Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2013) 21: 1166 – 1181

doi:10.3906/elk-1105-49

Capability-based task allocation in emergency-response environments: a coalition-formation approach

Afsaneh FATEMI,* Kamran ZAMANIFAR, Naser NEMATBAKHSH
Department of Computer Engineering, Faculty of Engineering, University of Isfahan, Iran

Received: 25.05.2011 • **Accepted:** 18.01.2012 • **Published Online:** 03.06.2013 • **Printed:** 24.06.2013

Abstract: This paper addresses coalition formation, based on agent capabilities, centered on task allocation in emergency-response environments (EREs). EREs are environments that need fast task completion as their main requirement. We propose a team-based organization model, based on an existing organization model for adaptive complex systems. The model has some key characteristics that are beneficial for EREs: agents act in dynamic, open domains; agents collaborate in completing group tasks; agents may have similar types of capabilities, but at different levels; tasks need different agent capabilities, at collective different levels; and agents are supervised in a partially decentralized manner. We formulate task allocation as a capability-based coalition-formation problem, propose a greedy myopic algorithm to form coalitions, and compare it with F-Max-Sum, another efficient myopic algorithm. Experiments in which utility is measured show that the capability-based approach outperforms the role-based one. The numerical experiments suggest that the proposed task allocation method is possibly scalable with growing numbers of agents.

Key words: Coalition formation, emergency-response environment, task allocation, team-based organization, capability-based task allocation

1. Introduction

Natural disasters and accidents result in life and financial losses annually worldwide. Upon occurrence, the emergency response units try to use the best of their facilities and proper decision-making processes based on priorities in order to provide rescue and relief for the injured and mitigate the negative effects.

These agents usually act and make decisions in a dynamic, open, and uncertain environment. During the rescue and relief operations, new agents may enter or exit the environment. Moreover, several events may occur in different parts of the environment that catch the attention of the rescue and relief personnel. Many other factors influence the decision-making process regarding the rescuer agents in such environments. The different capabilities existing in various agents, along with the different levels of capabilities and available resources of the agent, affect the decisions about task allocation. In addition, response-time limitation, continuous task occurrence, tasks' priorities, and the need for teamwork are some of the many other factors that require further research to improve the quality of responses to natural disasters.

The other important issue regarding disaster management is that the number of the occurred tasks is greater than the rescue workers having different available resources. Moreover, each task requires different levels of effort for it to be completed in the time allowed. To accomplish such tasks, many agents should work simultaneously according to their capabilities and available resources. Task-performing agents, the members

*Correspondence: a_fatemi@eng.ui.ac.ir

of coalitions, are usually selected based on their roles. In role-based coalition formation, some roles have been defined in the system and each role is able to do some specific actions. When a new task occurs, it needs some actions to be done to get completed. Thus, a coalition of agents enacting related roles should be formed. For example, if the task is extinguishing a fire, only agents enacting the "fire brigade" role could participate in completing it, while there may be, for example, some police agents that have enough capabilities or resources to put the fire out.

In the present paper, a capability-based greedy method is introduced for coalition formation in an open and dynamic environment. The agents are located in a geographically dynamic environment, where agents with different capabilities and capability levels have different coordinates. Many tasks occur in the environment that should be performed by these agents. In this paper, the accomplishing of group tasks is addressed. Two or more agents should simultaneously cooperate to accomplish a group task. In a group, there is no division of labor, and each individual performs the same task [1].

Our contribution includes a team-based organization model for adaptive computational systems (TOMACS), based on the idea of using agents' capabilities instead of their roles to allocate tasks. We expect an improvement in the overall utility by utilizing all of the agents' capabilities along with their role-specific actions. This is the privilege of our proposed capability-based method over familiar role-based ones. Moreover, we formalize the problem of task allocation as a capability-based coalition formation (CBCF) and propose a greedy coalition formation algorithm (GC2FA) to solve it approximately. Since we address the group tasks, a method for fair load balancing between agents is proposed, and the time needed to complete a task is computed based on it.

We briefly review some related works in Section 2. In Section 3, we describe the structure of the proposed organization. In Section 4, we address the problem of coalition formation via task allocation, and we propose a greedy myopic algorithm to solve it. The experimental results are presented in Section 5, and, finally, Section 6 provides the conclusions and some future works.

2. Related works

Ferreira et al. [2] describe the problem of task allocation among teams as finding the assignment that maximizes the overall utility.

In recent years, many centralized and decentralized algorithms have been proposed for task allocation in cooperative multiagent environments.

In the works presenting centralized algorithms, task allocation is considered to be performed in a subspace of all of the possible task-performing agent groups, a much smaller space than the real one. A good example is presented in [3], where reaction functions are proposed for task allocation. A central planner is used to allocate tasks to cooperative agents. Here, the existence of a single point of failure is usually inevitable and this may decrease the robustness of the system. This indicates that such methods cannot be used in the uncertain and dynamic environments in crisis management.

In the recent works performed on task allocation in cooperative multiagent environments, attempts have been made to simulate the problem as a distributed constraint optimization problem (DCOP) or a generalized assignment problem (GAP), and to convert the allocation issue into a known problem in the above fields and solve it by heuristic methods [4,5]. These studies emphasized the GAP and restrictions on the allocation and task implementation for heterogeneous tasks, and synergism was disregarded. Furthermore, they considered current agent roles as the selection criteria when forming coalitions.

A greedy method for task allocation was compared with 2 methods based on the Swarm-GAP algorithm

and low-communication approximate DCOP (LA-DCOP) in [2]. Ferreira et al. used the "extreme team" concept, defined in [5] and applied in [4], in their proposed methods and considered task interdependencies for the first time [2]. They provided token-based task allocation algorithms for large-scale systems, but they ignored the forming of agent coalitions to work on group tasks. Having considered the spatial and temporal constraints, the distributed Max-Sum algorithm is used to solve the task allocation problem. Max-Sum is an approximate algorithm based on message-passing with little local (re)computation and communication. It uses a constraint graph and maximizes a utility function of it, and it is proven to be suitable for solving a certain class of constraint optimization problems. Hence, it can be used in situations where the optimality of the solution can be sacrificed in favor of computational and communication efficiency.

A new version of the algorithm, the F-Max-Sum (FMS), was proposed in [6]. The FMS algorithm is an extension of the Max-Sum algorithm that defines new functions on variable and factor nodes to single out the states that matter to them. This reduces the number of states over which each factor has to compute its solution. Furthermore, the FMS algorithm introduces new functions to allow each variable to decide when to send messages to its other connected factors when the factor graph changes.

The authors of [6] suggested that their proposed method was a complete tool for decentralized team formation in the disaster management field, while it seems to not have been considered that the agents may enter into the environment, exit from it, or have different deadlines.

While each of the mentioned works bridges one or more gaps in the literature, there is still a shortcoming of solutions that consider CBCF for completing the tasks needing agents with the same capabilities at different levels. Furthermore, considering spatial and temporal constraints for tasks is an important issue for emergency-response scenarios.

3. Organizational structure

DeLoach presented a framework for adaptive complex systems, the organization model for adaptive complex systems (OMACS), in [7]. It defines the knowledge about a system's structure and capabilities that is needed for the system's reorganization at runtime in a changing environment.

We extend the OMACS with features of a team-based organization that aims to complete tasks considering the agent's capabilities, the team-based OMACS (TOMACS). Figure 1 shows the TOMACS as our proposed meta-model.

The organizational structure defines the informational, controlling, and communicational patterns and features of the task environment [8–10]. In this research, following the TOMACS, the initial structure of the organization is formed once the system begins to work and reorganization occurs during the system operation, along with the occurrence of reorganization triggers [11]. These triggers include the entrance of a new agent to the organization, the exit of an agent from the organization, the occurrence of a new task, and the completion of an existing task.

The environment is a 2-dimensional geographical space in which a number of agents have been distributed following a statistical distribution pattern or a given map.

Figure 2 shows the organizational structure model. In this model, the initial team formation occurs based on the location of each agent. This idea is taken from human organizations, where a person first tries to get help from his neighbors in emergency situations. The idea is used here wishing to have a simple, speedy, low-cost, and near-natural way to form some preliminary task-serving groups. These initial groups are only a basis to assign some agents as supervisors and to add some discipline to manage the agents.

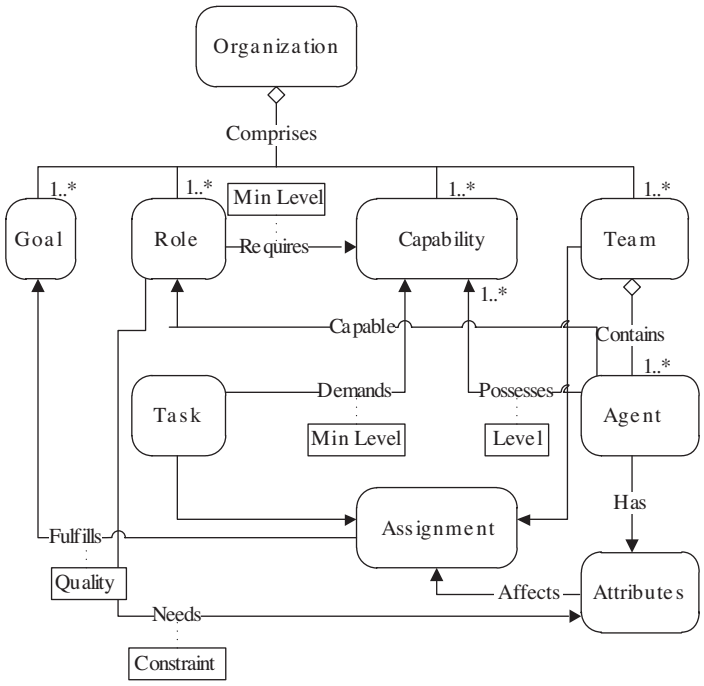

Figure 1. Team-based organization model for adaptive computational systems (TOMACS).

The context is partitioned into some segments and all of the agents placed in each segment form a team. The number of segments is varying as one of the system parameters.

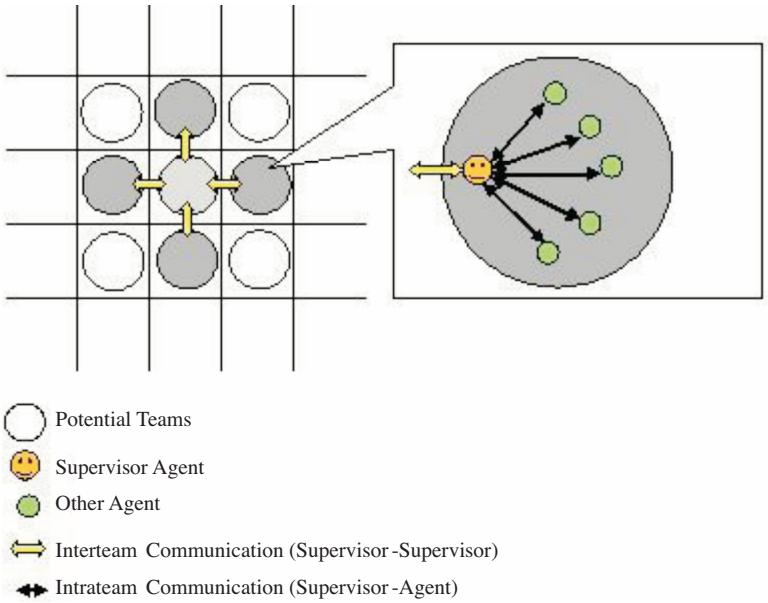

Figure 2. Organizational structure.

As mentioned in [12], a supervisor is required to manage each team. There are many strategies for team supervisor selection in the literature. In fact, leader election is one of the most commonly identified problems in social science. It is not our main problem in this research, and so we use a very simple approach that needs no

additional information or algorithm. All of the agents are assigned a birth-time when created in any multiagent environment. We used this intrinsic characteristic as the election criterion. The eldest agent among all of the team members is assigned as the supervisor. The interesting issue with the age parameter is its relation to the experience factor. It is also inspired from human organizations, where sometimes age is a criterion to select the leader. In future works, we will focus on selecting leaders based on their capabilities.

The communication protocols of the agents follow 4 main rules:

1. In each segment, direct bilateral message-passing exists between the supervisor and individual agents.
2. In each segment, no direct communication exists between ordinary individual agents.
3. The supervisor of each segment has direct bilateral communication with adjacent segments' supervisors, i.e. with upper, lower, right, and left adjacent segments.
4. An agent can sense around itself in a given radius. It can inform the corresponding team supervisor about sensed events or agents.

4. Task allocation via coalition formation

4.1. Preliminary definitions

From now on, we refer to the set of agents constructing the organization as A, the set of tasks to be accomplished by agents as T, the set of all of the recognizable agent capabilities as C, the set of all of the available agent roles as R, and the set of all of the possible spatial locations as U. In addition, L indicates the valid interval indicating the capability levels. Table 1 shows the primary definitions and some clarifying points.

Now each task can be defined as a tuple $t =< p_t o_t w_t d_t r_t >$ and each agent can be shown as a tuple $a =< p_{a,\theta}, s_{a,\theta}, r_{a,\theta}, f_a, v_a, m_a >$.

The time here is considered to be discrete. The tasks' occurrence and accomplishment times are measurable in time scales; the beginning time of the simulation is zero.

In emergency-response environments (EREs), time is a prevailing factor that cannot be overlooked. For example, a firefighter might be able to extinguish a given fire in 1 h, but the chances of trapped individuals making it out alive will be decreased and the inflicted damage will be great, while a team of 4 may contain the same fire in much less time. In both cases, the task is accomplished, but the second case shows a higher task accomplishment quality.

The other purposes of forming coalitions with agents are discussed in [13,14]. Of course, there may be some tasks that do not call for group performance due to their nature. In addition, some types of tasks may be better performed by smaller groups than by larger ones. Hence, it follows that the type of occurred task could influence the decision-making manner of the agents. This effect will be highlighted in the coming sections and the supervisor agents' decision-making algorithm.

4.2. Problem formalization

As mentioned before, each agent has some capabilities to varying levels, which are the most important factors in decision making when coalitions are forming to complete tasks. In addition, the agents' position and movement speed affect the coalition formation, because the agents need to move physically from their locations to the task location, and the tasks have deadlines. Thus, with greedy thinking, it seems that the best agents to complete a task are those nearest to it.

Table 1. Primary definitions.

Notation	Definition	Description
A	Set of all of the agents	$a_1, a_2, ..., a_n \in A$ n: The total number of agents.
T	Set of all of the tasks	$t_1, t_2, ..., t_m \in T$ m: The total number of tasks.
C	Set of all of the potential agent capabilities	$c_1, c_2, ..., c_k \in C$
R	Set of all of the defined roles for the agents	For example: R = {Supervisor, Rescuer}.
U	Set of all of the available spatial locations (as agent's position or task's occurrence target)	Here, the context is a M × N rectangular grid. Therefore, U is a set of all of the tuples as <x,y>, where $x \in [0, M]$, $y \in [0, N]$, and $x, y \in \mathbf{Z}^+$. The context is divi ded into X × Y segments. X and Y are system parameters.
L	The interval to which the capabilities belong.	Here, L = [0,100].
W_t	Set of <needed capability, min level> tuples for task t	$W_t \subset C \times L$
O_t	Occurrence time of task t	$O_t \in [0, \infty)$
d_t	Deadline of task t	$\mathrm{d_t} \in [0, \infty)$ Accomplishment of t after its deadline is worthless.
p_t	The place of task t 's occurrence	$p_t \in U$
$\mathrm{r_t}$	The priority of task t	Tasks could be prioritized based on their occurrence order, their severity, or other features.
$S_{a,\theta}$	Set of <capability, available level> tuples for agent a at time θ	$S_{a,\theta} \subset C \times L$ The time parameter shows that the agent capability level may vary with time.
f_a	Collaboration factor for agent a; the probability that a accepts cooperation in accomplishing a task	$0 \le f_a \le 1$ Here, as in rescue and relief applications, it is assumed to be 1. This means that none of the agents with sufficient capabilities will refuse any kind of contribution.
$p_{a,\theta}$	The position of agent a at time θ	$p_{a,\theta} \in U$ The time parameter shows that the agent is mobile, varying positions over time.
	The sight radius of agent a	The radius at which the agent can sense around itself.
	The movement speed of agent a	The velocity at which the agent can move around.
$r_{a,\theta}$	The role of agent a at time θ	$r_{a,\theta} \in R$ Here, agents can enact different roles due to their status in the organization. The time parameter shows this variability.

It is assumed, as in the real world, that agents have partial information about their environment. Each agent has a specific field of view and can sense around itself along a given radius.

The other issue that should be considered is the autonomy of the agents. As in the real world, the agents are free to accept or decline a task because of personal or environmental reasons. We show this with a collaboration factor, the probability that an agent accepts cooperation in accomplishing a task. In fully cooperative environments, this factor can be set to 1. The collaboration factor is only a sample of the personality features that could be effective on the agents' interactions and behaviors. The effects of personality traits on agent groups are surveyed more in [15].

While the agents are distributed spatially around the environment, they should move toward tasks to participate in their completion. The time needed to travel from one point to another is proportionate to their distance and movement speeds. It is given by the function $V : U \times U \times R^+ \rightarrow R^+$, where U denotes the set of all of the available locations for the agents. Hence, $v_{x,y,m}$ shows the time taken for an agent to travel from point x to point y with movement speed m.

In this paper, we aim to model the task assignment in an ERE as a CBCF problem, where agents have different capabilities with different levels of strengths and form coalitions in order to complete tasks, each with different needs for agent capabilities. The goal of CBCF is to maximize the number of completed tasks. Because of the nature of the ERE, this goal should be achieved along with the minimization of the task completion average time.

Keeping in mind that a coalition is formed when one or more agents work together on a task, we define Φ as the set of all of the possible allocations of coalitions to tasks. $\Omega : T \times \Phi \rightarrow \{0, 1\}$ is a function that returns 1 if a task is successfully completed, and 0 otherwise. $\Theta : T \rightarrow [0, \infty]$ returns the time spent to process a task. The "process" here means investigating the possibility of completing the task, and, if possible, forming a proper coalition and performing the task. Thus, the optimal objective function can be expressed as follows:

$$\arg\max_{\varphi \in \Phi} \frac{\sum_{t \in T} \Omega(t, \varphi)}{\sum_{t \in T} \Theta(t)}. \tag{1}$$

From now on, we will show how agents form coalitions and what constraints should be satisfied by them. Suppose that $\psi_\theta^{a,t}$ indicates that agent a works on task t from time θ. Hence, the set $\Psi = \left\{\psi_\theta^{a,t}\right\}_{a \in A, t \in T, \theta \in [0,\infty]}$ denotes all of the possible allocations of agents to tasks.

Similarly, if $k \in 2^A$ is a coalition of agents of A, $\psi_\theta^{k,t}$ shows that coalition k works on task t from time θ and we can define $\Psi^g = \left\{\psi_\theta^{k,t} \left| k = \left\{a \in A \left| \psi_\theta^{a,t} \in \Psi \right.\right\}, t \in T, \theta \in [0, \infty]\right.\right\}$ as the set of all of the possible coalition assignments to tasks.

If coalition k works on task t, it can decrease the workload of t regarding the capabilities of the agents of k. In this research, we assume that agent capabilities are simply additive and the potential capability of the group is calculated by adding the similar capability levels of its members. We also assume that all agent capability values would be decreased after working on a task demanding that capabilities. For example, the "Fire Extinguishing" capability level could be decreased from 100 to 50 after fire control because the "Water" resource is decreased and should be refilled or else it could participate in the next fire extinguishing with lower strength. In our next work, we will distinguish between capabilities with "Increasing" levels and "Persistent" ones. This separation will help us demonstrate a more realistic model of emergency-response systems. It

is obvious that the coalition formed to be allocated to a task should have enough aggregated capabilities to complete it. Formally, for task $t' \in T$, the following constraint should be satisfied:

$$\underset{i=1..|w_{t'}|}{\forall} < c'_i, l'_i > \in w_{t'} : \quad \psi_{\theta'}^{k',t'} \in \Psi^g \Leftrightarrow \sum_{j=1}^{|k'|} s_{a_j\, c=c'_i}.l_\theta \geq l'_i. \tag{2}$$

In addition, working a coalition k' on task t' from time θ' means that all of the members of the coalition should be present on t' before θ'. This can be presented as follows:

$$\psi_{\theta'}^{k',t'} \in \Psi^g \Rightarrow k' = \left\{ a \in A \,\middle|\, \psi_\theta^{a,t'} \in \Psi, t' \in T, \theta \leq \theta' \right\}. \tag{3}$$

We assumed that an agent can work only on one task at a time (Eq. (4)). It is obvious that an agent can work on 2 different tasks if it has enough time to travel between them (Eq. (5)). In other words, if tasks t_1 and t_2 are assigned to agent a at distinct times θ_1 and θ_2, and no other task is assigned to agent a between these times, then assuming that agent a moves with speed m from the point at which t_1 occurs to the location of t_2, θ_2 should be at least $V_{p_{t_1},p_{t_2},m}$ time units more than θ_1. This means that a feasible agent allocation must satisfy the following:

$$\psi_\theta^{a,t_1} \in \Psi \wedge \psi_\theta^{a,t_2} \in \Psi \Rightarrow t_1 = t_2, \tag{4}$$

$$\begin{aligned} \psi_{\theta_1}^{a,t_1}, \psi_{\theta_2}^{a,t_2} \in \Psi \wedge \left(\theta_3 \in [\theta_1, \theta_2], t_3 \in T, \psi_{\theta_3}^{a,t_3} \in \Psi \Rightarrow (\theta_3 = \theta_1 \wedge t_3 = t_1) \vee (\theta_3 = \theta_2 \wedge t_3 = t_2)\right) \\ \Rightarrow \theta_2 \geq \theta_1 + V_{p_{t_1},p_{t_2},m}. \end{aligned} \tag{5}$$

Note that given the situation in Eq. (4), no overlapping coalitions can be assigned to start different tasks in the same time. In other words:

$$\psi_\theta^{k_1,t_1}, \psi_\theta^{k_2,t_2} \in \Psi^g \Rightarrow k_1 \cap k_2 = \varphi. \tag{6}$$

4.3. Greedy capability-based coalition-formation algorithm (GC2FA)

Most of the proposed coalition-formation algorithms assume that an array of tasks exists in each time period that must be assigned to agent groups. If the system experiences a new task/agent entrance/departure, the computation for optimal group formatting should be conducted anew. In many cases, it may prevent the accomplishment of tasks in the expected time.

In domains like the ERE, the decision making and computations of solutions are considered to be time-critical, where we need algorithms to solve the problem in a rapid manner even if the solutions are not optimal.

In the present paper, a greedy method is proposed for solving the task allocation issue. In this method, the agents do not consider the array of the tasks, but rather focus on one task at a time. Because the environment is dynamic, this is a rational method, since in a dynamic state by a new task entrance, previous optimal solutions become practically obsolete. This is especially true for cases where the new task has more priority than the previous ones.

Moreover, the agents should try to maximize the number of accomplished tasks in the least possible time in order to be able to increase the possibility of accomplishing future tasks. After the completion of each task, the assigned agents are announced as free agents and are ready to participate in other tasks.

In the proposed method, the agents are divided into some preliminary groups based on their locations. The coalition formation algorithm is triggered by the new tasks' occurrence, and the task-performing coalitions

are formed beginning with getting help from the agents inside the respective segment. The pseudocode for the GC2FA is provided in Algorithm 1.

The proposed coalition formation and task allocation methods are distributed in nature. For each task, the supervisor (if any) of the task's occurrence place is the main decision maker to form task-accomplishing groups and assign tasks to them. If tasks occur in short time intervals, their allocation process may be done in parallel by different supervisors. The distributed nature of the proposed approach is intended to improve the system utility.

Algorithm 1. Coalition formation algorithm (GC2FA).

```
// T_θ is the task priority queue of the supervisor running GC2FA at time θ
While ≠ ∅ and time is not over
    t = the task with the highest priority in T_θ
    A_θ = the set of idle agents of corresponding segment at time θ
    K = ∅                          // K is the coalition forming to accomplish task t
    While k is not capable to do t and there is not any not-tested agent in
            a = the nearest A_θ member to t that is not tested and has at least
            one of the capabilities needed by t
            Negotiate with a about t
            If a accepts to participate in performing t
                K ← k + a
    End while
    If k is not capable to do t
        Ask help from the adjacent supervisors     // while k is not capable
                to do t, each adjacent su pervisor runs the same algorithm
    Else   // Assign t to k
        A_θ ← A_θ - k
        θ ← θ + Duration   //Duration is computed as described in Section 4.4
        T_θ ← T_θ - t
        Update the capability levels of k members
End While
```

4.4. Load balancing in task assignment and calculating task duration

When a coalition is assigned to a task, the members use their capabilities to satisfy its requirements to be completed. The load should be distributed between the agents in a balanced manner. As an example, let us assume that $A = \{a_1 a_2 a_3$ and $C = \{c_1 c_2 c_3 c_4\}$, and the capability-level pairs are as follows for the agents and task t:

$$a_1 :< c_1, 76 >< c_2, 5 >< c_3, 45 >< c_4, 20 >,$$

$$a_2 :< c_1, 50 >< c_2, 100 >< c_3, 75 >,$$

$$a_3 :< c_2, 25 >< c_4, 40 >,$$

$$t :< c_1, 65 >< c_2, 12 >< c_3, 40 >< c_4, 50 > .$$

For balanced task assignment, agents should fairly assign their capabilities. For instance, for capability c_1, we should have $\Delta\frac{a_1.c_1}{a_1.c_1} = \Delta\frac{a_2.c_1}{a_2.c_1}$ and $\Delta\Delta a_1.c_1 + a_2.c_1 = \mathrm{t}.c_1$. Note that $\Delta a_i.c_j$ shows the decreasing amount of capability c_j level of agent a_i after performing t. The same reasoning for other capabilities and agents brings us the following:

$\Delta\frac{a_1.c_1}{76} = \Delta\frac{a_2.c_1}{50}$ and $\Delta\Delta a_1.c_1 + a_2.c_1 = 65$,

$\Delta\frac{a_1.c_2}{5} = \Delta\frac{a_2.c_2}{100} = \Delta\frac{a_3.c_2}{25}$ and $\Delta\Delta\Delta a_1.c_2 + a_2.c_2 + a_3.c_2 = 12$,

$\Delta\frac{a_1.c_3}{45} = \Delta\frac{a_2.c_3}{75}$ and $\Delta\Delta a_1.c_3 + a_2.c_3 = 40$,

$\Delta\frac{a_1.c_4}{20} = \Delta\frac{a_3.c_4}{40}$ and $\Delta\Delta a_1.c_{43} + a_3.c_4 = 50$.

The above equations are easily solvable as follows:

$\Delta a_1.c_{1=} 76 \times\frac{65}{76+50}$ and $\Delta a_2.c_{1=} 50 \times\frac{65}{76+50}$,

$\Delta a_1.c_{2=} 5 \times\frac{12}{5+100+25}$ and $\Delta a_2.c_{2=} 100 \times \frac{12}{5+100+25}$ and $\Delta a_3.c_{2=} 25 \times\frac{12}{5+100+25}$,

$\Delta a_1.c_{3=} 45 \times\frac{40}{45+75}$ and $\Delta a_2.c_{3=} 75 \times \frac{40}{45+75}$,

$\Delta a_1.c_{4=} 20 \times \frac{50}{20+40}$ and $\Delta a_3.c_{4=} 40 \times \frac{50}{20+40}$.

In other words, the capabilities of task-performing coalition members are updated as follows:

$$\underset{i=1..|w_{t'}|}{\forall} < c_i', l_i' > \in w_{t'} : \psi_\theta^{k,t'} \in \Psi^g \Leftrightarrow \forall j = 1 \ldots |k|, \Delta\theta S_{a_{j_{c=c_i'}}} l = \theta S_{a_{j_{c=c_i'}}} l \times \theta \frac{\acute{l_i}}{\sum_{j=1}^{|k|} S_{a_{j_{c=c_i'}}}.l}. \tag{7}$$

For simplicity and without damaging the integrity of the problem, we assume that each cooperating agent decreases one unit of the workload of the task in each time unit. Therefore, if the agents start concurrently working on the task simultaneously, the maximum capability decrease in the coalition members will show the time to accomplish the task after forming the coalition. For example, in the above example, the time needed to complete t after allocating to the agents is equal to:

$\mathrm{Max}(\Delta\Delta\Delta a_1.c_1, a_2.c_1, a_1.c_2,\ \Delta a_2.c_2,\ \Delta a_3.c_2,\ \Delta\Delta a_1.c_3, a_2.c_3,\ \Delta a_1.c_4,\ \Delta a_3.c_4)$.

Hence, task t will be accomplished after $76 \times\frac{65}{76+50}$ time units from the start point.

5. Experimental results

Our experiments consist of 3 parts. First we compare the proposed CBCF method with another myopic algorithm proposed in [6] against the rate of successful task handling. A second series of experiments shows the effect of problem size on system utility. In the last part, we study the effect of segmentation size on the utility.

Regarding a large part of the literature, the RoboCup Rescue simulation environment seems to be a well-recognized environment in which to develop and benchmark multiagent techniques, but some inconsistencies encourage us to develop another simulation environment. Lack of control over the simulation environment and the difficulty of its manipulation, special communication infrastructures that restrict some types of communication, limitation of the agent roles to only 3 types, and the existence of some central agent types in contrast to our distributed decision-making idea are some examples of these inconsistencies. In addition, the RoboCup Rescue simulator is provided with a role-based insight, but we prefer to use a more flexible capability-based one. Hence, a new tool was developed to simulate, data-log, and evaluate the proposed organizational model using JADE, the Java Agent Development Framework [16]. All of the parts of our experiments are done using this tool. The number of tasks is specified before the simulation starts. Each round of simulation continues

until all of the tasks are completed successfully, all of the tasks fail because of their deadlines, or some tasks are completed and the other tasks fail because the total strength of the coalition is not sufficient to do them or tasks are not completed before their deadlines.

5.1. Capability-based versus FMS myopic coalition formation

Rumchurn et al. [6,13] modeled the RoboCup Rescue domain as an ERE in terms of a coalition formation with spatial and temporal constraints. They provided a DCOP formulation of the problem and proposed a myopic solution using the FMS algorithm [6], an improvement of the Max-Sum algorithm [17]. They showed that the proposed myopic solution is a good approximation of the optimal one. We applied our proposed capability-based algorithm and the FMS-based approach to the same data set and compared the rate of the completed tasks.

In this experiment, we apply the proposed GC2FA and the FMS algorithm to the same data set. Both algorithms are implemented in Java in the JADE environment. The number of agents is fixed at 10 and the number of tasks varies from 10 to 60. The positions of the agents are created with a random distribution on a 300 $\times$ 300 grid, where the travel time between 2 points is computed based on Euclidian distance. The workload of the tasks is generated uniformly from 0 to 100 for each capability. We repeat the experiments 20 times and compare the mean total number of the tasks completed by both strategies. Figure 3 shows the results.

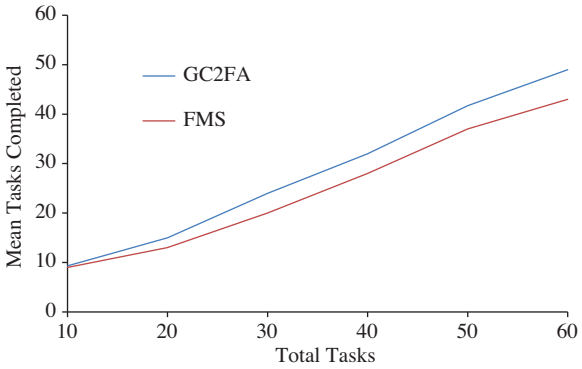

Figure 3. Number of tasks completed by agents, comparing the GC2FA and FMS algorithms.

As can be seen in Figure 3, the GC2FA outperforms the FMS by up to 13.5% (for 40 tasks). It should be noted that the GC2FA is much faster than the FMS. This is because the FMS algorithm finds the optimal solution of all of the considered solutions but the GC2FA is a greedy algorithm that is acceptable for EREs, where finding a fast reasonable solution is more important than finding an optimal one.

5.2. Effect of problem size on utility

Scalability is an important quality factor for multiagent systems. In EREs, the system is expected to preserve its acceptable response time as the problem size grows. In a rescue and relief system, the main goal is to rescue more civilians in the shortest period of time. Therefore, the utility of the system could be defined as the rate of the completed tasks divided by the mean task accomplishment time. The mean task accomplishment time is computed considering both the completed tasks and the failed tasks, but some time is consumed to investigate their completion possibility.

In hierarchical organizational models, the organizational tree is expanded horizontally and vertically as the size of the mass grows. The big organization tree makes the adaptation process complicated and time-

consuming. The hierarchical organization model proposed in [18] is an example. Hence, these models are not suitable for large-scale critical mass.

In the proposed team-based model, the agents initially form some teams due to their location. These teams are potential candidates for performing tasks in a limited area. A greedy coalition formation algorithm tries to find minimal coalitions if the candidate team is not strong enough to complete the task. This strategy is intended to motivate a better task accomplishment in a near-time–efficient and near-resource–efficient manner than in hierarchical strategies. In addition, it seeks a uniform work load distribution and system scalability.

The proposed team-based model is implemented together with a simple greedy team-formation algorithm introduced in the previous sections, and it performs several experiments in order to evaluate the system scalability and effectiveness. The number of agents and total number of tasks are varying as the system's parameters. For each problem size, we measure the percentage of completed tasks and the task accomplishment time, and we compute the utility. Each experiment is repeated 20 times for each input set and the average of the results is computed. Only some of the values from the infinite set of possible values are considered for the experiments. Table 2 shows the summary of the parameters and results.

Table 2. Some experimental results.

Number of agents	Number of tasks	Mean rate of successful task handling	Mean time to accomplish a task (ms)	Utility
10	5, 10, 20	0.95	9.8	0.097
20	10, 20, 40	1	8.5	0.118
50	25, 50, 75	0.8	10.1	0.079
75	50, 75, 100	0.75	13.2	0.057
100	50, 100, 150	0.73	12.7	0.057
150	75, 150, 200	0.77	11.8	0.065
200	100, 200, 250	0.8	14.7	0.054
250	100, 200, 250	0.69	14	0.049
300	100, 200, 250	0.87	12.74	0.068

The results show smooth changes in the utility function when the problem size is increased. This shows that the proposed team-based model is possibly scalable enough to be used in medium-scaled multiagent environments.

The changes of the utility function with an increasing problem size are shown in Figure 4. As the experiments show, although the system has more utility in small problem sizes (number of agents and tasks less than 50), the utility has no high oscillations or reductions at larger problem sizes. The utility remains around 0.06 for problem sizes between 60 and 300 and shows smooth changes. This means that the system is scalable for numbers of agents and tasks less than 300. This is noteworthy because the system is open and the problem size may change during execution.

It seems that rapid team formation, proper load distribution between the agents, and team-based task handling contributes to the system's effectiveness.

5.3. Effect of segmentation on utility

Eq. (1) shows that the overall utility of an emergency-response system would be increased as more tasks are completed in less time. In this research, the organizational model and coalition-formation method are proposed

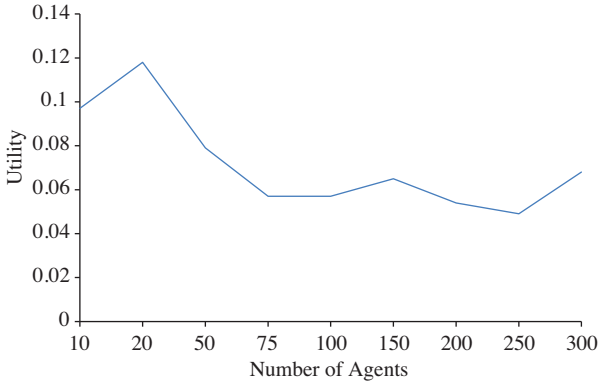

Figure 4. Utility of the team-based model at different problem sizes.

paying attention to this aspect. System parameters seem to have a great effect on system behavior and its utility function. The number of segments that the physical environment is divided into is one of these parameters. As argued in Table 1, the environment is assumed as an $M \times N$ grid that is divided into $X \times Y$ segments with X horizontal and Y vertical parts. The number of segments directly affects the initial teams formed from the agents and their communications. If $X = Y = 1$, then we have a supervised agent organization based on direct supervision. Increasing X and Y increases the number of preliminary teams and when $X = M$ and $Y = N$, we would have potentially $M \times N$ distinct preliminary one-member teams that each try to do the tasks individually or get help from their neighbors.

In this part of the experiments, some scenarios in different problem sizes are selected and 20 runs of simulations are done for each state. The system utility is computed as the average of the results. Each set of experiments is performed for both sight = 50 and sight = 100. The agents and tasks are generated due to a random spatial distribution and the tasks are of medium complexity size (maximum workload for each needed capability = 500).

Figure 5 shows the utility changes against some different numbers of segments. Here it is assumed that $X = Y$. This means that the environment is divided into the same number of vertical and horizontal divisions. The horizontal axis shows this number and the system utility is shown in the vertical axis.

The results show that with the same number of agents and tasks, the best utility occurs around a specific segment's number. For 10 agents and 10 tasks, this peak point occurs at point 1. This means that for small size problems, segmentation may bring out more overhead. For 30 agents and 30 tasks, the best utility occurs at point 2. This means that for these settings, it is better to divide the context into 2 × 2 segments. The proper number of segments seems to be increased when the problem size increases. For 50 agents and 50 tasks, the best result is obtained with 5 segments. This behavior is reasonable: when the number of segments increases, the number of agents placed in each group decreases. As a consequence, the cumulative capability of the preliminary groups decreases and more intrasegment communications are needed to complete a task. On the other hand, segmentation may cause some reduction in intersegment communications. This is because of the proposed communication protocol between the team supervisor and the other agents. These 2 aspects are in a trade off, and a balance should be found between them. The results are highly related to the dynamics of the environment and the tasks' and agents' attributes. The analysis of the behavior of the agents in different situations needs more research and simulation, but we can rationally suggest an upper bound for the number of segments. It seems that if the agents are distributed uniformly, such that almost all of the segments include an active team, the tasks are sensed and considered with higher probability. Hence, if the agents and tasks are

distributed uniformly, $|A|$ shows the number of agents, and $M = N$, then $\left\lceil \sqrt{|A|} \right\rceil$ seems to be a good upper bound for the number of segments in each dimension. For example, for 20 agents, it is better to simulate only for $1 \leq M \leq 5$ It is $1 \leq M \leq 6$ for 30 agents, and $1 \leq M \leq 8$ for 50 agents.

It can also be noticed that the utility level increases when the agents' sight radius increases. This is reasonable because as the agent's sight increases, it can sense the occurring events more than before, and so it can plan to complete them with the help of other agents. The experiments show that increasing the agents' sight radius will result in a positive shift in utility, but the near-ideal point of the segment numbers remains the same. This is also shown in Figures 5a–5d.

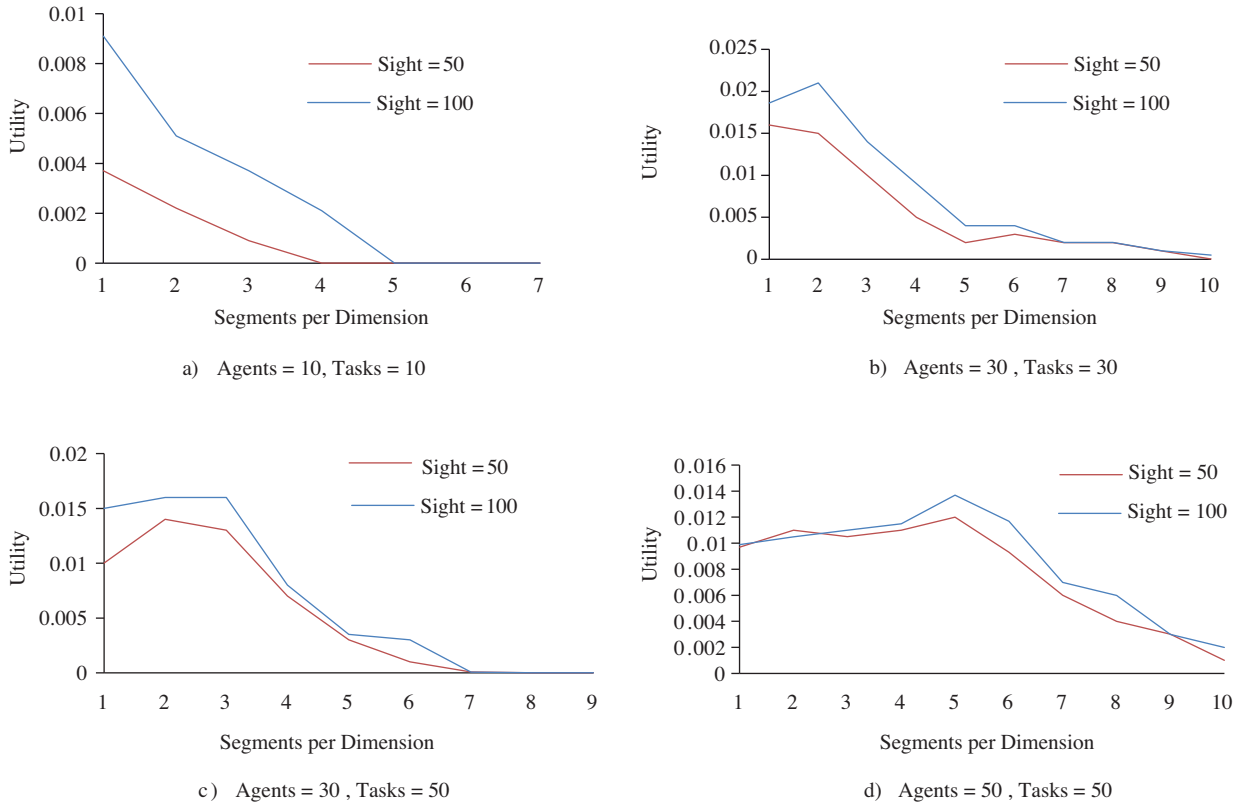

Figure 5. Utility in different numbers of segments.

6. Conclusion and future work

In this paper, the problem of task allocation in EREs was addressed as a coalition-formation problem in a team-based organizational model. More specifically, a team-based organizational model was formulated based on the OMACS framework. A coalition-formation problem with spatial and temporal constraints was formulated and a simple greedy capability-based algorithm was proposed to solve it with a myopic point of view. In so doing, we have provided a capability-based approach instead of the previous role-based methods to form coalitions to complete tasks in EREs.

Experiments show that the proposed capability-based method outperforms the FMS in the number of tasks completed. The work loads are fairly distributed between the teammates. In addition, it seems to scale with growing problem sizes because of its semi-decentralized method in management and decision making. It seems that segmentation has a great effect on the system utility. More experiments and analysis are needed

to find the exact relation between the parameters. Different tunable organizational factors make the system flexible enough for better efficiency. The effect of these factors should be investigated through simulation.

Future work can initially involve proposing other coalition formation algorithms and testing the effect of task and environmental factors on system efficiency. To do so, the authors are going to develop a more effective simulation environment in order to support the open, dynamic, and uncertain environment's properties. Varying agent capabilities, different types of tasks, variable number of segments, the agents' changeable views, and controllable output information are some features to be developed as tools in the future works.

Acknowledgment

This research was fully supported by the Graduate School of the University of Isfahan and the Iranian Ministry of Science, Research, and Technology.

References

[1] A. Campbell, A.S. Wu, "Multi-agent role allocation: issues, approaches, and multiple perspectives", Autonomous Agents and Multi-Agent Systems, Vol. 22, pp. 317–355, 2011.

[2] P.R. Ferreira, F.D. Santos, A.L. Bazzan, D. Epstein, S.J. Waskow, "RoboCup rescue as multiagent task allocation among teams: experiments with task interdependencies", Autonomous Agents and Multi-Agent Systems, Vol. 20, pp. 421–443, 2010.

[3] X. Zheng, S. Koenig, "Reaction functions for task allocation to cooperative agents", Proceedings of the 7th International Conference on Autonomous Agents and Multiagent Systems, Vol. 2, pp. 559–566, 2008.

[4] F.D. Santos, A.L. Bazzan, "Towards efficient multiagent task allocation in the RoboCup rescue: a biologically-inspired approach", Autonomous Agents and Multi-Agent Systems, Vol. 22, pp. 465–486, 2011.

[5] P. Scerri, A. Farinelli, S. Okamoto, M. Tambe, "Allocating tasks in extreme teams", Proceedings of the 4th International Joint Conference on Autonomous Agents and Multiagent Systems, pp. 727–734, 2005.

[6] S.D. Ramchurn, A. Farinelli, K.S. Macarthur, N.R. Jennings, "Decentralized coordination in RoboCup rescue", The Computer Journal, Vol. 53, pp. 1–15, 2010.

[7] S.A. DeLoach, "OMACS: a framework for adaptive, complex systems", In: V. Dignum, Ed., Multi-Agent Systems: Semantics and Dynamics of Organizational Models, Hershey, PA, USA, IGI Global, 2009.

[8] M. Ghijsen, W. Jansweijer, B. Wielinga, "Towards a framework for agent coordination and reorganization, AgentCoRe", Proceedings of the International Conference on Coordination, Organizations, Institutions, and Norms in Agent Systems III, pp. 1–14, 2008.

[9] R. Kota, N. Gibbins, N.R. Jennings, "Decentralised structural adaptation in agent organisations", Proceedings of the AAMAS Workshop on Organized Adaptation in Multi-Agent Systems, Estoril, Portugal, pp. 54–71, 2008.

[10] M. Schwaninger, Intelligent Organizations: Powerful Models for Systemic Management, 2nd ed., Berlin, Springer, 2009.

[11] A. Fatemi, K. Zamanifar, N. Nematbakhsh, O. Askari, "A team-based organizational model for adaptive multi-agent systems", Proceedings of the 3rd International Conference on Agents and Artificial Intelligence, Vol. 2, pp. 469–472, 2011.

[12] H. Mintzberg, Structures in Five: Designing Effective Organizations, 1st ed., Hoboken, NJ, USA, Prentice Hall, 1993.

[13] S.D. Ramchurn, M. Polukarov, A. Farinelli, C. Truong, N.R. Jennings, "Coalition formation with spatial and temporal constraints", Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems, Vol. 3, pp. 1181–1188, 2010.

[14] T.C. Service, J.A. Adams, "Coalition formation for task allocation: theory and algorithms", Autonomous Agents and Multi-Agent Systems, Vol. 22, pp. 225–248, 2011.

[15] S.N. Givigi, H.M. Schwartz, "Swarm robot systems based on the evolution of personality traits", Turkish Journal of Electrical Engineering and Computer Sciences, Vol. 15, pp. 257–282, 2007.

[16] F. Bellifemine, G. Caire, A. Poggi, G. Rimassa, "JADE: A software framework for developing multi-agent applications, lessons learned", Information and Software Technology, Vol. 50, pp. 10–21, 2008.

[17] A. Farinelli, A. Rogers, A. Petcu, N.R. Jennings, "Decentralised coordination of low-power embedded devices using the Max-Sum algorithm", Proceedings of the 7th International Conference on Autonomous Agents and Multi-Agent Systems, pp. 639–646, 2008.

[18] M. Ghijsen, W. Jansweijer, B. Wielinga, "Adaptive hierarchical multi-agent organizations", Interactive Collaborative Information Systems, Studies in Computational Intelligence, Vol. 281, pp. 375–400, 2010.