Resilient Team Formation with Stabilisability of Agent Networks for Task Allocation

Jose Barambones
Universidad Politécnica de Madrid, Spain
j.barambones@upm.es

Florian Richoux
AIST, Japan
florian.richoux@aist.go.jp

Rocardo Imbert
Universidad Politécnica de Madrid, Spain
ricardo.imbert@upm.es

Katsumi Inoue
NII, Japan
inoue@nii.ac.jp

***Abstract*—Team formation (TF) faces the problem of defining teams of agents able to accomplish a set of tasks. Resilience on TF problems aims to provide robustness and adaptability to unforeseen events involving agent deletion. However, agents are unaware of the inherent social welfare in these teams. This paper tackles the problem of how teams can minimize their effort in terms of organisation and communication considering these dynamics. Our main contribution is twofold: first, we introduce the Stabilisable Team Formation (STF) as a generalisation of current resilient TF model, where a team is stabilisable if it possesses and preserves its inter-agent organisation from a graph-based perspective. Second, our experiments show that stabilisability is able to reduce the exponential execution time in several units of magnitude with the most restrictive configurations, proving that communication effort in subsequent task allocation problems are relaxed compared with current resilient teams. To do so, we developed SBB-ST, a branch-and-bound algorithm based on Distributed Constrained Optimisation Problems (DCOP) to compute teams. Results evidence that STF improves their predecessors, extends the resilience to subsequent task allocation problems represented as DCOP, and evidence how Stabilisability contributes to resilient TF problems by anticipating decisions for saving resources and minimizing the effort on team organisation in dynamic scenarios.**

I. Introduction

Resilient systems are resistant against perturbations caused by unforeseen events and capable of recover from a risky state as efficiently, cheap, and quick as possible. Resilience is studied in diverse areas, such as ecology [Hol73], psychology [Cou02], disaster management [BCE+03], society [WHCK03], community [LAP+10], engineering [YM16], and rescue response [ORBI16]. Resilience in AI is applied in Multi-Agent Systems (MAS) to extend robustness concepts [SMI+16], [SOI+13], and is modelled by different approaches such as parametric model checking [AMI16], stochastic models [PSZ+18], and constraint-based problems [OSC+15], [COSI15], [DSOI18], [SOI+18].

Team formation (TF) is a well-known constraint-based optimization problem that aims to find a team of agents able to accomplish a set of tasks efficiently, i.e., minimizing a cost function. The TF problem has been extended applying different resilience concepts: An agent team is *robust* if it maintains its efficiency when a number of agents are removed from it [OSC+15], and *recoverable* if the team is efficient and there is another set of agents capable to restore the efficiency if such deprivation occurs [DSOI18]. Although these features prepare teams against unexpected events and ensure their functionality, costs and teams are modelled with regard to a goal and obviating agent relationships. In a real scenario, agents conform teams to face a subsequent task allocation problem, where agents must negotiate and collaborate to share and fulfill involved tasks. Such organisation get impacted from dynamics that imply a removal/addition of agents in the team. Current resilient TF literature ignores such implications concerning team organisation and its possible states, so it is not adequate to represent the task allocation effort.

To illustrate it, consider a rescue response scenario, application extensively used for planning problems with MAS [RFMJ10], [PGCF+15], [SSV16]. Heterogeneous agents such as drones, vehicles, rescue staff and facilities, must coordinate to face emerging field tasks. Agents are assembled into teams able to accomplish these tasks according to their role and available resources. Fig. 1 shows a resilient team composed of drones (imagery, video, or delivering) and a base camp (data collection and monitoring). Each agent possesses an action range that determines which tasks and communication with other agents can be performed. Drones need to reassign tasks due to drone 4 failure. According to the TF literature, the team is robust because both tasks are accomplishable despite a member malfunction. But, even if the task is still within the range of neighbouring drones 3 and 5, they are out of their communication ranges (in fact, drone 5 is isolated), leading to unacceptable risks such as task conflicts, mismanagement, or accidents in the worst case. Success on tasks does not depend exclusively on agents' skills, but also on their organisation. The problem arises when the team is unable to self-organize given these dynamics, despite its robustness or recoverability.

Stability or Stabilisability definitions are broadly studied in different Computer Science areas, being prominent on Artificial Intelligence [TIW96], [WLL10], [SOI+13], [FAMO14], [COSI15], Distributed and Dynamic Systems [OW91], [Reb93], [TM04], [SVR+10]. In this paper we introduce the stabilisability concept from Schwind et al. [SOI+13], concretely on MAS in TF problems from a graph-based perspective. A team is *stabilisable* if it preserves certain graph properties to guarantee or minimize its organisation effort. Thus, resilience is extended for both team formation and decentralized task allocation problems.

Our main contribution is a generalisation of the resilient TF model including stabilisability. We compare the time performance of computing both recovery and stabilisable solutions.

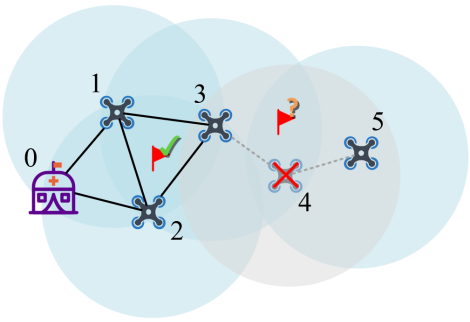


Fig. 1: Drone team in a rescue response scenario, where the drones 3 and 5 on the right are not capable to negotiate the eastern task after drone 4 went down.

To do so, we design SBB-ST, a TF algorithm based on distributed problem solving using DCOP framework. During experimentation, we configure stabilisability with different graph properties to compute solutions and we evaluate the effort of such teams during inter-agent organisation. Results show that stabilisability can significantly reduce the computation time compared with recoverable search, and in the subsequent task allocation problems for these teams.

The paper is structured as follows: Section 2 describes the context of the research. Section 3 formalizes the Stabilisable Team Formation proposal. Section 4 describes SBB-ST, our branch-and-bound algorithm based on the DCOP framework. Section 5 shows our experiments and results. Section 6 exposes our discussion on the study and methods used. The last sections describe related work, conclusions, and future research lines[1].

II. Background

According to Schwind and Okimoto et al. [SOI+13], [SMI+16], Resilience is defined as the capability of a system to resist and react to critical changes, decomposed in the following properties:

- *Resistance*: Maintain overall costs under a threshold against some perturbations. This property is inherited from the definition of robustness [Die17].
- *Recoverability*: The system must be capable to return from a risky state to a quality one as gracefully and efficiently as possible.
- *Functionality*: Maintain and guarantee the average quality state over the time.
- *Stabilisability*: Prevent, anticipate and restrict those decisions that involve an awkward system configuration.

These definitions are broad and their detailed description depend on the problem that it is being faced. When resilience is incorporated to a new problem, the first property implemented is resistance, also called robustness [COSI15], [OSC+15], [ORBI16], [Die17], [OSD+18]. Then, the remaining properties are considered according to the problem needs and their viability. In fact, stabilisability was defined as a desired but accessory property from resilience itself due to be considered as a function specified by the system controller or a social component to manage the system [SMI+16]. However, none of the properties but stabilisability, as it is shown in the following sections, is capable to model how agents are related and how they interact, something that have a big impact on the resistance and reaction of the MAS against unforeseen events.

This paper scopes resilient terminology for TF problems using MAS. Later, we concrete the definition of stabilisability of the agent network to introduce our resilient proposal.

A. Resilient Team Formation Problem

Team formation represents the problem of assembling an agent team to accomplish a set of tasks with the minimum cost.

1) Robust (Resistant) Team: A basic TF problem is described as the following question $TF = \langle A, P, f, \alpha \rangle$ [OSC+15] where:

- **Agents** $A = \{a_1, a_2, \ldots, a_n\}$ is the set of agents. A subset of selected agents is called a *team* $T \subseteq A$.
- **Tasks** $P = \{t_1, t_2, \ldots, t_n\}$ is the set of tasks. A subset of selected tasks is called a *goal* $G \subseteq P$.
- **Deployment Cost function** $f : 2^A \to \mathbb{N} \cup \{+\infty\}$ is the attached cost for agents in A, usually defined as the sum of costs of all agents in T: $f(T) = \sum_{a_i \in T} f(a_i)$
- **Allocation map** $\alpha : A \to 2^P$ the mapping of accomplishable tasks by agents.

Definition 1 (team efficiency)**.** *T is said to be efficient w.r.t. G if agents in T are able to accomplish all tasks from G.*

Definition 2 (c-cost)**.** *T is said to be c-costly if $f(T) \leq c$ and T is efficient w.r.t. G.*

Definition 3 (k-robustness)**.** *T is said to be $\langle c, k \rangle$-robust (or resistant) w.r.t. G if is c-costly and any set of k agents can be removed from T without disturbing the efficiency.*

In other words, solving a TF problem is finding a team with the lowest cost capable of fulfilling all the tasks of the goal. For simplicity, we assumed single costs for agents and simple tasks, but TF problems may represent more complex conditions. In the same way that a goal is divided on tasks, a "mayor task" or mission can be modelled as another set of sub-tasks. Moreover, according to the cost function, agents may possess different cost based on different contributions and its affordable tasks.

2) Recoverable Team: As a generalisation of the robust TF problem [DSOI18], a Recoverable Team Formation problem (RTF) is the following question $\langle TF, h \rangle$ where:

- **Team Formation tuple** $TF = \langle A, P, f, \alpha \rangle$
- **Recovery Cost function** $h : 2^A \to \mathbb{N} \cup \{+\infty\}$ is the cost of agents in the recovery team $T_{rec} \subseteq (A \setminus T)$. Agents from T_{rec} become part of the team when the initial T loses k agents. As f, h can be defined as the sum of costs: $h(T) = \sum_{a_i \in T} h(a_i)$

[1] Sources are accessible in the following GitLab repository: https://gitlab.com/team-formation/jade-stf

Definition 4 (k-recoverability). *T is $\langle c_d, k, c_r\rangle$-recoverable w.r.t. G if the deployment cost of T is c_d and, for any removal of k agents T_{rem} from T, there is an auxiliary T_{rec} with the worst recovery cost c_r such that $(T \setminus T_{rem}) \cup T_{rec}$ is efficient.*

Corollary 1. *If T is $\langle c_d, k, 0\rangle$-recoverable and $\forall a \in A, h(\{a\}) > 0$, T is $\langle c_d, k\rangle$-robust.*

A team is *k-recoverable* if exists an initial efficient team and a complementary "rescue" team that restores the efficiency after a deprivation of k agents without imposing any additional cost, just substituting the initial cost of deprived agents by the recovery cost of the new ones while preserving the c-costly condition. We define T' as a new team state such that $T' = (T \setminus T_{rem}) \cup T_{rec}$. The main idea is to minimize costs through restoring efficiency rather than composing robust teams. Okimoto et al. [OSC+15] proved that the robust TF is NP-hard as an optimisation problem, in the same fashion than the optimisation version of RTF and k-RTF[2] problems as a generalisation of TF [DSOI18].

B. Agent networks for Task Allocation

The main approaches for distributed problem solving through MAS are the following [CFMR14]:

- *Coalition/Team Formation problem*: How agents shall be grouped in profitable teams or coalitions.
- *Distributed Constraint Optimisation Problem*: How agents perform joint decisions for coordinated actions.
- *Task/Resource Allocation Problem (TA)*: How the distribution of resources among the agents are addressed.

TF is a well-suited scenario that combines to a greater or lesser extent all these issues. TF can be represented as a constraint-based problem, where teams are assembled based on allocation costs using optimisation techniques. However, TF is a one-shot problem that involves the goal success, not the fine-grained allocation from agents to tasks. Indeed, inter-agent communication is highly dependent of how teams have been formed. Runtime performance in TA algorithms depends on agent communication, mainly because the agent network defines how the search space is built and traversed. In distributed problem solving, agent communication in well-known algorithms and their variants follows concrete topologies, i.e., trees [PF05a], factor graphs [FRPJ08], or scale-free graphs [JZW13]. Furthermore, the number of messages, the message size, and the agent memory size are inherent of the complexity in distributed algorithms with MAS [FPY16]. Accordingly, studies show how graph networks have an impact in the performance [Gd08], [OHLN06]. However, performance worsens when the agent communication graph is far from the ideal configuration for those algorithms. Several algorithms have tools to reorganize agents, either by eliminating edges or loops, or assigning hierarchy/weights to agents but not all of them have techniques for dealing with dynamics concerning the agent network and/or agent addition/deletion.

[2]k-RTF is a RTF problem where $k \geq 0$ is a parameter.

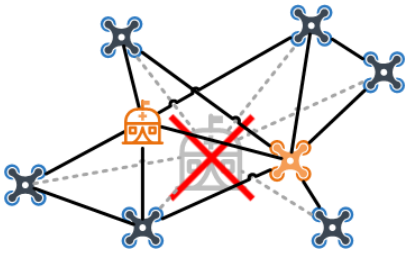


Fig. 2: Agent negotiation worsens as the graph becomes dense or loses some property needed for the distributed task allocation algorithm.

Example 1. *Fig. 2 continues the rescue response example with a recoverable team, where an auxiliary camp and a drone replace the lost base camp. Although the rescue team has guaranteed the goal efficiency, its inclusion provokes the communication graph to become from sparse to dense. Considering the problem complexity in addition with the density of the communication between agents, time performance can be severely affected, as shown in Section V, incapacitating the agents to accomplish tasks in the worst case.*

From a resilient point of view, the current TF model possesses the ability of assembly resistant and recoverable teams against disruptive events but lacks of such social welfare modelling. *Stabilisability* definition represents an update to model all these implications. With all this, a resilient team, either robust or recoverable, can be also defined as stabilisable if it maintains certain properties in its initial and possible future team states, based on its communication network. In the following sections, we demonstrate how stabilisability anticipates decisions that guarantees the inter-agent organisation once the team is assembled and, at the same time, relaxes the communication effort during further reorganisation when such team changes.

III. Stabilisable Team Formation

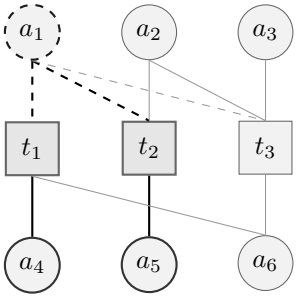
Stabilisability emerges as the condition for a team T to guarantee or minimize the organisation effort considering the the initial organisation of a selected T and any possible future T' provoked by disrupting events.

Definition 5 (STF problem). *A Stabilisable TF problem is the following tuple $STF = \langle RTF, SC\rangle$ where $SC = \langle \delta, l\rangle$ is the Stabilisability Configuration defined by:*

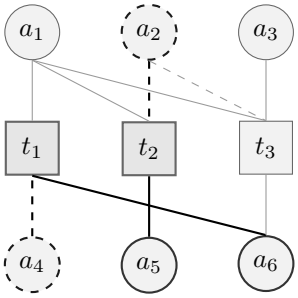
- **Network representation** δ: Data structure for representing the agent network.
- **Graph function** l: $2^A \rightarrow \mathbb{R}^+$ determines the stabilisability according to a graph function or property applied over δ.

We denote by Θ the *communication state set* to be the set of all sub-graphs $\theta \in \Theta$ of a team $T \in A$. That is, all possible efficient teams T' generated from the combination of $T \setminus T_{rem} \cup T_{rec}$ where k agents have been removed from T and agents from T_{rec} have been added.

a_i	α_i	$f(a_i)$	$h(a_i)$	$\delta = \text{neighbors}(a_i)$
a_1	$\{t_1, t_2, t_3\}$	5	2	$\{a_2, a_3, a_4, a_5, a_6\}$
a_2	$\{t_2, t_3\}$	3	1	$\{a_1, a_3, a_5, a_6\}$
a_3	$\{t_3\}$	2	1	$\{a_1, a_2, a_6\}$
a_4	$\{t_1\}$	1	1	$\{a_1, a_6\}$
a_5	$\{t_2\}$	2	3	$\{a_1, a_2\}$
a_6	$\{t_1, t_3\}$	1	2	$\{a_1, a_2, a_3\}$



a) $\langle 3, 2, 2\rangle$-recoverable team T_1=$\{a_4, a_5\}$, T_{rec}=$\{a_1\}$



b) $\langle 3, 2, 2\rangle$-stabilisable team T_2=$\{a_2, a_4, a_5, a_6\}$ bounded by the condition of no neighboring agents

Fig. 3: STF problem composed by 6 agents, 3 tasks, deployment and recovery costs as f and h respectively. Teams obtained w.r.t a goal $G = \{t_1, t_2\}$.

Definition 6 (graph bounding)**.** *Θ is said to be bounded by a value v from l if $\forall\theta \in \Theta$, $l(\theta) \leq v$*

Namely, Θ represents all possible states on how agents communicate within the assembled team, and l represents a network requirement applied for such state set. Accordingly, the team is defined as stabilisable if any network representation is bounded by a value in l.

Definition 7 (k-stabilisability)**.** *T is $\langle c_d, k, c_r, l\rangle$-stabilisable w.r.t. G if T is $\langle c_d, k, c_r\rangle$-recoverable and Θ is bounded by l.*

Fig. 3 shows a recoverable and stabilisable teams for a STF problem. T_1 and T_2 possess initial teams with cost 3 that if they are removed ($k = 2$), the recovery team take action with a cost of 2. Both teams are just efficient (0-robust) w.r.t to G but the addition of T_{rec} restores the efficiency any case.

Example 2. *Suppose that in our scenario the drones from the team must fulfil the tasks by their own once they leave the outpost. That is, agents must be capable to finish the task allocation problem being isolated from each other once they go into action. To guarantee that, we are looking for a team where no inter-communication is needed at all, including for those standby agents if any fail. For simplicity, we assume that 2 agents are connected and exchange messages if they have tasks in common (for example: negotiation, resource sharing, conflicts resolution, etc.). In this case, T_2 provides such stabilisability according to l : |neighbours(drone)| = 0, meaning that there is no edges between agents in action. Subgraphs from the communication state set $\Theta^{k=2}_{\delta(T)}$ between the teams are different: T_1 initially does not have tasks in common $(\theta_{\{a_4,a_5\}})$ but the selected $T_{rec} = \{a_1\}$ implies a negotiation of one task when an agent is removed, either t_1 with a_4 $(\theta_{\{a_1,a_4\}})$ or t_2 with a_5 $(\theta_{\{a_1,a_5\}})$. On the other hand, agents in T_2 are completely independent and can be allocated to tasks without further conflicts with other members.*

Figures 4 and 5 extend the example, where both teams have the same overall cost and $k = 2$. T_1 provides stabilisability with regard to l that obliges not sharing tasks between agents. Figure 4 shows all possible states of T_1 in order to accomplish the goal, that is the communication state set represented by $\Theta^{k=2}_{\delta(T)}$. The team initially does not have tasks in common $(\theta_{\{a_4,a_5\}})$ but the selected $T_{rec} = \{a_1\}$ implies a negotiation of one task when an agent is removed, either t_1 with a_4 $(\theta_{\{a_1,a_4\}})$ or t_2 with a_5 $(\theta_{\{a_1,a_5\}})$. On the other hand, agents in figure 4 have no neighbors under any circumstances, so they are completely independent and can be allocated to tasks without further interaction.

Proposition 1 (STF Generalisation)**.** *Any k-stabilisable team T is also k-recoverable.*

Proof. (by reducing to absurd): Be R and S the sets of all possible recoverable and stabilisable teams assembled from A respectively. So, $\exists T \in (\neg R \wedge S) \Rightarrow T$ is $\neg$¡c_d, k, c_r¿-recoverable $\wedge$ ¡c_d, k, c_r, l¿-stabilisable. A recoverable team has no stabilisability constraint, so SC is the lowest restrictive configuration possible where any agent is capable to communicate with any other into the team. That is, some δ bounded by a complete graph $l = K_{|T|}$ as L, so T is $\neg$¡c_d, k, c_r, L¿-recoverable. By definition 4, k and costs are preserved, so $T \rightarrow \neg L_T \wedge l_T$. From a graph perspective, bound functions $L = (T, E_L)$ and $l = (T, E_l)$ are general graphs where T and E represents the agent set as vertexes and the set of communication edges. The number of edges in a complete undirected graph is $|E_L| = |T|(|T| - 1)$ and any other graph with T vertexes contains at least a subset of such edges $E_l \subseteq E_L$. So the contradiction emerges when $\nexists T \rightarrow l_t \nsubseteq L_t$ and the generalisation is proven. □

Stabilisable teams can be described as a subset of the recoverable solution set in a TF problem. The strength of stabilisability through agent networks lies in avoiding those teams whose evolution or reconfiguration is not feasible in a practical sense. In critical real-world scenarios, agents should be aware not only in forming teams, but also manage interaction, avoid conflicts, and communication overloads. These circumstances become even more remarkable in scenarios subject to unforeseen events that forces agents to be reorganized.

A. Stabilisability through Agent networks

As mentioned above, stabilisability is defined from a graph-perspective as agent networks. As graphs, there are properties that characterize certain network organisations or topologies suitable for task allocation algorithms with MAS. Figure 6

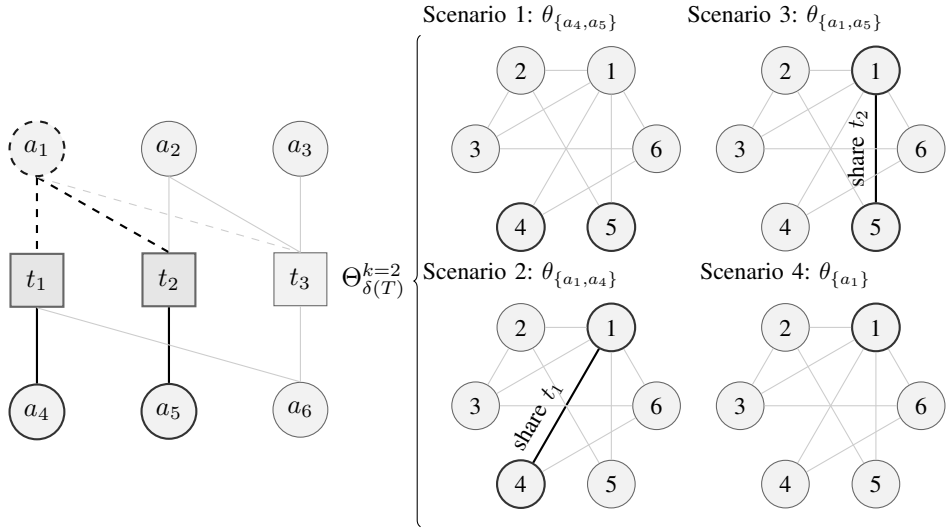


Fig. 4: Network representation δ: $neighbors(a_x)$ from recoverable team T_1 in figure 3.a, where two agents are connected if agents have tasks in common. $\Theta^{k=2}_{\delta(T)} = \{\theta_{\{a_4,a_5\}}, \theta_{\{a_1,a_4\}}, \theta_{\{a_1,a_5\}}, \theta_{\{a_1\}}\}$

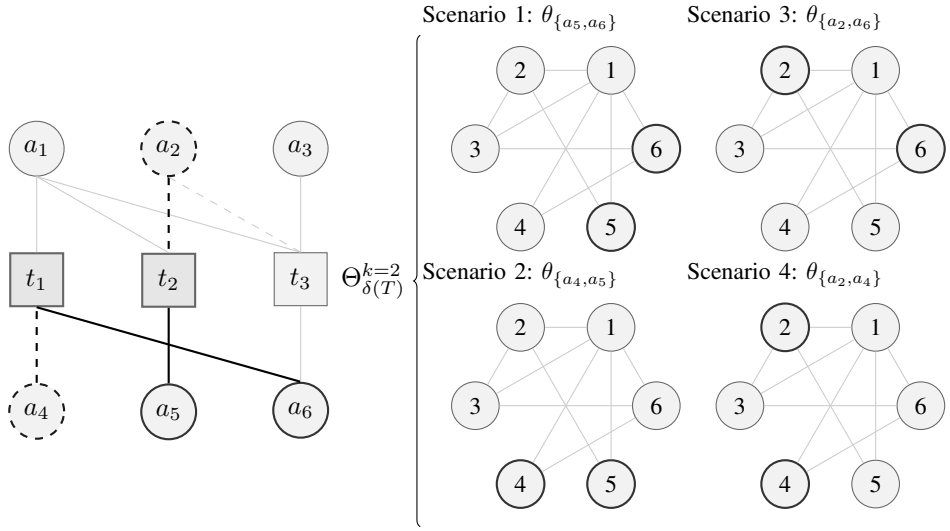


Fig. 5: $\langle 3,2,2,l\rangle$-stabilisable $T = \{a_5, a_6\}$, $T_{rec} = \{a_2, a_4\}$ and $l : |neighbors(a_x)| = 0$ from table in figure 3.b. l is the stabilisability constraint as a naive function that forces no task sharing between agents (equivalent to graph density of the team as 0 with no edges). $\Theta^{k=2}_{\delta(T)} = \{\theta_{\{a_5,a_6\}}, \theta_{\{a_4,a_5\}}, \theta_{\{a_2,a_6\}}, \theta_{\{a_2,a_4\}}\}$

shows different well-known properties used in this paper to formalize the graph function l:

- *l as Density function*: determines the distance between the number of edges in the graph and the potential number of edges. This measure is intuitive for representing how dense or sparse is the agent network, from a full connected team (equals to 1) or a complete agents isolation (equals to 0).
- *l as Clustering function*: related to graph density, it is defined as the coefficient of the number of triangles and the number of connected triples of vertices or triads in the graph [New03]. This measure is suited to measure how far the graph is from being a tree (equals to 0) or a complete graph (equals to 1).
- *l as Arboricity bounding function*: Albertson et al. [AH96] introduced the concept of bounding functions for graph families. A function *b bounds graphs from above* if there is an infinite graph family where $b(|Vertexes_G|) = |Edges_G|$ holds for any graph G and $b(|V_H|) \geq |E_H|$ for any subgraph H of G. We apply the bounding function $b(V) = n(V-1)$ to represent the *disjoint union of n trees*, used by Haas to characterize Nash-Williams arboricity [Haa02].

Computing stabilisability depends on the agent graph representation and the property to measure. Graph functions are faster depending on different representations where algorithm

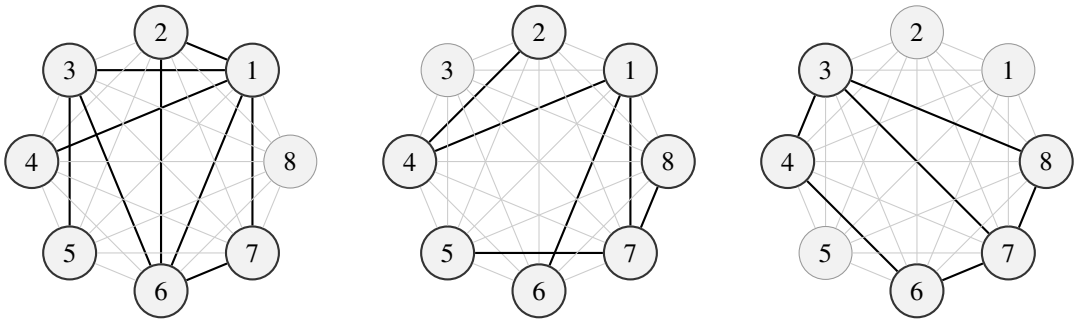


Fig. 6: Examples of agent network from teams bounded by different l_n based on different graph functions: density $l_d \leq 0.5$ (here, 0.43), clustering $l_c = 0$ (tree graph), and 2-disjoint tree bounding function l_b.

runtimes depend on $|V|$ and $|E|$. E.g. an Adjacency Matrix is a $|A| \times |A|$ data structure such that the element $\delta_{ij} \neq 0$ if agents A_i and A_j are connected. Edge and adjacency lists are other well-known representations for graphs.

To model the graph network and defining the stabilisability condition through the graph function is a powerful tool for approximating TF problems to more realistic scenarios. In concrete distributed problems, teams preserve their performance only if preserve their topology. Generally, MAS-based algorithms for solving distributed optimisation problems organize agents on trees or pseudo-trees when an optimal solution is needed [FPY16]. However, our approach enables team to fit also with other graph representations, *i.a.* lattices, cycles, grids, or planar graphs. Indeed, we can provide some flexibility with regard to the desired graph representation depending on how the stabilisation parameter has been defined. That is, how a team is farther or closer from a concrete representation.

For a better comparison during experimentation and further discussion, we selected the aforementioned functions assuming that the obtained teams must perform subsequent task allocation problems with different algorithms characterized by their agent network representation.

IV. Algorithm

Given a STF problem instance and an integer k, the aim is to find all the teams T efficient w.r.t a set of tasks from a goal G, recoverable with the minimum cost, and stabilisable w.r.t a graph property, for all possible deprivation of k agents from T. Our algorithm is a first approximation for solving resilient TF problem using the DCOP framework. A DCOP is defined by the tuple $\langle A, X, D, C, \lambda \rangle$ where A, X, D, and C are the set of agents, variables, domains for variables, and constraints as cost functions respectively, and λ is the mapping function from agents to variables. Thus, the goal is to find a complete variable assignment that satisfies all the cost functions and, at the same time, such cost is minimized (or maximize as utility functions) [FPY16]. In TF context, the domain corresponds to the inclusion or not of an agent to a team (3 states: not in team, in initial team, or in recovery team), each agent only decides over its own inclusion (usually, one variable per agent), and the constraint set refers to the different rules: costs, accomplishable tasks and stabilisability.

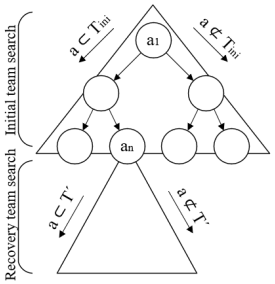


Fig. 7: SBB-ST based on DFS tree from distributed BnB.

A. SBB-ST

Our algorithm, namely *Synchronous Branch-and-Bound for Stabilisable Teams*, is based on *SyncBB* algorithm [HY97]. By the definition of Branch-and-Bound algorithm, SyncBB and SBB-ST are complete DCOP algorithms that guarantee the optimal solution. The main idea is the partial assignment propagation of variables from visited to non-visited agents. The search space in SyncBB is pruned when sub-optimal solutions are found. Our algorithm differs from its predecessor by expanding the search space to find both efficient and stabilisable teams, using different agent ordering by costs. Algorithm 1 summarizes the SBB-ST outline, and Fig. 7 depicts SBB-ST as a DFS tree in pre-order, updating the cost at the leaf in each phase, and then returning the minimum. According to branch-and-bound (BnB) definition, descendants of a node can be ignored if the cost goes below or above lower and upper bound, respectively (same with k). Essentially, agents behave as a BnB search in a distributed fashion as nodes, waiting for a message (line 6), and sending the output to its neighbour (parent or children, line 29). Non-leaf agents add partial information from parent, e.g., using vectors (lines 8, 9, 14 and 15), and the leaf check stabilisability and computes the solution (lines 16-19). SBB-ST performs two consecutive searches: the first one looking for an initial efficient team, and the second one for the recovery team up to k removed

Algorithm 1 SBB-ST (SyncBB for Stabilisable Teams)

Input: A STF problem $\langle A, P, f, \alpha, h, SC \rangle$, a goal G, and k
Output: All k-stabilisable teams

Let *max_cost* $\leftarrow \infty$, *teams* $\leftarrow \emptyset$, *possible_teams* $\leftarrow 3^{|A|}$
Let $T_{ini} \leftarrow \emptyset$, $T_{rec} \leftarrow \emptyset$, $T' \leftarrow \emptyset$,
Let *agent_mode* $\leftarrow$ EFFICIENT
for all agents in A **do**
  while *agent_mode* $\neq$ FINISHED **do**
    agent_mode $\leftarrow$ *get_message(agent_from, data)*
    if *agent_mode* $=$ EFFICIENT **then**
      Update data from current T_{ini} and accomplished tasks with the partial knowledge of the agent
      Update the accumulated cost from T_{ini} and compare with *max_cost* threshold
      if agent has full knowledge from T_{ini} **and** T_{ini} is efficient, costly and stabilisable **then**
        agent_mode $\leftarrow$ RECOVERY
      end if
    else if *agent_mode* $=$ RECOVERY **then**
      Update data from current T_{rec}, T' and accomplished tasks with the partial knowledge of the agent
      Update the accumulated cost from current T' and compare with *max_cost* threshold
      if agent has full knowledge from T_{ini} and T_{rec} **and** T' is still efficient, costly and stabilisable **then**
        Append $[T_{ini}, T_{rec}]$ into *teams*
        if accumulated cost ¡ *max_cost* **then**
          max_cost $\leftarrow$ accumulated cost
        end if
      end if
    end if
    discarded teams $\leftarrow$ sub-optimal teams from current agent and its BnB descendants
    possible_teams $\leftarrow$ *possible_teams* - *discarded teams*
    if *possible_teams* $= 0$ **then**
      agent_mode $\leftarrow$ FINISHED
    end if
    Update *next_agent*
    send_message(agent_mode, next_agent, data)
  end while
end for
return *teams*

agents. The efficient search space is traversed to find those initial teams able to accomplish the set of tasks in G. Agents send the partial team assignment from parents to children, where each agent concatenates the information with its local knowledge, i.e., the variable assignment, the accumulated cost, the task accomplishment, and the team network representation. Once the leaf agent possesses the full knowledge referring the initial team assignment, it computes the team efficiency and the stabilisability according to l.

The recovery search starts when an efficient team is found. During this search, agents are revisited and its information updated depending on their previous assignment: an agent can be assigned as recovery agent only if it is not member of the efficient team. Agents that are members of the efficient team can only be deprived as part of T_{rem}. Search space is pruned in those sub-trees where $T_{rem} > k$. Thus, all possible team states $T' = T \setminus T_{rem} \cup T_{rec}$ are found. Similarly, once the leaf possesses the full knowledge referring both initial and recovery assignments, it checks if the efficiency and the stabilisability is still conserved. A team assignment is solution when all combinations of $T \cup T'$ satisfy all constraints.

Figure 8 shows an example of the search space as a binary tree and how SSB-ST traverses it during recovering phase. In this example a_4 is the leaf agent in the efficient phase. Once a_4 assigns the value of variable that controls and checks the efficiency, recovery mode starts with a_1 as the highest priority agent. a_1 is part of the initial team, so it updates its partial assignment as a deprived agent from the initial team or not. Then, a_1 updates the accumulated cost and accomplishable values, according to the assignment, and sends the information to a_3 as the next priority agent. a_3 was not assigned as the initial team, so it can be part of the recovery team. Thus, it computes the partial assignment as an agent in the recovery team or not in the same way. Given $k = 1$ means that only 1 agent can be deprived form the initial team, so only a_1 or a_4 can be assigned as removed. If a potential agent to be deprived exceeds k then the solution is sub-optimal and the sub-tree is pruned. Similarly, the sub-tree is pruned if the accumulated cost exceeds the threshold of previously found solutions. The leaf agent during recovery team a_4 receives the recovery vector, fulfils the partial assignment, and checks all the constraints. In this example, the 1-stabilisable team $[1, 0, 2, 1]$ satisfies the problem with a total cost of 7.

Concerning completeness, SBB-ST as a search algorithm based on DFS, where agents are ordered based on heuristics attached to different agent costs, functions, or constraints, and the tree height is determined by the number of agents. By definition, DFS is complete if the search tree is finite, so it will come up with a solution [RNC+96]. In the TF optimization model, all Resilient constraints attached to k are defined as hard constraints (robustness, recoverability, and stabilisability), so they must be satisfied by any feasible solution to the model [FPY16]. Said that, SBB-ST completeness over a STF problem is preserved.

The stabilisability constraint has some implications during search pruning: In the same way that the sub-tree search is pruned when the k constraint is no longer satisfied, the recovery search phase can be omitted when the initial partial team does not satisfy the graph bounding constraint. According to definition 6, a solution must satisfy such constraint for all possible graphs emerging from such team. In other words, a team is not a solution when there is only one configuration of agents from such team that does not meet the stabilisation constraint. In this way, when an initial team is sub-optimal, all its subsequent possible configurations are also sub-optimal.

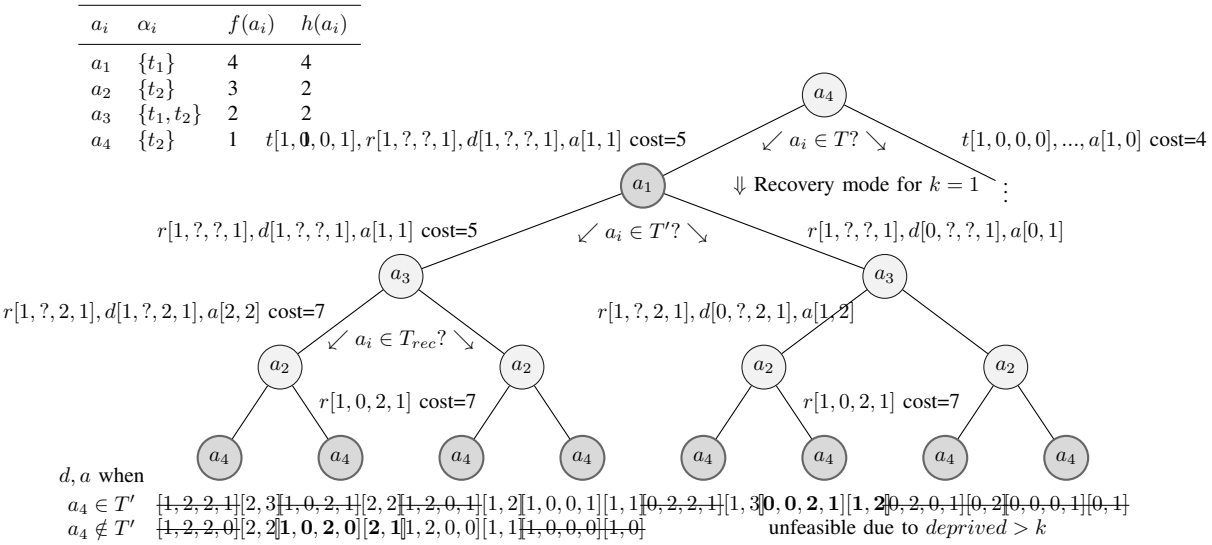


Fig. 8: Recovery phase example. Variable domain is defined by 0 =*no team assigned*, 1 =*in initial team*, 2 =*in recovery team*. Assignment data vectors for the initial $[t]$eam T', $[r]$ecovery team T_{reco}, T' with agents $[d]$eprived according to k, and $[a]$ccomplished tasks by agents in T' according the table. Solution: $\langle 5, 1, 2\rangle$-recoverable $T = \{a_1, a_4\}, T_{rec} = \{a_3\}$ w.r.t. G.

1) Algorithm Complexity: SBB-ST is based on *Synchronous Branch and Bound* algorithm (SyncBB), a well-known algorithm for DCOPs [HY97]. Operations from DCOP algorithms grow exponentially to compute solutions, either by the domain size or the number of agents, among others [FPY16].

Proposition 2. *k-STF problem is NP-hard*

DCOP and TF are NP-hard problems by definition as optimisation problems, but NP-complete as decision problems (that is, if a solution exists). As explained in Sec. III, STF is a generalisation of RTF. Using k as a parameter, and transforming k-STF onto a DCOP problem proves that the complexity remains [MSTY03], [OSC+15], [DSOI18].

Proposition 3. *SBB-ST performs $3^{|A|}S$ operations, where S is the number of operations from $O(l(|A|))$.*

SyncBB performs $d^{|A|}$ operations [HY97], where d is the size of the largest domain in the DCOP. In TF, domain corresponds to the assignment of an agent into a team (initial, recovery or none). The leaf agent in SBB-ST computes all possible assignments and checks the stabilisability through l. S is attached to the stabilisability complexity and performance depends on both the graph representation and the graph function to be applied. From the graph properties defined on figure 6, density and arboricity functions are $O(|T|)$ to count the number of edges if δ is represented as an adjacency list but $O(|T|^2)$ as adjacency matrix, whilst clustering is $O(|T|^{2.8})$ if we use the Strassen algorithm to calculate the number of triangles in the team network.

Proposition 4. *If $k = 0$, SBB-ST performs $2^{|A|}S$ operations.*

As explained above, in the efficient phase agents share the partial assignment for the initial team, so agents decide if they are into the initial team or not. Further, the recovery search tree is pruned by the k-constraint, so sub-trees that have more than k agents removed from are not visited.

V. Experiments

Our experiments are focused on testing the following hypotheses:

- To solve Resilient Team Formation problem requires less computation effort when the obtained team is stabilisable and recoverable or robust rather than just recoverable or robust.
- Given a subsequent Task Allocation problem to be solved by an agent team obtained from a previous solved TF problem, its resolution can be relaxed and its computational effort is reduced when the team has been defined as a stabilisable team, rather than just a recoverable or robust team, assuming a set of possible emerging dynamics concerning inter-agent communication resulting topology and agent deprivation.

Figure 9 sketches the experimentation process. First experiment uses SBB-ST to obtain recoverable teams (RTF) according to Demirovic and Okimoto proposals and stabilisable teams (STF) [OSC+15], [DSOI18]. As explained in Section III, RTF teams are equivalent to STF with no stabilisation constraint. Second experiment solve the graph colouring problem as TA problem to compare the performance between both team sets using different well-known DCOP algorithms. The constant k represents the dynamic of agent deprivation, so the second experiment include all the scenarios emerging from an obtained team where $0 <= n <= k$ number of agents are not available to solve the problem. From the figure example, a 2-resilient team with 5 agents emerges different TA problems,

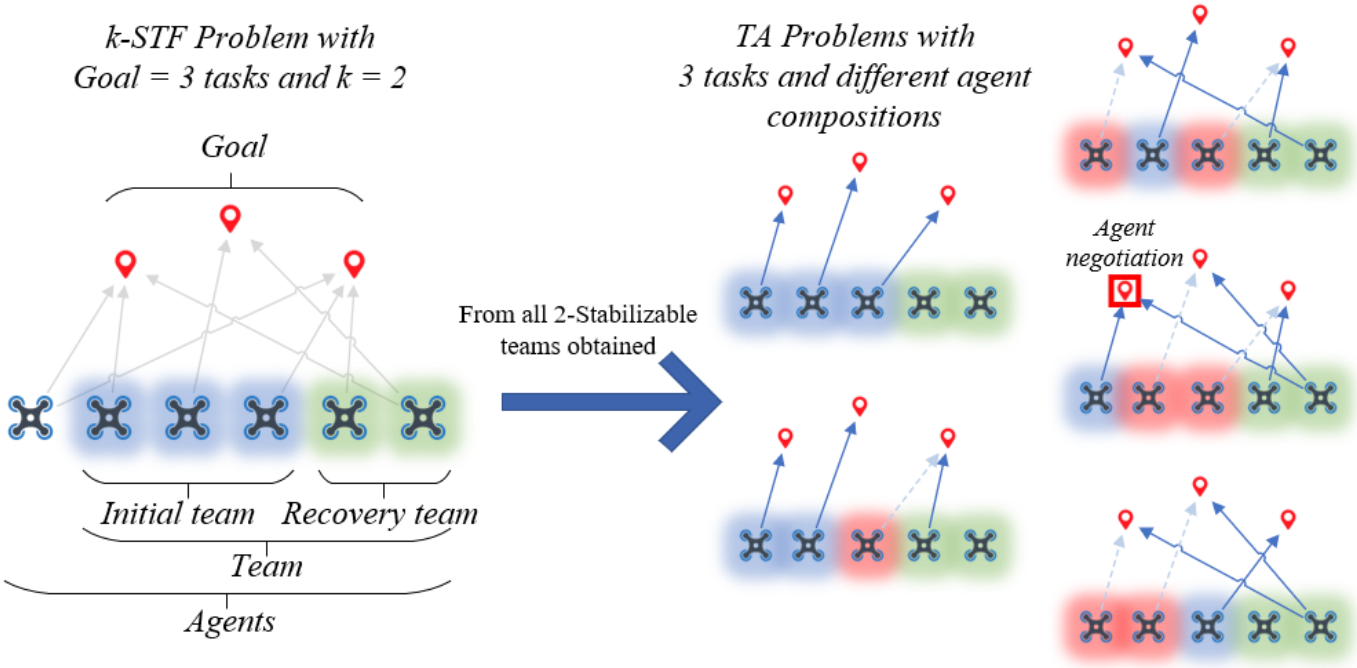


Fig. 9: Experiments configuration: First experiment consists on solving the STF problems given *A, k, tasks, costs, allocations, network, and graph function* with both allocation and network maps being different and random (allocate a team to the goal with initial and recovery sub-teams). Second experiment consists on measure performance from all possible TA problems as graph colouring problems for these teams (allocate agents to tasks individually, with resulting communication constraints) obtained on experiment 1.

depending on how many agents from the initial team (0, 1, or 2) cannot be part of the allocation.

In a nutshell, we aim to compare both computation time and performance from teams obtained between our proposal and its predecessors at different levels: first, the allocation between a team and a mission as a set of tasks. Second, the allocation between team members and single tasks. Experimentation also aims to show how teams can be assembled according to some topology criteria. Accordingly, we aim to observe how such prior criteria support teams to relax distributed task allocation problems and improves team performance, assuming that teams are highly dependent to their topology.

Concerning the experiment setup, we considered TF instances from 5 to 25 agents with a goal of 10 tasks, with 100 instances per agent set size. This allows us to compare run times in the optimal version of Demirovic and Okimoto. Larger instances have been discarded because the incorporation of heuristics to improve scalability but to the detriment of solution optimally is outside the scope of our study. Accomplished tasks per agent α_a are generated uniformly at random from 1 to 8 tasks [OSC+15], and costs f and h are equal according to the experiments from Demirovic et al. [DSOI18]. For our experiments, DFS ordering for BnB solving is also random. Regarding stabilisability configuration candidates, $\delta = |A| \times |A|$ is the Adjacency Matrix[3] from α, and l is defined from different graph functions described in Sec. III.1. A set of discrete threshold values for l were selected: 0.05, 0.4 and 0.7 for density and clustering functions [Gd08], and $\{3,2,1\}$-*disjoint tree union* bounding function for arboricity [Haa02]. We compared our proposal (STF) with its predecessors from Demirovic et al. and Okimoto et al. [OSC+15], [DSOI18] works (RTF), where no stabilisability configuration is applied. The number of instances on the second experiment is equal to the number of teams obtained per instance on first experiment, plus the number of teams that can be assembled according to k deprivation. This tends to $\sum_{k=0}^{2} 100\binom{|T|}{k}$ where $3 <= |T| <= |A|$. For example, a solved $k = 2$ instance provides one team possesses as many configurations or scenarios as possible combinations of teams with 0, 1 or 2 removed agents (see figures 4 and 5), resulted on 100^3 instances per stabilisable configuration approximately. Tests were performed on an Intel Xeon 6230 (20 cores@2.1 GHz, 32GB RAM). To compute both recoverable and stabilisable solutions, we developed SBB-TF algorithm using JADE, a Java framework for MAS deployment [BCG07]. For TA problem simulation and DCOP benchmark, we used FRODO, a Java framework for distributed combinatorial optimisation [LOS09].

[3]For simplicity, non-directed/weighted graphs are assumed in δ.

A. Time performance comparison

SBB-ST found both recoverable and stabilisable solutions from generated TF instances. Figure 10 summarizes RTF time performance (red line) against different STF configurations.

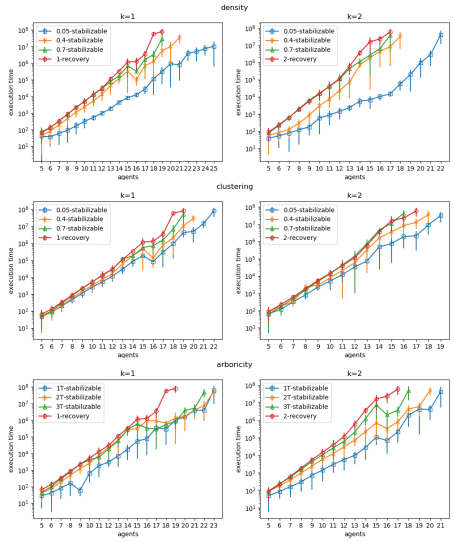


Fig. 10: Average and SD results from TF instances with 5-25 agents in milliseconds (100 instances per agent set size). Time performance is compared between recovery teams and stabilisable teams with different network configurations.

Overall, times obtained by STF with low values are significantly improved as opposed to RTF. Regarding k-parameter implications, for each feasible team T, one needs to consider every possible removal of k agents and compute its cost and stabilisability. The number of possible T' combinations is exponential with respect to k so computing teams remains expensive. Demirovic et al. [DSOI18] defined a set of heuristics in his proposal to obtain a drop in the number of operations at the expense of losing the completeness. SBB-ST shows how such drop can be improved as a complete algorithm if the stabilisability computation remains inexpensive.

For our experiments, DFS ordering for BnB solving are random in order to obtain and discuss certain behaviours related with the search pruning based on constraint satisfaction. Given the same timeout for solving instances, the recovery approach is not capable to solve ¿19-agent for k=1 and ¿18-agent for k=2 instances respectively, whilst the stabilisable constraint allows the resolution of larger instances. This phenomenon is caused by the search space pruning attached to stabilisation constraint when it is more restrictive than the recovery cost minimization constraint. Stabilisability with bounding functions near to sparse graphs relaxes the search regardless the topology, to such an extent that graphs 1-connected graphs and less (from unconnected sparse graphs to tree graphs) greatly prune the recovery phase, which practically halves the height of the DFS tree. In combination with this, the constraints of cost and especially k allow to reduce the height of the DFS tree during the recovery phase ($k = 0$ prunes the recovery search in the same fashion because there is no recovery team). As k grows, the more the tree is traversed in height, so it is inferred that the graph function works as a maximum depth of the DFS tree based on the connectivity of the active agents from both sub-teams. This limit is exceeded only when a solution has been found, since it is the last agent in the hierarchy that has the complete assignment. Therefore, we can state that the pruning of the search space is inversely proportional to the connectivity attached to the graph function.

This behaviour has some implications regarding scalability, where graph functions reduce the exponential time growing attached to k in a greater or lesser extent, up to several units of magnitude for most restrictive configurations. As expected, time performance and scalability are deteriorated as bound restrictions from stabilisability get relaxed. This is because the computation that is required for obtaining the number of edges or the number of triangles from $\delta(T)$. The use of adjacency lists for graph representation improves the computation complexity ($O(|T|)$). Another solution for this in larger instances is to obtain such values offline, or the application of heuristics at the expense of guaranteeing the optimal solution. Overall, performance on larger problems becomes significantly better where high connected teams are not comprised by a large number of disjoint trees or preserve a low graph connectivity.

Discussing about these results, this experiment aims to compare STF model with its predecessors by Demirovic et al. and Okimoto et al. [OSC+15], [DSOI18] in order to evidence the issues attached to the scalability and the high dimensionality of the problem. For this reason, we consider that the most legitimate way to conduct the first experiment was to replicate under the same conditions as its predecessors, within the scope of our resources. Assuming that generated instances have a random component, they are generated according to the literature and have the same restrictions, so the results are legit and comparable. One aspect that should not be ignored in these problems is obtaining an adequate solution (optimal or not) in a reasonable time. Once the usefulness of the organisation between agents through stabilisation is evidenced, the next step could be the study and implementation of heuristics and asynchronous computing capable to obtain faster and even more scalable solutions.

B. Stabilisability impact on Team Task Allocation

This experiment shows how RTF and STF teams manage their inter-agent organisation on subsequent Task Allocation instances. To do so, we modelled each team obtained from the previous experiment as a DCOP where the agents must allocate the tasks individually from G. Concretely, teams solve the graph colouring problem built from the constraint graph based on agent capabilities from α_T, and $\delta(T)$ of selected teams. Agents must allocate tasks as colours where connected agents represent a task in common. A solution is found when a subset of agents is coloured differently with all tasks. Given a team, this experiment considers all possible scenarios according to k representing the dynamic of agent deprivation and recovery agents entering into action. Thus, each TF instance generates, in turn, as many TA instances as the number of teams found plus their different team states T' according to k possible deprived agents.

Fig. 11: Medians and confidence intervals obtained from task allocation problems solved by obtained RTF and STF teams w.r.t tasks from instances with 3-24 agents. The number of instances per graph line tend to 100^3, according to all possible team scenarios attached to k values. Line colours represent different bounding values for each stabilisability configuration: 0.05, 0.4 and 0.7 for density and clustering, 1,2,3-disjoint tree for arboricity, and non-stabilisable teams (recovery).

We selected a set of well-known DCOP algorithms from the literature differentiated by their agent organisation: DPOP as depth-first search pseudo-tree ordering [PF05a], Max-Sum as a factor graph (tasks-agents bipartite graph) [FRPJ08], and MGM as a constraint graph [MPT04]. These algorithms are implemented by FRODO framework for benchmarking and allows a good visualisation on how a prior team restriction on communication relaxes the computation of TA problems based on DCOP problems.

Figure 11 shows the measures of solved graph colouring instances from obtained RTF and STF teams. The red line represents recovery teams from Demirovic et al. where no stabilisation configuration was considered on experiment 1. Blue, orange, and green lines represent stabilisable teams with concrete configurations based on the graph properties described in Section III.1. All teams solved their respective TA instances using the 3 DCOP algorithms.

Meeting our expectations, stabilisability fulfils the relaxation of intra-team organisation. Regardless of k removed agents and the inclusion of any agent from T_{rec}, teams that are bounded by any stabilisability configuration also bounds computation and communication effort. Recoverable teams in figure 11 shows the medians of different measures from the candidates subject to grow exponentially according to its operation: DPOP complexity grows exponentially with regard to the message size depending on the induced width of the pseudo-tree [PF05a], while Max-Sum and MGM in the number of messages depending on the number of agents, neighbours, and running iterations needed to solve the instance [MPT04], [FRPJ08].

With no exception, worst results arise from recoverable teams, whilst stabilisable teams not only improve the characteristics of these algorithms that grow exponentially over time, but also lowers the magnitude scale in some cases. This behaviour is expected when we prioritize in those teams whose graph representation allows a relaxation of the most expensive computation between DCOP agents, either the exchange/size of messages, their reorganization in certain structures such as pseudo-trees, or loops removal, among others. Our results show how stabilisability based on density and clustering reduce the average computation in a same fashion: for a bound of 0.7 on density and clustering, both message size and number decrease with regard to recovery instances with all DCOP algorithms. For bounds 0.4 and 0.05 message size grow for DPOP becomes linear and the number of exchange messages between agents is greatly reduced. We observe that teams tend to possess disconnected communication graphs or even isolated agents that they do not need to negotiate with tasks in common, so the assignment becomes more trivial as its configuration gets more restrictive. Eventually, the same problem can be solved by a 24-agent recoverable team that needs 10 megabytes of memory per agent and tens of thousands of messages to compute a solution, or a stabilisable team with the same number of agents but with no more than 10 kilobytes of memory per agent and less than 1000 messages. Regarding arboricity, we observe that all bounds transform the exponential growing of all algorithms to linear. That is because team connectivity can be characterized through their number of possible trees that can be partitioned. In other words, n-TREE connected teams tend to be n-edge-connected. Intuitively, same configurations impact differently on the DCOP candidates. Restrictive stabilisation based on clustering and arboricity is useful for those algorithms where agents are connected as trees or pseudo-trees, but including inefficient tree shapes, such as the tree width for DPOP. However, the cases were marginal and the computation effort for these teams were compensated with the fact that their agent network is practically a tree, so the algorithm execution is relaxed because costly operations such as agent tree ordering and loops removal can be obviated. Indeed, having algorithms where communications loops are avoided involve a direct impact in their optimally is even more powerful, such as the case of restrictive stabilisation for the Max-Sum family. By definition, Max-Sum is an incomplete algorithm but complete if there are no loops in the network graph [FRPJ08]. Thus, all TA instances solved with 1-TREE Max-Sum teams were optimal.

We notice that k values has some implications on resulting team connectivity. As k grows, teams require a larger amount of recovery agents to maintain the stabilisation bound with agents into action. When resulted teams has no recovery agents, k is directly linked with the onset of k-cliques, meaning that agents must be fully adjacent in some degree to ensure the affordability of tasks. As the number of recovery agents grows into the team, cliques on ongoing agents disappear and the connectivity is reduced. Consequently, it becomes difficult or even unfeasible to find stabilisable teams with a larger k and low recovery subset of agents. On the other hand, relaxing stabilisation criteria allows teams with a greater connectivity but at the expense of SBB-ST performance.

Discussing on this experiment and representing instances as task allocation problems, we decided to use the DCOP formalism because is a well-known framework for distributed problem solving, and the proximity of TF to related problems such as coalition structure generation [RMWJ15], [CFMR14]. DCOP literature is extensive and can be classified on different categories: graph representation, methodology, completeness, and algorithm families, among others. We selected the most representative candidates given the taxonomy from Fioretto et al. [FPY16] with regard to their graph classification. In order to amplify the impact of our results, we selected the candidates not exclusively by its network characterisation, but also for their popularity and further variants in the literature. After conducting several experiments with some inherited and novel algorithms, results seem to be correlated with their predecessors, because of the fact that experimentation is focused on agent team structure/topology, instead of solely improving the performance of concrete DCOP algorithms. Accordingly, we avoid developing novel inherited algorithms on FRODO due to this and the enormous effort to be required to implement them from scratch according to our resources.

In any case, results confirm that anticipate decisions regarding intra-team organisation are useful to save resources and

minimize the effort against unforeseen dynamics. Restrictive stabilisation is suited for scenarios where agent communication is expected to be prohibitive, so agent autonomy is guaranteed. In fact, STF gives the possibility of custom network configurations, e.g., cycles, lattices, or planar graphs.

VI. Discussion

Team formation is a suitable starting point for modelling agent-based distributed problems but is far from a profound representation due to its abstraction. Resilience on MAS not only extends these problems, but also approaches more real scenarios capable of representing their dynamics and (unforeseen) events. While definitions related with resilience, stabilization and other terms are not exclusive to MAS, we want to clarify that the terminology used is inherited to the research context and the mentioned papers we work on, without pretending to overshadow their use in other different contexts.

As far as we are concerned, the way that resilience has developed in the TF literature is valuable but not feasible in a practical sense. Forming teams, whether resilient or not, is a NP problem, so the premise that a instance must be resolved in a reasonable time should not be ignored. Secondly, complementing the different resilient characteristics are not exempt from conflicts, so the balancing of these must be considered when applying the model. In other words, it is important to manage when robustness (k), recoverability (T_{reco}), or stabilisability (agent network) need to be enhanced depending on the definition, requirements, and priorities of the problem, while seeking the best possible cost and/or performance. Our intention was to design a cross feature, i.e. to model resilience with a variable capable to be managed in such a way that does not complicate the aforementioned balancing issue.

Team Formation is an interesting approach in MAS planning, but it is even more interesting when put in context with other related problems such as task allocation. As explained before, forming agent teams or coalitions are a subset of distributed problem solving, but in more realistic scenarios the final objective is allocating resources/tasks to perform. To the extent of our understanding, DCOPs are suitable to handle these problems but literature tends to focus directly on the allocation, obviating the benefits of previous team formation.

Our motivation is based on the hypothesis that agents comprised as a resilient team are more capable to reorganize/reallocate resources more agile than agents that were not previously organized. We intend to study in this research how contextualize the resilient team formation is valuable to solve other distributed problems. To do so, our approach is based on divide-and-conquer strategy, in such a way that if an event occurs, it is more productive and generates less impact rethink part of the problem than the whole instance. For this reason, we decided to perform the aforementioned interconnected experiments: firstly, we aimed to solve the inherited time performance issue from recoverable team search. Second, we studied how stabilisable team search relaxes the subsequent task allocation problem regardless its dynamics. Furthermore, our hypothesis includes the fact that resilience can be extended considering not only the agent resources, but also its interaction and dependencies with other agents. The study of the agent network from a graph-theory perspective has been done previously and suits with the definition of stabilisability [PF05b], [Gd08], [CRXH10]. Thus, this definition is useful to cover both sides of our hypothesis.

Whilst there is room for interpreting our results, they must not be considered in a absolutely way. RTF and STF have been implemented using branch-and-bound solvers, with JADE as our agent-based software framework, and FRODO for Task Allocation through DCOPs as benchmark. As expected, obtained data on larger instances and relaxed configurations reflected the same issues with regard to exceeding time execution, so we obviated them. Moreover, we wanted to observe how the graph function allowed us to find the desired teams as close as possible to trees without losing the perspective on the performance. Accordingly, selected values for stabilisability configurations support us to observe the connectivity implications during the SBB-ST operation. To the extent of our understanding, SBB-ST performance worsens more when there is an instance with high connectivity than solving a large instance. So, we can deduce that depending on how dense or sparse is the agent set in our TF problem, we can apply in advance the best data structure and graph algorithms for calculating stabilisability in a reliable way. Regarding the benchmarking on the second experiment, although we are aware of the vast list of current DCOP algorithms in the literature, we decided to base our experimentation and results on those provided by the FRODO software for comparison. After an iteration with different more recent algorithms, we observed that the results were correlated with their predecessors since, although these proposals provide significant improvements, they retain their dependence on the topology of the team of agents. It is important to emphasize that our approach aims to show the importance of establishing a previous configuration in a distributed team before entering a dynamic environment, instead of simply improving some specific algorithms. That is why we consider that DCOPs are a suitable framework for this problem and with a very extensive real-world application.

In any case, our results can represent a first approximation of our expectations and trends in further experiments. We consider that stabilisation is a useful tool to quickly solve resilient TF problems if it is enough restrictive. With SBB-ST we fulfil the purpose of comparing our proposal with its predecessors. It is important to note that the study of resilience on these problems with larger instances is still pending in the literature. To resolve stabilisability with different data structures based on agent connectivity, include other graph properties based on desired networks on teams, and to apply techniques for extending the experiment with larger instances are suggestions to incorporate a different analysis.

VII. Related Work

Schwind et al. [SMI+16], [SOI+13] formalized *Resilience* for constraint-based dynamic systems with agents. In

[SMI+16], *Stabilisability* is described as a feature distinguishable from resilience, where it is just a function related to the system manager/controller. Indeed, Panerati et al. [PSZ+18] modelled resilience for Hidden Markov models excluding stabilisability. In this paper, Stabilisability is modelled not only as an addition to TF problems, but also as a valuable component for resilient systems where the team state is considered. This description is in accordance with the definition of *self-stabilisation* by Dijkstra [Dij74]. Alberola et al. [AJG13] contribution is related with our definition of Stabilisability, where they model transition costs for MAS reorganisation. They propose an agent architecture where costs are attached to modifications on roles and capabilities to provide a set of services. In our context, Stabilisability is defined from a graph-based perspective to guarantee teams performance through agent network topologies. Gaston et al. [Gd08] define *diversity support* as the ratio of possible skill combinations supported by an agent communication graph to allocate the largest set of tasks. From this definition, the paper conducts an study of how diversity support depends on the graph network in dynamic team formation. Faye et al. [FAMO14] introduced Stabilisation on Coalition Formation Problems as a Markov Decision Process, where coalitions were aimed to maintain persistent under certain dynamics, including network topology change, task evolution and stochastic events. This work assumes the impossibility to devise an efficient agent coordination based only on initial knowledge given the unpredictable availability from agents due to dynamic behaviours, whilst our contribution aims to prove how such knowledge, being modelled as a distributed constrained problem, can preserve both efficiency and error-tolerant organisation of agents regardless communication dynamics. Later, TF problem were defined as a bi-objective constraint problem, where both overall cost and k are the values to optimize [OSC+15]. Moreover, TF and similar problems such as Coalition Structure are suited to be faced by the DCOP framework in an online manner suits to face these complex problems [UIY+10], [FPY16].

VIII. Conclusions

This paper deploys and improves the resilience of TF problems extending its implications to task allocation problems by modelling STF. Our main contribution is to demonstrate how stabilisable teams are able to be robust, recoverable, and minimize the intra-team organisation effort that involves deprivation and inclusion of agents, assuring resilience in decentralized systems. Such organisation is modelled from the agent communication point of view, where STF generalizes RTF model from by including the stabilisability configuration. We performed a set of experiments with different configurations using well-known graph properties. In comparison with recoverable teams, our results confirm that stabilisability is capable to reduce the computation on forming teams by in several units of magnitude with the most restrictive configurations. At the same time, we show that stabilisable teams reduce and bound the computation and communication attached to inter-agent organisation when such teams must face subsequent task allocation problems. However, relaxed configurations in conjunction with complex graph functions can be counterproductive for larger instances. To mitigate it, low-complexity functions and offline methods are considered. To compare performance and team organisation from both models, we designed SBB-ST, a branch-and-bound algorithm based on DCOP framework. Stabilisability contributes to resilient TF problems reducing their computation significantly and, at the same time, anticipating team decisions to save resources and minimize efforts during team organisation in dynamic scenarios. We have been aware of several implications on our study and our results have allowed us to not discard our hypothesis and let us continue with further realistic experimentation, from simulation frameworks to ongoing research work with embedded and IoT devices.

Against the approaches given considering resistance and recoverability as the determinant properties for resilience, stabilisability is the only one able to provide a model for inter-agent interaction. As we demonstrated, this improves resilience in concrete MAS scenarios, in such a way that the resistance and reaction against unforeseen events are reinforced.

The strength of stabilisability through agent networks sojourns in avoiding those teams whose evolution or reconfiguration is not feasible in a practical sense. Thus, agents are aware in forming teams and manage better the inter-agent interaction through avoiding conflicts and communication overloads.

IX. Future work

Experiments in simulated real-world scenarios will be conducted to test stabilisability in dynamic environments. Concretely, we will use RoboCup Rescue Agent Simulation platform to assess stabilisability with different task allocation models from literature. Further, we will extend our experiments with users' inclusion in such environments, where dynamics arises from perturbations caused by unexpected events from human interaction with the system. This approach is quite interesting in disaster response, human-agent planning, autonomous collective driving, and other problems related to ubiquitous computing and human-agent collectives [JMN+14]. In addition, we plan to improve the performance and scalability of team formation problems through the study of different approaches based on heuristics, dynamic programming, and parallelisation, inter alia.

Our intention is also to extend stabilisability as a soft or stochastic constraint, study the application of weighted, directed, and other graph properties. Modelling stabilisability to other resilient systems or introducing the resilience framework to other problems are also proposals for further work, e.g., coalition structure generation, swarm intelligence, distributed machine learning, or planning, among others.

X. Acknowledgments

The present work has been possible thanks to the Inoue Laboratory as a member of the NII Internship Program 2019, the Madrid Human-Computer Interaction Lab at UPM,

and the Madrid Super-computing and Visualization Center (CeSViMa).

REFERENCES

[AH96] Michael O. Albertson and Ruth Haas. Bounding functions and rigid graphs. *SIAM Journal on Discrete Mathematics*, 9(2):269–273, 1996.

[AJG13] Juan M. Alberola, Vicente Julián, and Ana García-Fornes. Using cost-aware transitions for reorganizing multiagent systems. *Eng. Appl. of AI*, 26(1):63–75, 2013.

[AMI16] Alexander Andreychenko, Morgan Magnin, and Katsumi Inoue. Analyzing resilience properties in oscillatory biological systems using parametric model checking. *Biosystems*, 149:50–58, 2016.

[BCE+03] Michel Bruneau, Stephanie E. Chang, Ronald T. Eguchi, George C. Lee, Thomas D. O'Rourke, Andrei M. Reinhorn, Masanobu Shinozuka, Kathleen Tierney, William A. Wallace, and Detlof von Winterfeldt. A framework to quantitatively assess and enhance the seismic resilience of communities. *Earthquake Spectra*, 19(4):733–752, 2003.

[BCG07] Fabio Luigi Bellifemine, Giovanni Caire, and Dominic Greenwood. *Developing Multi-Agent Systems with JADE*. 2007.

[CFMR14] Jesús Cerquides, Alessandro Farinelli, Pedro Meseguer, and Sarvapali D. Ramchurn. A tutorial on optimization for multi-agent systems. *Comput. J.*, 57(6):799–824, 2014.

[COSI15] Maxime Clement, Tenda Okimoto, Nicolas Schwind, and Katsumi Inoue. Finding resilient solutions for dynamic multi-objective constraint optimization problems. *ICAART*, pages 509–516, 2015.

[Cou02] D. L. Coutuj. How resilience works. *Harvard Business Review*, 80(5):46–56, 2002.

[CRXH10] Shanjun Cheng, Anita Raja, Jiang Xie, and Ivan Howitt. Dlb-sdpop: A multiagent pseudo-tree repair algorithm for load balancing in wlans. *Proceedings - 2010 IEEE/WIC/ACM International Conference on Intelligent Agent Technology, IAT 2010*, 2:311–318, 2010.

[Die17] Thomas G. Dietterich. Steps toward robust artificial intelligence. *AI Magazine*, 38(3):3–24, 2017.

[Dij74] Edsger W. Dijkstra. Self-stabilizing systems in spite of distributed control. *Communications of the ACM*, 17(11):643–644, 1974.

[DSOI18] Emir Demirovic, Nicolas Schwind, Tenda Okimoto, and Katsumi Inoue. Recoverable team formation: Building teams resilient to change. *AAMAS*, pages 1362–1370, 2018.

[FAMO14] P. F. Faye, S. Aknine, Sene M., and Sheory O. Stabilizing agent's interactions in dynamic contexts. *2014 IEEE 28th International Conference on Advanced Information Networking and Applications*, pages 925–932, 2014.

[FPY16] Ferdinando Fioretto, Enrico Pontelli, and William Yeoh. Distributed constraint optimization problems and applications: A survey. *CoRR*, abs/1602.06347, 2016.

[FRPJ08] Alessandro Farinelli, Alex Rogers, Adrian Petcu, and Nicholas R. Jennings. Decentralised coordination of low-power embedded devices using the max-sum algorithm. *AAMAS*, pages 639–646, 2008.

[Gd08] Matthew E. Gaston and Marie desJardins. The effect of network structure on dynamic team formation in multi-agent systems. *Computational Intelligence*, 24(2):122–157, 2008.

[Haa02] Ruth Haas. Characterizations of arboricity of graphs. *Ars Combinatorica*, 63, 2002.

[Hol73] C. S. Holling. Resilience and stability of ecological systems. *Annual Review of Ecology and Systematics*, 4(1):1–23, 1973.

[HY97] Katsutoshi Hirayama and Makoto Yokoo. Distributed partial constraint satisfaction problem. *CP*, 1330:222–236, 1997.

[JMN+14] N. R. Jennings, L. Moreau, D. Nicholson, S. Ramchurn, S. Roberts, T. Rodden, and A. Rogers. Human-agent collectives. *Commun. ACM*, 57(12):80–88, 2014.

[JZW13] Yichuan Jiang, Yifeng Zhou, and Wanyuan Wang. Task allocation for undependable multiagent systems in social networks. *IEEE Transactions Parallel Distributed Systems*, 24(8):1671–1681, 2013.

[LAP+10] Patricia H. Longstaff, Nicholas J. Armstrong, Keli Perrin, Whitney May Parker, and Matthew A. Hidek. Building resilient communities: A preliminary framework for assessment. *Homeland Security Affairs*, 6(3):1–23, 2010.

[LOS09] Thomas Léauté, Brammert Ottens, and Radoslaw Szymanek. FRODO 2.0: An open-source framework for distributed constraint optimization. pages 160–164, 2009.

[MPT04] Rajiv T. Maheswaran, Jonathan P. Pearce, and Milind Tambe. Distributed algorithms for DCOP: A graphical-game-based approach. *ISCA PDCS*, pages 432–439, 2004.

[MSTY03] Pragnesh Jay Modi, Wei-Min Shen, Milind Tambe, and Makoto Yokoo. An asynchronous complete method for distributed constraint optimization. *AAMAS*, pages 161–168, 2003.

[New03] Mark E. J. Newman. The structure and function of complex networks. *SIAM Review*, 45(2):167–256, 2003.

[OHLN06] Hisashi Ohtsuki, Christoph Hauert, Erez Lieberman, and Martin Nowak. A simple rule for the evolution of cooperation on graphs and social networks. *Nature*, 441:502–5, 2006.

[ORBI16] Tenda Okimoto, Tony Ribeiro, Damien Bouchabou, and Katsumi Inoue. Mission oriented robust multi-team formation and its application to robot rescue simulation. *IJCAI*, pages 454–460, 2016.

[OSC+15] Tenda Okimoto, Nicolas Schwind, Maxime Clement, Tony Ribeiro, Katsumi Inoue, and Pierre Marquis. How to form a task-oriented robust team. *AAMAS*, pages 395–403, 2015.

[OSD+18] Tenda Okimoto, Nicolas Schwind, Emir Demirovic, Katsumi Inoue, and Pierre Marquis. Robust coalition structure generation. *PRIMA*, 11224:140–157, 2018.

[OW91] C.M. Özveren and A.S. Willsky. Output stabilizability of discrete-event dynamic systems. *IEEE Transactions on Automatic Control*, 36(8):925–935, 1991.

[PF05a] Adrian Petcu and Boi Faltings. A scalable method for multiagent constraint optimization. *IJCAI*, pages 266–271, 2005.

[PF05b] Adrian Petcu and Boi Faltings. Superstabilizing, fault-containing distributed combinatorial optimization. *Proceedings of the 20th National Conference on Artificial Intelligence - Volume 1*, page 449–454, 2005.

[PGCF+15] Marc Pujol-Gonzalez, Jesus Cerquides, Alessandro Farinelli, Pedro Meseguer, and Juan Antonio Rodriguez-Aguilar. Efficient inter-team task allocation in robocup rescue. *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, page 413–421, 2015.

[PSZ+18] Jacopo Panerati, Nicolas Schwind, Stefan Zeltner, Katsumi Inoue, and Giovanni Beltrame. Assessing the resilience of stochastic dynamic systems under partial observability. *PLOS ONE*, 13(8):1–21, 2018.

[Reb93] R. Rebarber. Conditions for the equivalence of internal and external stability for distributed parameter systems. *IEEE Transactions on Automatic Control*, 38(6):994–998, 1993.

[RFMJ10] Sarvapali Ramchurn, Alessando Farinelli, Kathryn Macarthur, and Nicholas Jennings. Decentralized coordination in robocup rescue. *The Computer Journal*, 53:1447–1461, 2010.

[RMWJ15] Talal Rahwan, Tomasz P. Michalak, Michael J. Wooldridge, and Nicholas R. Jennings. Coalition structure generation: A survey. *Artificial Intelligence*, 229:139–174, 2015.

[RNC+96] Stuart J. Russell, Peter Norvig, John F. Candy, Jitendra M. Malik, and Douglas D. Edwards. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Inc., 1996.

[SMI+16] Nicolas Schwind, Morgan Magnin, Katsumi Inoue, Tenda Okimoto, Taisuke Sato, Kazuhiro Minami, and Hiroshi Maruyama. Formalization of resilience for constraint-based dynamic systems. *J. Reliable Intelligent Environments*, 2(1):17–35, 2016.

[SOI+13] Nicolas Schwind, Tenda Okimoto, Katsumi Inoue, Hei Chan, Tony Ribeiro, Kazuhiro Minami, and Hiroshi Maruyama. Systems resilience: a challenge problem for dynamic constraint-based agent systems. *AAMAS*, pages 785–788, 2013.

[SOI+18] Nicolas Schwind, Tenda Okimoto, Katsumi Inoue, Katsutoshi Hirayama, Jean-Marie Lagniez, and Pierre Marquis. Probabilistic coalition structure generation. *KR*, pages 663–664, 2018.

[SSV16] R. Sheh, S. Schwertfeger, and A. Visser. 16 years of robocup rescue. *KI - Künstliche Intelligenz*, 30(3–4):267–277, 2016.

[SVR+10] B.T. Stewart, A.N. Venkat, J.B. Rawlings, S.J. Wright, and G. Pannocchia. Cooperative distributed model predictive control. *Systems and Control Letters*, 59(8):460–469, 2010.

[TIW96] K. Tanaka, T. Ikeda, and H.O. Wang. Robust stabilization of a class of uncertain nonlinear systems via fuzzy control: Quadratic stabilizability, h-infinity control theory, and linear matrix inequalities. *IEEE Transactions on Fuzzy Systems*, 4(1):1–13, 1996.

[TM04] S. Tatikonda and S. Mitter. Control over noisy channels. *IEEE Transactions on Automatic Control*, 49(7):1196–1201, 2004.

[UIY+10] Suguru Ueda, Atsushi Iwasaki, Makoto Yokoo, Marius-Calin Silaghi, Katsutoshi Hirayama, and Toshihiro Matsui. Coalition structure generation based on distributed constraint optimization. *AAAI*, page 197–203, 2010.

[WHCK03] Brian Walker, C. S. Holling, Stephen Carpenter, and Ann Kinzig. Resilience, adaptability and transformability in social-ecological systems. *Ecology and Society*, 9(2), 2003.

[WLL10] Z. Wang, Y. Liu, and X. Liu. Exponential stabilization of a class of stochastic system with markovian jump parameters and mode-dependent mixed time-delays. *IEEE Transactions on Automatic Control*, 55(7):1656–1662, 2010.

[YM16] Yoshiki Yamagata and Hiroshi Maruyama. *Urban Resilience: A Transformative Approach*. Springer, 2016.