from an instance $\langle X, Y, \varphi, p\rangle$ of MAXMIN-SAT, for

skills from S^{φ}. Let $X = \{x_1, x_2, x_3, x_4\}$, $Y = \{y_1, y_2, y_3, y_4\}$, and φ is formed of the set of four clauses $\{(x_1, x_2, \overline{x_3}), (\overline{x_1}, x_4, \overline{y_1}), (x_2, y_2, \overline{y_3}), (y_1, \overline{y_2}, y_3)\}$. Since φ has four clauses, S^{φ} is formed of four skills $s_1^{\varphi}, s_2^{\varphi}, s_3^{\varphi}, s_4^{\varphi}$ (one skill for each clause from φ), and for each one of these skills s_i^{φ}, $\beta(s_i^{\varphi})$ is defined as follows:

$$\begin{aligned}
\beta(s_1^{\varphi}) &= \{a_0, a_1, b_1, a_2, b_2, \overline{a_3}, \overline{b_3}\} \ \text{(clause } (x_1, x_2, \overline{x_3})) \\
\beta(s_2^{\varphi}) &= \{a_0, \overline{a_1}, \overline{b_1}, a_4, b_4, b_1, \overline{b_1}\} \ \text{(clause } (\overline{x_1}, x_4, \overline{y_1})) \\
\beta(s_3^{\varphi}) &= \{a_0, a_2, b_2, \overline{a_2}, b_3, \overline{b_3}\} \quad \text{(clause } (x_2, y_2, \overline{y_3})) \\
\beta(s_3^{\varphi}) &= \{a_0, a_1, \overline{a_1}, b_2, \overline{b_2}, a_3, \overline{a_3}\} \ \text{(clause } (y_1, \overline{y_2}, y_3)) \ .
\end{aligned}$$

Let us associate now the skill-to-agent function β with the agent-to-skill function $\alpha : A \mapsto 2^S$ as $\alpha(a) = \{s \in S \mid a \in \gamma(s)\}$ for every agent $a \in A$.

So, we have associated with any instance $\langle X, Y, \varphi, p\rangle$ of MAXMIN-SAT a weighted TF problem description $\langle A, S, f, w, \alpha\rangle$ in time polynomial in the size of $\langle X, Y, \varphi, p\rangle$ (with in addition β serving as an intermediate function to characterize α).

Let us now show that $\langle X, Y, \varphi, p\rangle$ is a "yes" instance of MAXMIN-SAT if and only if there does not exist a $\langle k, t\rangle$-partially robust team $T \subseteq A$ such that T is efficient and $f(T) \leq c$, with $k = n + 1$, $t = (2n + p + 1)/|S|$, and $c = 2n$.

(Only if part) Assume that $\langle X, Y, \varphi, p\rangle$ is a "yes" instance for MAXMIN-SAT. So for any assignment ω_X of X, there exists an assignment ω_Y of Y such that the assignment $\omega_X \cup \omega_Y$ makes at most p clauses in φ true. Now, let $T \subseteq A$ be any team such that T is efficient and $f(T) \leq 2n$. We need to show that T is not $\langle k, t\rangle$-partially robust, with $k = n + 1$ and $t = (2n + p + 1)/|S|$.

First, let us remark that if $a_0 \notin T$, since T is efficient and $f(T) \leq 2n$, the team $T \cup \{a_0\}$ is also efficient and $f(T \cup \{a_0\}) \leq 2n$; in addition, T is $\langle k, t\rangle$-partially robust only if $T \cup \{a_0\}$ is $\langle k, t\rangle$-partially robust. This means that we can assume that $a_0 \in T$ without any harm. Second, each agent from A except a_0 has a unit cost, i.e., for each $a \in A \backslash \{a_0\}$, $f(a) = 1$. So if $|T \backslash \{a_0\}| = m < 2n$, then any addition of $2n - m$ agents $T' \subseteq A \backslash T$ to T can be done without any harm. That is to say, we still have that $T \cup T'$ is efficient, $f(T \cup T') \leq 2n$, and T is $\langle k, t\rangle$ -partially robust only if $T \cup T'$ is $\langle k, t\rangle$-partially robust. So overall, let us assume that $a_0 \in T$ and $|T \backslash \{a_0\}| = 2n$, and it is enough to prove that T is not $\langle k, t\rangle$-partially robust. Lastly, since T is efficient, it necessarily covers all skills from $S = S^* \cup S^{\varphi}$. On the one hand, all skills from S^{φ} are trivially covered by T since $a_0 \in T$ and for each $s_i^{\varphi} \in S^{\varphi}$, $a_0 \in \beta(s_i^{\varphi})$.

On the other hand, all skills from $S^* = \{s_i^I, s_i^{II}, s_i^{III}, s_i^{IV} \mid i \in \{1, \ldots, n\}\}$ are covered as well by T. So for each $i \in \{1, \ldots, n\}$, $\beta(s_i^I) \cap T \neq \emptyset$, $\beta(s_i^{II}) \cap T \neq \emptyset$, $\beta(s_i^{III}) \cap T \neq \emptyset$, and $\beta(s_i^{IV}) \cap T \neq \emptyset$. By construction of those $\beta(s_i^I)$, $\beta(s_i^{II})$, $\beta(s_i^{III})$, $\beta(s_i^{IV})$, for $i \in \{1, \ldots, n\}$, and since $|T \backslash \{a_0\}| = 2n$, it means that for each $i \in \{1, \ldots, n\}$, one has either (i) $\{a_i, b_i\} \subseteq T$ and $\{\overline{a_i}, \overline{b_i}\} \cap T = \emptyset$, either (ii) $\{\overline{a_i}, \overline{b_i}\} \subseteq T$ and $\{a_i, b_i\} \cap T = \emptyset$.

Let us now show that T is not $\langle k, t\rangle$-partially robust, i.e., one can find a set $T' \subseteq T$, $|T'| \leq k$, and such that $cov(T \setminus T') < t$. Let us now define the assignment ω_X of X from T as follows: for each $i \in \{1, \ldots, n\}$, $\omega_X(x_i) = 1$ if and only if $\{a_i, b_i\} \subseteq T$. In particular, from this definition of ω_X and because of the structure of T we know that (i) $\omega_X(x_i) = 1$ if and only if ($\{a_i, b_i\} \subseteq T$ and $\{\overline{a_i}, \overline{b_i}\} \cap T = \emptyset$), and (ii) $\omega_X(x_i) = 0$ if and only if ($\{\overline{a_i}, \overline{b_i}\} \subseteq T$ and $\{a_i, b_i\} \cap T = \emptyset$). Yet we know that $\langle X, Y, \varphi, p\rangle$ is a "yes" instance for MAXMIN-SAT. This means that there is an assignment ω_Y of Y such that the assignment $\omega_X \cup \omega_Y$ makes at most p clauses in φ true. We associate with such an assignment ω_Y a set of agents T' to remove from T as follows. First, let $a_0 \in T'$ (i.e., one removes a_0 from T). Second, for each $i \in \{1, \ldots, n\}$, if $\omega_X(y_i) = 1$ then one removes either a_i or $\overline{a_i}$

from T depending on whether a_i or $\overline{a_i}$ is in T; and if $\omega_X(y_i) = 0$ then one removes either b_i or $\overline{b_i}$ from T depending on whether b_i or $\overline{b_i}$ is in T. At this stage, we can remark that (i) T' contains a_0 and exactly one element of $\{a_i, b_i, \overline{a_i}, \overline{b_i}\}$ for each $i \in \{1, \ldots, n\}$; and (ii) $T \setminus T'$ contains exactly one element of $\{a_i, b_i, \overline{a_i}, \overline{b_i}\}$ for each $i \in \{1, \ldots, n\}$. Accordingly, $|T'| = n + 1$, so $|T'| \leq k$. It remains to show that $cov(T \setminus T') < t$.

By definition of T and T', we have that $T \setminus T'$ covers exactly $2n$ skills from the set S^*. And it can be verified by construction of the $\beta(s_r^\varphi)$, $c_r \in \varphi$, that for each clause c_r from φ, c_r is made true by the assignment $\omega_X \cup \omega_Y$ if and only if $\beta(s_r^\varphi) \cap (T \setminus T') \neq \emptyset$, if and only if the skill s_r^φ is covered by $T \setminus T'$. Thus the number of skills from S^φ that are covered by $T \setminus T'$ is equal to the number of clauses in φ that are made true by $\omega_X \cup \omega_Y$. Yet from the initial assumption, $\omega_X \cup \omega_Y$ makes at most p clauses in φ true. This means that at most p skills from S^φ are covered by $T \setminus T'$. To summarize, since on the one hand $T \setminus T'$ covers exactly $2n$ skills from the set S^*, and on the other hand $T \setminus T'$ covers at most p skills from S^φ, we get that $T \setminus T'$ covers at most $2n + p$ skills from S, i.e., $|\alpha(T \setminus T')| \leq 2n + p$.

Let us compute $w(T \setminus T')$. We already know that $T \setminus T'$ contains exactly one element of $\{a_i, b_i, \overline{a_i}, \overline{b_i}\}$ for each $i \in \{1, \ldots, n\}$. So by definition of the skill weight function $w : 2^S \mapsto [0, 1]$, we have that $w(\alpha(T \setminus T')) = \sum_{s_j \in T \setminus T'} w(s_j)$: indeed, we do not fall in the case where $w(\alpha(T \setminus T')) = 1$ since for each $i \in \{1, \ldots, n\}$, $\{a_i, b_i\} \not\subseteq \alpha(T \setminus T')$ and $\{\overline{a_i}, \overline{b_i}\} \not\subseteq \alpha(T \setminus T')$, and $S^\varphi \not\subseteq \alpha(T \setminus T')$ (recall that p is initially assumed to be strictly lower than $|\varphi| = |S^\varphi|$).

So we got that $|\alpha(T \setminus T')| \leq 2n + p$ and $w(\alpha(T \setminus T')) = \sum_{s_j \in T \setminus T'} w(s_j)$. Thus $\sum_{s_j \in T \setminus T'} w(s_j) = (2n + p)/|S|$. Hence, $cov(T \setminus T') = w(\alpha(T \setminus T')) = (2n + p)/|S|$. Yet $t = (2n + p + 1)/|S|$, thus $cov(T \setminus T') < t$.

We have proved that for any team T such that T is efficient and $f(T) \leq c$, one can find a set $T' \subseteq T$, $|T'| \leq k$, such that $cov(T \setminus T) < t$, with $c = 2n$, $k = n + 1$, and $t = (2n + p + 1)/|S|$. This means that there does not exist a $\langle k, t \rangle$-partially robust team $T \subseteq A$ such that T is efficient and $f(T) \leq c$, with $k = n + 1$, $t = (2n + p + 1)/|S|$, and $c = 2n$. This concludes the (Only if) part of the proof.

(If part) Assume that there does not exist a $\langle k, t \rangle$-partially robust team $T \subseteq A$ such that T is efficient and $f(T) \leq c$, with $k = n + 1$, $t = (2n + p + 1)/|S|$, and $c = 2n$. Let ω_X be any assignment of X. We need to show that there is an assignment ω_Y of Y such that $\omega_X \cup \omega_Y$ makes at most p clauses in φ true.

Let us associate with ω_X the team $T \subseteq A$ as follows:

$$\begin{aligned} T = \; & \{a_0\} \\ & \cup \{a_i, b_i \mid \omega_X(x_i) = 1, x_i \in X\} \\ & \cup \{\overline{a_i}, \overline{b_i} \mid \omega_X(x_i) = 0, x_i \in X\}. \end{aligned}$$

One can check that T is efficient: all skills from S^φ are covered by a_0, all for each $i \in \{1, \ldots, n\}$:

- the skill s_i^I is covered by either a_i or $\overline{a_i}$;
- the skill s_i^{II} is covered by either b_i or $\overline{b_i}$;
- the skill s_i^{III} is covered by either a_i or $\overline{b_i}$;
- the skill s_i^{IV} is covered by either $\overline{a_i}$ or b_i.

Yet from our initial assumption, we know that T is not $\langle k, t\rangle$-partially robust. This means that there exists a set $T' \subseteq T, |T'| \leq k$, such that $cov(T \setminus T') < t$. Yet we know that $t < 1$, since $|S| = 4n + |\varphi|$, $t = (2n + p + 1)/|S|$, and we initially assumed that $p < |\varphi|$. So we know that $w(T \setminus T') < 1$, thus by definition of the skill weight function w, this means that:

(i) for each $i \in \{1, \ldots, n\}$, $\{a_i, b_i\} \nsubseteq \alpha(T \setminus T')$ and $\{\overline{a_i}, \overline{b_i}\} \nsubseteq \alpha(T \setminus T')$; and
(ii) $S^{\varphi} \nsubseteq \alpha(T \setminus T')$.

From (ii) above, since a_0 covers all skills from S^{φ} and $a_0 \in T$, this means that a_0 must necessary be removed from T and thus T' necessary contains a_0. Yet $|T'| \leq k = n + 1$. So from (i) above and by contruction of T, this means that for each $i \in \{1, \ldots, n\}$, one needs to remove from T exactly one element among $\{a_i, b_i\}$ (in the case where $\{a_i, b_i\} \subseteq T$), or exactly one element among $\{\overline{a_i}, \overline{b_i}\}$ (in the case where $\{\overline{a_i}, \overline{b_i}\} \subseteq T$). So to summarize the structure of T':

- T' contains a_0;
- for each $i \in \{1, \ldots, n\}$, T' contains either exactly one element from $\{a_i, \overline{a_i}\}$, or exactly one element from $\{b_i, \overline{b_i}\}$.

And as a consequence, to summarize the structure of $T \setminus T'$:

- $T \setminus T'$ does not contain a_0;
- for each $i \in \{1, \ldots, n\}$, $T \setminus T'$ contains either exactly one element from $\{a_i, b_i\}$, or exactly one element from $\{\overline{a_i}, \overline{b_i}\}$.

Now, we associate with T' the assignment ω_Y of Y defined for each $i \in \{1, \ldots, n\}$ as $\omega_Y(y_i) = 0$ in the case where $\{a_i, \overline{a_i}\} \cap T' \neq \emptyset$, and thus $\omega_Y(y_i) = 1$ in the other case where $\{b_i, \overline{b_i}\} \cap T' \neq \emptyset$.

At this point, from the sole structure of $T \setminus T'$ we know that for each $i \in \{1, \ldots, n\}$, exactly one skill among $\{s_i^I, s_i^{II}\}$ and exactly one skill among $\{s_i^{III}, s_i^{IV}\}$ is covered by $T \setminus T'$. Thus exactly $2n$ skills from S^* are covered by $T \setminus T'$. And by definition of the skill weight function w, $w(\alpha(T \setminus T')) = \sum_{s \in \alpha(T \setminus T')} w(s) = |\alpha(T \setminus T')|/|S|$. Yet $w(\alpha(T \setminus T')) = cov(T \setminus T') < t = (2n + p + 1)/|S|$. Since $|\alpha(T \setminus T') \cap S^*| = 2n$, thus means that at most p skills from S^{φ} are covered by $T \setminus T'$, i.e., $|\alpha(T \setminus T')| < p$. Yet it can be verified by construction of $T \setminus T'$ and by definition of $\beta(s_r^{\varphi})$ for each clause c_r from φ that $T \setminus T'$ covers a skill s_r^{φ} if and only if the assignment $\omega_X \cup \omega_Y$ makes the clause c_r true. This precisely means that the assignment $\omega_X \cup \omega_Y$ makes at most p clauses from φ true.

We have proved that for any assignment ω_X of X, there is an assignment ω_Y of Y that makes at most p clauses from φ true. This means that $\langle X, Y, \varphi, p\rangle$ is a "yes" instance for MAXMIN-SAT and concludes the (If) part of the proof.

We have proved that $\langle X, Y, \varphi, p\rangle$ is a "yes" instance of MAXMIN-SAT if and only if there does not exist a $\langle k, t\rangle$-partially robust team $T \subseteq A$ such that T is efficient and $f(T) \leq c$, with $k = n + 1$, $t = (2n + p + 1)/|S|$, and $c = 2n$. This provides a reduction from MAXMIN-SAT to the complementary problem of DP-PR-TF. Yet MAXMIN-SAT is $\mathbf{\Pi_2^P}$-hard. Therefore, DP-PR-TF is $\mathbf{\Sigma_2^P}$-hard.

This concludes the proof of Prop. 4.3. □

Proposition 5 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 $|\{a_i \in T \mid \alpha(a_i) \cap S' \neq \emptyset\}| \geq k+1$.

Proof (Only if part) We show the contrapositive of the statement. If $T \subseteq A$ is not efficient, it is trivially not $\langle k,t\rangle$-partially robust. Now, let $S' \subseteq S$, $w(S \backslash S') < t$, let $T' = \{a_i \in T \mid \alpha(a_i) \cap S' \neq \emptyset\}$ and assume that $|T'| \leq k$. By definition of T', for each agent $a_i \in T \backslash T'$, $\alpha(a_i) \cap S' = \emptyset$. Thus $\alpha(T \backslash T') \subseteq S \backslash S'$. Since $w(S \setminus S') < t$ and w is monotone, $w(\alpha(T \backslash T')) < t$. Hence, $cov(T \setminus T') < t$, and so $pc(T,k) < t$, which precisely means that T is not $\langle k,t\rangle$-partially robust.

(If part) We show the contrapositive of the statement. Let $T \subseteq A$, and assume that T is not $\langle k,t\rangle$-partially robust and efficient. By definition, $pc(T,k) < t$, i.e., there exists $T' \subseteq T$, $|T'| \leq k$, $cov(T \setminus T') < t$, so $w(\alpha(T \backslash T')) < t$. Let $S' = S \backslash \alpha(T \backslash T')$. Accordingly, $w(S \backslash S') = w(S \backslash (S \backslash \alpha(T \backslash T'))) = w(\alpha(T \backslash T')) < t$. And by definition of S', for each $a_i \in T \setminus T'$, $\alpha(a_i) \cap S' = \emptyset$. Hence, since $|T'| \leq k$, we get that $|\{a_i \in T \mid \alpha(a_i) \cap S' \neq \emptyset\}| \leq k$. □

Appendix B: Parameters involved in the generation protocol of the populated map instances

In this section, we provide more details on the list of parameters involved in the generation protocol of our populated map instances. This protocol has been implemented into a software program that is publicly available [6].

In the first step, one generates an elevation map made of water parts, lands and mountains using Perlin noise [67]. This elevation map is created according to the following parameters:

- resolution r: an integer within range [1, 7] that determines the size of the map;
- complexity c: an integer within range [1, 4]. The higher the value c, the shorter the variation of the map in terms of elevation.

Given these two parameters, an n x n grid of numbers is created using Perlin noise, with $n = 2^{r+1}$. 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).

In the second step, the map is populated by iteratively adding an individual on the grid. This step is characterized by the following parameters:

- number of seeds s: an integer specifying the number of "starting cities", i.e., the number of cells used as a starting point for populating;
- total population p: the total number of inhabitants in the map;
- maximum density l: the maximum number of inhabitants in the same cell;
- sparse degree d: the distance within which a new inhabitant can be added nearby an already populated cell. The higher the value d, the more spread out the population on the map.

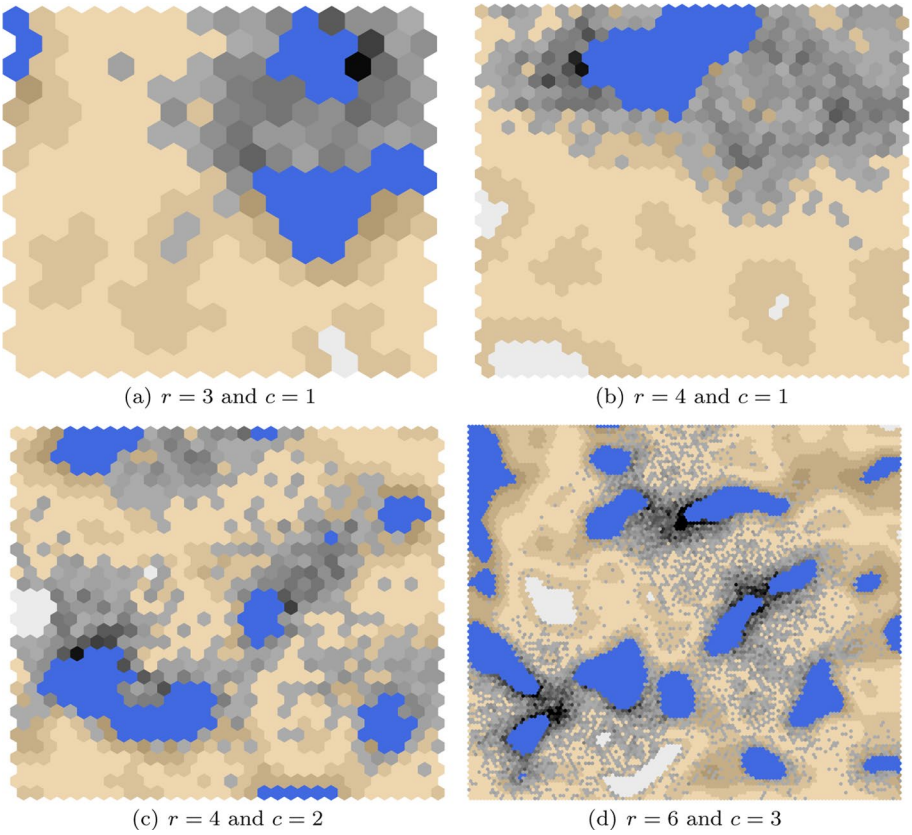


Fig. 11 Examples of generated maps with different values for the resolution r and the complexity c. The values of the remaining parameters were chosen by default

Thus, initially s individuals are added in s 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 that depends on d and such that 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 l individuals cannot host a new individual. The process is repeated p times which at last corresponds to the total population in the map. Figure 11 depicts some examples of populated maps using this method. Note that the values of s, p, l and d, when unspecified in the software program, are chosen by default according to r, e.g., for $s = r$, we set $d = r - 1$ and $p = 4^{r+1}$. 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 l individuals.

In the third step, a populated map is translated into a weighted TF problem description $\langle A, S, f, w_\Sigma, \alpha \rangle$. This step uses the following parameter:

– set of facility types $a \subseteq \{type_1, \ldots, type_{10}\}$: each type $type_i$ of agent corresponds to a facility that has a deployment cost equal to i and a cover range equal to $i - 1$.

The values of these parameters used to generate each one of our four sets of map instances were set as follows:

– *map-r1*: $r = s = 1, c = 1, p = 16, l = 20, d = 1, a = \{type_1, type_2\}$;
– *map-r2*: $r = s = 2, c = 1, p = 64, l = 20, d = 2, a = \{type_1, type_2, type_3\}$;
– *map-r3*: $r = 3, c = 1, s = 4, p = 256, l = 20, d = 3, a = \{type_1, type_2, type_3, type_4\}$;
– *map-r4*: $r = 4, c = 1, s = 7, p = 1024, l = 20, d = 4, a = \{type_1, type_3, type_5\}$.

Acknowledgements This work has benefited from the support of the JSPS KAKENHI Grant Number JP20K11947, the JSPS KAKENHI Grant Number JP21H04905 and the JST CREST Grant Number JPMJCR22D3, Japan.

References

1. Garey, M. R., & Johnson, D. S., Freeman W. H (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*.
2. Okimoto, T., Schwind, N., Clement, M., Ribeiro, T., Inoue, K., & Marquis, P. (2015). How to form a task-oriented robust team. In: Proceedings of the 14th international conference on autonomous agents and multiagent systems (AAMAS'15), pp. 395–403.
3. Demirović, E., Schwind, N., Okimoto, T., & Inoue, K. (2018). Recoverable team formation: Building teams resilient to change. In: Proceedings of the 17th international conference on autonomous agents and multi-agent systems (AAMAS'18), pp. 1362–1370.
4. Ahmadi-Javid, A., Seyedi, P., & Syam, S. S. (2017). A survey of healthcare facility location. *Computers & Operations Research, 79*, 223–263.
5. Schwind, N., Demirović, E., Inoue, K., Lagniez, J. (2021). Partial robustness in team formation: Bridging the gap between robustness and resilience. In: Proceedings of the 20th international conference on autonomous agents and multiagent systems (AAMAS'21), pp. 1154–1162.
6. Schwind, N., Demirović, E., Inoue, K., & Lagniez, J. (2022) TF-map-solver. https://github.com/nicolas-schwind/TF-map-solver
7. Juárez, J., Santos, C. P., & Brizuela, C. A. (2021). A comprehensive review and a taxonomy proposal of team formation problems. *ACM Computing Surveys, 54*(7), 1–33.
8. Bruneau, M., Chang, S. E., Eguchi, R. T., Lee, G. C., O'Rourke, T. D., Reinhorn, A. M., Shinozuka, M., Tierney, K., Wallace, W. A., & von Winterfeldt, D. (2003). A framework to quantitatively assess and enhance the seismic resilience of communities. *Earthquake Spectra, 19*, 733–752.
9. Schwind, N., Magnin, M., Inoue, K., Okimoto, T., Sato, T., Minami, K., & Maruyama, H. (2016). Formalization of resilience for constraint-based dynamic systems. *Journal of Reliable Intelligent Environments, 2*(1), 17–35.
10. Andrejczuk, E., Bistaffa, F., Blum, C., Rodríguez-Aguilar, J. A., & Sierra, C. (2019). Synergistic team composition: A computational approach to foster diversity in teams. *Knowledge-Based Systems, 182*(104), 799.
11. Kurtan, C., Yolum, P., & Dastani, M. (2020). An ideal team is more than a team of ideal agents. In: Proceedings of the 24th European conference on artificial intelligence (ECAI'20), vol. 325, pp. 43–50.
12. Barambones, J., Richoux, F., Imbert, R., & Inoue, K. (2020). Resilient team formation with stabilisability of agent networks for task allocation. *ACM Transactions on Autonomous and Adaptive Systems, 15*(3), 1–24.
13. Igarashi, A., Ota, K., Sakurai, Y., & Yokoo, M. (2019). Robustness against agent failure in hedonic games. In: Proceedings of the 28th international joint conference on artificial intelligence (IJCAI'19), pp. 364–370.
14. Okimoto, T., Ribeiro, T., Bouchabou, D., & Inoue, K. (2016). Mission oriented robust multi-team formation and its application to robot rescue simulation. In: Proceedings of the 25th international joint conference on artificial intelligence (IJCAI'16), pp. 454–460.

15. Terazawa, R., & Fujita, K. (2022) Allocation considering agent importance in constrained robust multi-team formation. In: Proceedings of the 14th international conference on agents and artificial intelligence (ICAART'22), pp. 131–138.
16. Malencia, M., Kumar, V., Pappas, G. J., & Prorok, A. (2021). Fair robust assignment using redundancy. *IEEE Robotics and Automation Letters, 6*(2), 4217–4224.
17. Bachrach, Y., & Shah, N. (2013). Reliability weighted voting games. In B. Vöcking (Ed.), *Algorithmic Game Theory* (pp. 38–49). Springer.
18. Bachrach, Y., Meir, R., Feldman, M., & Tennenholtz, M. (2011). Solving cooperative reliability games. In: Proceedings of the 27th conference on uncertainty in artificial intelligence (UAI'11), p. 27-34.
19. Bachrach, Y., Kash, I., & Shah, N. (2012). Agent failures in totally balanced games and convex games. In P. W. Goldberg (Ed.), *Internet and Network Economics* (pp. 15–29). Springer.
20. Kaminka, G. A., & Tambe, M. (2000). Robust agent teams via socially-attentive monitoring. *Journal of Artificial Intelligence Research, 12*, 105–147.
21. Kwak, J.y., Yang, R., Yin, Z., Taylor, M.E., & Tambe, M. (2011). Towards addressing model uncertainty: Robust execution-time coordination for teamwork. In: 2011 IEEE/WIC/ACM international conferences on web intelligence and intelligent agent technology, vol. 2, pp. 204–207
22. Rahwan, T., Michalak, T. P., Wooldridge, M., & Jennings, N. R. (2015). Coalition structure generation: A survey. *Artificial Intelligence, 229*, 139–174.
23. Sandholm, T., Larson, K., Andersson, M., Shehory, O., & Tohmé, F. (1999). Coalition structure generation with worst case guarantees. *Artificial Intelligence, 111*(1–2), 209–238.
24. Michalak, T. P., Rahwan, T., Elkind, E., Wooldridge, M., & Jennings, N. R. (2016). A hybrid exact algorithm for complete set partitioning. *Artificial Intelligence, 230*, 14–50.
25. Rahwan, T., Ramchurn, S. D., Jennings, N. R., & Giovannucci, A. (2009). An anytime algorithm for optimal coalition structure generation. *Journal of Artificial Intelligence Research, 34*, 521–567.
26. Rothkopf, M. H., Pekeč, A., & Harstad, R. M. (1998). Computationally manageable combinatorial auctions. *Management Science, 44*(7), 1131–1147.
27. Yeh, D. (1986). A dynamic programming approach to the complete set partitioning problem. *BIT Computer Science and Numerical Mathematics, 26*(4), 467–474.
28. Ieong, S., & Shoham, Y. (2005). Marginal contribution nets: a compact representation scheme for coalitional games. In: Proceedings of the 6th ACM conference on electronic commerce (EC'05), pp. 193–202.
29. Conitzer, V., & Sandholm, T. (2006). Complexity of constructing solutions in the core based on synergies among coalitions. *Artificial Intelligence, 170*(6–7), 607–619.
30. Ohta, N., Iwasaki, A., Yokoo, M., Maruono, K., Conitzer, V., & Sandholm, T. (2006). A compact representation scheme for coalitional games in open anonymous environments. In: Proceedings of the 21st national conference on artificial intelligence (AAAI'06), pp. 697–702.
31. Shrot, T., Aumann, Y., & Kraus, S. (2010). On agent types in coalition formation problems. In: Proceedings of the 9th international conference on autonomous agents and multiagent systems (AAMAS'10), pp. 757–764.
32. Bachrach, Y., Kohli, P., Kolmogorov, V., & Zadimoghaddam, M. (2013). Optimal coalition structure generation in cooperative graph games. In: Proceedings of the 27th AAAI conference on artificial intelligence (AAAI'13), pp. 81–87.
33. Okimoto, T., Schwind, N., Demirović, E., Inoue, K., & Marquis, P. (2018). Robust coalition structure generation. In: Proceedings of the 21st international conference on principles and practice of multi-agent systems (PRIMA'18), pp. 140–157.
34. Ismaili, A., Hazon, N., Watanabe, E., Yokoo, M., & Kraus, S. (2019). Complexity and approximations in robust coalition formation via max-min k-partitioning. In: Proceedings of the 18th international conference on autonomous agents and multiagent systems (AAMAS'19), p. 2036-2038
35. Schwind, N., Okimoto, T., Inoue, K., Hirayama, K., Lagniez, J., & Marquis, P. (2018). Probabilistic coalition structure generation. In: Proceedings of the 16th international conference on principles of knowledge representation and reasoning (KR'18), pp. 663–664.
36. Schwind, N., Okimoto, T., Inoue, K., Hirayama, K., Lagniez, J., & Marquis, P. (2021). On the computation of probabilistic coalition structures. *Autonomous Agents and Multi Agent Systems, 35*(1), 14.
37. Karp, R. M. (1972). *Reducibility among combinatorial problems* (pp. 85–103). Springer.
38. Basu, S., Sharma, M., & Ghosh, P. S. (2015). Metaheuristic applications on discrete facility location problems: A survey. *OPSEARCH, 52*, 530–561.
39. Farahani, R. Z., Asgari, N., Heidari, N., Hosseininia, M., & Goh, M. (2012). Covering problems in facility location: A review. *Computers & Industrial Engineering, 62*(1), 368–407.

40. Beraldi, P., & Ruszczynski, A. (2002). The probabilistic set-covering problem. *Operations Research, 50*(6), 956–967.
41. Chiang, C., Hwang, M., & Liu, Y. (2005). An alternative formulation for certain fuzzy set-covering problems. *Mathematical and Computer Modelling, 42*(3), 363–365.
42. Daskin, M. S. (1995). *Network and Discrete Location: Models, Algorithms, and Applications*. John Wiley and Sons.
43. Fischetti, M., & Monaci, M. (2012). Cutting plane versus compact formulations for uncertain (integer) linear programs. *Mathematical Programming Computation, 4*(3), 239–273.
44. Hwang, M., Chiang, C., & Liu, Y. (2004). Solving a fuzzy set-covering problem. *Mathematical and Computer Modelling, 40*(7), 861–865.
45. Lutter, P., Degel, D., Büsing, C., Koster, A. M., & Werners, B. (2017). Improved handling of uncertainty and robustness in set covering problems. *European Journal of Operational Research, 263*(1), 35–49.
46. Pereira, J., & Averbakh, I. (2013). The robust set covering problem with interval data. *Annals of Operations Research, 207*(1), 217–235.
47. Snyder, L. V., & Daskin, M. S. (2005). Reliability models for facility location: The expected failure cost case. *Transportation Science, 39*(3), 400–416.
48. Church, R. L., & Revelle, C. S. (1974). The maximal covering location problem. *Papers of the Regional Science Association, 32*, 101–118.
49. Berman, O. (1994). The p maximal cover - p partial center problem on networks. *European Journal of Operational Research, 72*(2), 432–442.
50. Berman, O., & Krass, D. (2002). The generalized maximal covering location problem. *Computers & Operations Research, 29*(6), 563–581.
51. Bläser, M. (2003). Computing small partial coverings. *Information Processing Letters, 85*(6), 327–331.
52. Nozick, L. (2001). The fixed charge facility location problem with coverage restrictions. *Transportation Research Part E: Logistics and Transportation Review, 37*(4), 281–296.
53. Berman, O., Drezner, T., Drezner, Z., & Wesolowsky, G. O. (2009). A defensive maximal covering problem on a network. *International Transactions in Operational Reseach, 16*(1), 69–86.
54. Church, R. L., & Scaparra, M. P. (2004). Identifying critical infrastructure: The median and covering facility interdiction problems. *Annals of the Association of American Geographers, 94*(3), 491–502.
55. Clarke, E.M., Grumberg, O., Jha, S., Lu, Y., & Veith, H. (2000). Counterexample-guided abstraction refinement. In: Proceedings of the 12th international conference on computer aided verification (CAV'00), pp. 154–169.
56. Janota, M., & Silva, J.P.M. (2011). Abstraction-based algorithm for 2QBF. In: Proceedings of the 14th International conference on theory and applications of satisfiability testing (SAT'11), pp. 230–244.
57. Janota, M., Klieber, W., Marques-Silva, J., & Clarke, E. M. (2016). Solving QBF with counterexample guided refinement. *Artificial Intelligence, 234*, 1–25.
58. Kleine Büning, H., & Bubeck, U. (2009). Theory of quantified boolean formulas. In A. Biere, M. Heule, H. van Maaren, & T. Walsh (Eds.), *Handbook of satisfiability, frontiers in artificial intelligence and applications* (Vol. 185, pp. 735–760). IOS Press.
59. Goultiaeva, A., Gelder, A.V., & Bacchus, F. (2011). A uniform approach for generating proofs and strategies for both true and false QBF formulas. In: Proceedings of the 22nd international joint conference on artificial intelligence (IJCAI'11), pp. 546–553.
60. Papadimitriou, C. H. (1994). *Computational complexity*. Addison-Wesley.
61. Beasley, J. E. (1990). OR-library: Distributing test problems by electronic mail. *Journal of the Operational Research Society, 41*(11), 1069–1072.
62. Bergman, D., & Ciré, A.A. (2016). Multiobjective optimization by decision diagrams. In: Proceedings of the 22nd international conference on principles and practice of constraint programming (CP'16), pp. 86–95.
63. Stidsen, T. R., Andersen, K. A., & Dammann, B. (2014). A branch and bound algorithm for a class of biobjective mixed integer programs. *Management Science, 60*(4), 1009–1032.
64. Gao, C., Yao, X., Weise, T., & Li, J. (2015). An efficient local search heuristic with row weighting for the unicost set covering problem. *European Journal of Operational Research, 246*(3), 750–761.
65. Kadioglu, S., & Sellmann, M. (2009). Dialectic search. In: Proceedings of the 15th International conference on principles and practice of constraint programming (CP'09), pp. 486–500.

66. Musliu, N. (2006). Local search algorithm for unicost set covering problem. In: Proceedings of the 19th international conference on industrial, engineering and other applications of applied intelligent systems, Advances in Applied Artificial Intelligence (IEA/AIE'06), pp. 302–311.
67. Perlin, K. (1985). An image synthesizer. In: Proceedings of the 12th annual conference on computer graphics and interactive techniques (SIGGRAPH'85), pp. 287–296.
68. Downey, R.G., & Fellows, M.R. (2013) Fundamentals of parameterized complexity. Texts in Computer Science, Springer.
69. Jansen, B. M. P. (2017). On structural parameterizations of hitting set: Hitting paths in graphs using 2-SAT. *Journal of Graph Algorithms and Applications, 21*(2), 219–243.
70. Rossi, F., van Beek, P., Walsh, T. (eds) (2006). Handbook of constraint programming, foundations of artificial intelligence, vol. 2. Elsevier.
71. Balyo, T., Heule, M. J., Iser, M., Järvisalo, M., & Suda, M. (Eds.). (2022). *SAT Competition 2022: Solver and Benchmark Descriptions.* Department of Computer Science: University of Helsinki.
72. Karnouskos, S., & de Holanda, T.N. (2009). Simulation of a smart grid city with software agents. In: Al-Dabass D, Katsikas SK, Koukos I, Zobel RN (eds) Proceedings of the 3rd UKSim european symposium on computer modeling and simulation (EMS'09), pp. 424–429.
73. Meyer, A.R., & Stockmeyer, L.J. (1972). The equivalence problem for regular expressions with squaring requires exponential space. In: Proceedings of the 13th annual symposium on switching and automata theory (SWAT'72), pp. 125–129.