Comparing and Evaluating Organizational Models: A Multi-Agent Programming Contest Case Study

Mariana Ramos Franco, Jaime Simão Sichman

Laboratório de Técnicas Inteligentes (LTI)
Escola Politécnica (EP)
Universidade de São Paulo (USP)
mafranko@usp.br, jaime.sichman@poli.usp.br

Abstract. An important subset of multi-agent systems (MAS) are based on *organizational models*. These models try to define pre-defined intended agent interaction patterns. Given an application domain, however, the choice of a particular organizational model that better solves the problem is still an open problem. In order to guide this choice, a MAS developer must have the opportunity to test distinct organizational models easily. In this work, we compare and evaluate different organization models of a MAS, whose goal is to evolve in the "Agents on Mars" scenario proposed in the Multi-Agent Programming Contest (MAPC).

1 Introduction

Recently, there have been a movement towards the explicit design and use of organizations in multi-agent systems (MAS). An organization helps to better model the problem being tackled, and it helps to increase the system's efficiency, by defining the MAS structure and the rules which the agents must follow to achieve individual and system level goals. However, in many cases it is difficult to define the organizational model that better solves the problem.

Trying to contribute to this issue, we present in this paper an experimental analysis of the overall result of different organization-oriented MAS, which were created for the "Multi-Agent Programming Contest" scenario.

2 Background

2.1 Agent Organizational Models

Organization is a complex notion: there are several views, definitions, and approaches to characterize them, addressing different issues: it is a supra-individual phenomena [1], it is defined by the designer or by the actors involved [2], and it is a pattern of predefined [3] or emergent [4] cooperation.

We will adopt here the following definition [5]:

"An organization is a supra-agent pattern of emergent cooperation or predefined cooperation of the agents in the system, that could be defined by the designer or by the agents themselves, in order to achieve a purpose."

One important issue is the relation between organizational constraints and agents' autonomy, as studied by Castelfranchi [6]. When seen as a predefined cooperation pattern, an organization aims to constrain the agents' autonomy. In fact, this limitation aims to guarantee that the global goals are achieved in an optimized way. If agents follow strictly their organizational constraints, they will know what to do, when and with whom to interact in crucial problem solving situations.

Given that an organization constrains the agents' autonomy, a further step is to investigate how this limitation can be properly engineered and designed. Coutinho et al. [7] propose some modeling dimensions for organizational design: *(i)* the *structural dimension*, mainly composed of notions like roles and groups, as used in the AGR model [8]; *(ii)* the *interactive dimension*, characterized by dialogical interaction structures, as used in the Electronic Institutions model [9]; *(iii)* the *functional dimension*, formed by goal/task decomposition structures, as proposed by the TAEMS model [10]; and *(iv)* the *normative dimension*, in which we find the concepts of norms, rights, rules, like used in the OPERA model [11].

However, the organizational design problem has not been solved so far by researchers in business and management domains. This problem can be stated as: how to find an optimal constraint set that could guarantee global efficiency for a given task scenario? The same problem arises concerning multi-agent organizations [12].

In this paper, we present a comparison and evaluation of different organization models, that were applied to "Agents on Mars" scenario, described next.

2.2 MAPC

The "Multi-Agent Programming Contest"[1] (MAPC) is held every year since 2005, and it is an attempt to stimulate research in MAS programming techniques [13]. In the contest, two teams of agents are located in the same environment and compete directly in a scenario set by the organizers. By being a direct competition, it is an interesting testbed to evaluate and compare different systems, allowing to identify strengths and weaknesses, and thus promoting the development of all participants.

Since 2011, a scenario called "*Agents on Mars*" has been used. In this scenario, two teams of 28 agents compete to explore and dominate the best top wells of the planet. The environment is represented by a weighted graph, where the vertices denote wells and possible locations for the agents, and the edges indicate the possibility of crossing from one vertex to another with an energy cost for the agent. Each vertex has a value corresponding to its water well usefulness, and this value is used to calculate the value of the areas occupied by the agents.

[1] `http://multiagentcontest.org`.

A zone is a subgraph covered by a team according to a coloring algorithm based on the notion of domain [14]. Several agents from different teams can be located in a single vertex, but the team with the highest number of agents dominates this vertex, which receives the dominant team color. An uncolored vertex inherits the color from its neighbourhood dominant team. Hence, if the graph contains a subgraph with a colored border, all the nodes that are within this boundary receive the same color. This means that an agent team may cover a subgraph which has more vertices than the number of its members. Figure 1 shows a map with the colored subgraphs.

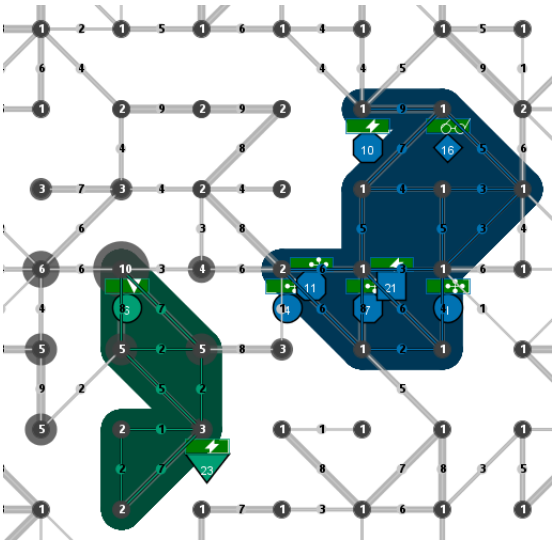


Fig. 1: *"Agents on Mars"* Scenario.

At the beginning of the simulation, the map is unknown to the agents. Thus, each team needs to explore the graph before starting to conquer areas, since the agents have a limited view of the map and only perceive their neighbour vertices. Additionaly, sometimes a team needs to sabotage the other team to increase its area, or to defend areas in order not to loose them to the opponent.

Each team consists of 28 players, that can play 5 different roles: explorers (Exp), sentinels (Sen), saboteurs (Sab), inpectors (Ins) and repairers (Rep). These roles define the characteristics of each agent, such as life level, maximum energy, strength, and visibility. The roles also limit the possible actions that the agent can perform in the environment, as shown in Table 1. For instance, explorers can find water wells and help to explore the map, while sentinels have long distance sensors and thus can observe larger areas, saboteurs can attack and disable enemies, inspectors can spy opponents, and repairers can repair damaged agents.

A team receives a cash reward whenever it reaches a major milestone. This reward can be used to empower the agents, increasing, for instance, their maximum energy or strength. Different milestones can be reached during a competition, such as dominating areas with fixed values (e.g., 10 or 20), having performed a successful number of attacks or well-succeded defenses. If not used, the reward is added to the team's total score.

The goal of each team is to maximize its score, defined as the sum of the values obtained by the occupied zones with the earned (and not yet spent) rewards in each step of the simulation, as shown in Equation 1:

$$score = \sum_{p=1}^{steps} (zones_p + rewards_p) \quad (1)$$

Table 1: "Agents on Mars" Roles and Actions.

	Explorer	**Repairer**	**Saboteur**	**Sentinel**	**Inspector**
recharge	x	x	x	x	x
attack			x		
parry		x	x	x	
goto	x	x	x	x	x
probe[2]	x				
survey[3]	x	x	x	x	x
inspect[4]					x
buy	x	x	x	x	x
repair		x			
skip	x	x	x	x	x

We present next a proposal of an agent team, called LTI-USP, based on different organizational models, to solve this problem.

3 LTI-USP Agent Team

3.1 Architecture

The architecture of the LTI-USP team is shown in Figure 2. In this architecture, we used BDI agents. Each agent is composed of plans, a belief base and its own world model. The agent decides which plan will be executed according to its beliefs and the local view of the world.

[2] A priori, the agents have no knowledge about the value of water wells. A team only gets the full value of a vertex after one team agent have analyzed the water well.

[3] Initially, the agents do not know what is the cost of crossing from one vertex to another. An agent needs to survey it to find the value of each edge.

[4] This action collects information about the opponents present in neighboring vertices, such as energy and role.

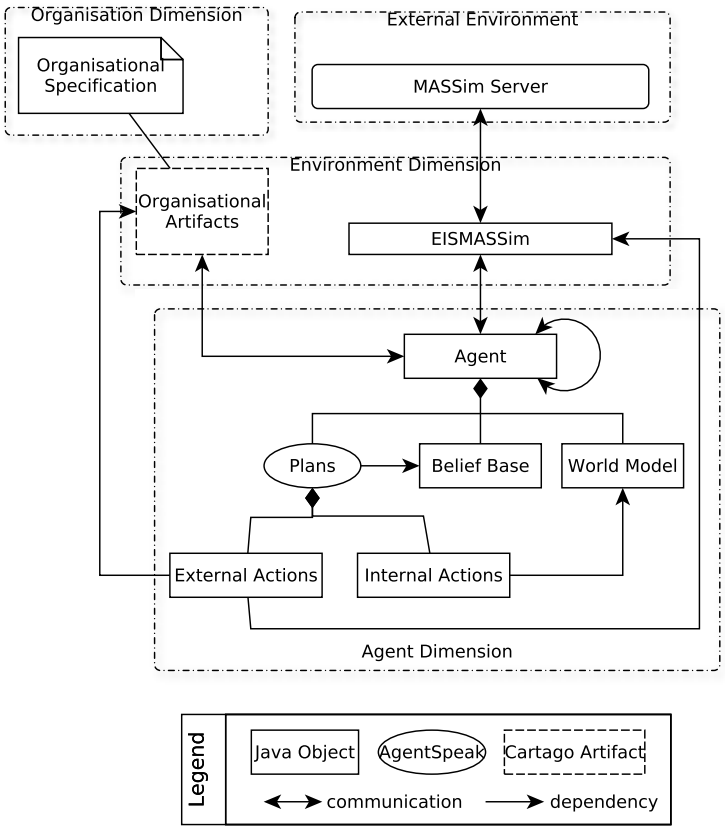


Fig. 2: LTI-USP Team Architecture.

The world model consists of a graph developed in *Java*, using simple data structures and classes. It captures every detail received from the MASSim contest server, such as explored vertices and edges, opponents' position, disabled teammates, etc. At each step, the agent's world model is updated with the percepts received from the MASSim server, and with the information received from the other agents.

Some of the percepts received from the MASSim server are also stored in the agent's belief base, such as the agent's role, energy, position and team's rewards, thus allowing the agent to have a direct access to these information without have to access its world model. Percepts about vertices, edges and other agents were not stored in the belief base so as to not compromise the agent's performance, as it could be very expensive to update and to access the belief base with so much information. Moreover, since we wanted to update a belief when a new instance was inserted (instead of adding a second one), we decided to use an indexed belief base in which some beliefs are unique and indexed for faster access.

Our team was developed using *JaCaMo*[1]. *JaCaMo* [15] is a platform for multi-agent programming which supports all levels of abstractions – agent, environment, and organization – that are required for developing sophisticated MAS, by combining three separate technologies: *Jason*[2] [16], for programming autonomous agents; *CArtAgO*[3] [17], for programming environment artifacts; and *Moise*[4] [18], for programming multi-agent organizations.

Jason is a Java-based interpreter for an extended version of the AgentSpeak programming language, suitable for programming BDI agents.

CArtAgO is a framework for environment programming based on the A&A meta-model [19]. In *CArtAgO*, the environment can be designed as a dynamic set of computational entities called artifacts, organized into workspaces, possibly distributed among various nodes of a network [15]. Each artifact represents a resource or a tool that agents can instantiate, share, use, and perceive at runtime. For this project, we did not create any new artifact; we only made use of the organizational artifacts provided in *Moise*.

Moise [18,20] is an organizational model for MAS based on three complementary dimensions: *structural*, *functional* and *normative*. The model enables a MAS designer to explicitly specify its organizational constraints, and it can be also used by the agents to reason about their organization. We used the *Moise* model to define the agent's roles, groups and missions.

Agents communicate with the MASSim server through the EISMASSim environment-interface included in the contest software-package. EISMASSim is based on EIS[5] [21], which is a proposed standard for agent-environment interaction. It automatically establishes and maintains authenticated connections to the server and abstracts the communication between the MASSim server and the agents to simple Java-method-calls and call-backs. In order to use this interface, we extended the *JaCaMo* default agent architecture to perceive and to act not only on the *CArtAgO* artifacts, but also on the EIS environment as well.

3.2 Strategies

The main strategy of our team is to divide the agents into two or more subgroups: one in charge of attacking the opponents (`infantry`), and the others (`squads`) in charge of occupying the best zones in the graph. Moreover, regarding the agents' roles, we decided not to map the five types specified in the scenario (Exp, Ins, Rep, Sab and Sen) directly to the roles in our team. Instead, we defined additional different roles in our system according to the adopted strategy, as shown in Figure 3.

Each of these roles has a mission associated to it, and can be played by one or more type of agents. For example, the `map_explorer` role can be played only

[1] Available at http://jacamo.sourceforge.net/.
[2] Available at http://jason.sourceforge.net/.
[3] Available at http://cartago.sourceforge.net/.
[4] Available at http://moise.sourceforge.net/.
[5] Available at http://sourceforge.net/projects/apleis/.

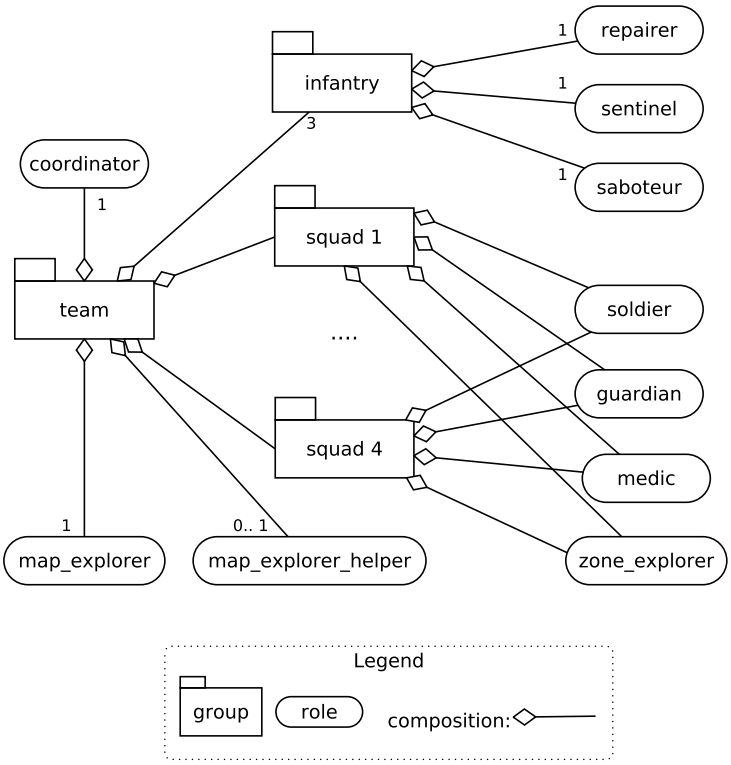


Fig. 3: LTI-USP Team *structural specification.*

by the explorer type, while the `soldier` role can be played by all types of agents. Below we describe the missions related to each role:

- **map_explorer** (Exp): Explores the whole graph by probing every vertex and surveying all edges on its path;
- **map_explorer_helper** (Exp): Helps the `map_explorer` to explore the graph, but only in the first 250 steps. After that, the agent leaves this role to adopt the `soldier` role in the `best_zone` subgroup;
- **soldier** (all types): Tries to occupy one of the best zones indicated by the `coordinator` agent. When all the vertices of the designated best zone are occupied the `soldier` starts to look to the neighbour vertices of the team's zone to which he can move to increase the zone size;
- **guardian** (Sab): Defends the subgroup best zone by attacking any opponent that is close to the team's zone, or trying to invade it;
- **medic** (Rep): Occupies the center of the designate best zone and is responsible for repairing the agents in the subgroup, or other agents which eventually need to be repaired, such as the `map_explorer`. In our team, the damaged agents move to the repairers position in order to be repaired;

- **zone_explorer** (Exp): Explores the team's zone by probing the vertices whose values are unknown. When all vertices are probed, the `zone_explorer` helps the `soldiers` to increase the zone size;
- **saboteur** (Sab): Attacks any close opponent, or the opponent who occupies a good vertex;
- **sentinel** (Sen): Tries to sabotage the opponent by moving inside its zone;
- **repairer** (Rep): Follows the `saboteur`, but always staying two vertices away from it, in order to be prepared to repair the `saboteur` when necessary, but without taking too much risk;
- **coordinator** (none): Agent internal to our system which does not communicate with the MASSim server. It builds its local view of the world through the percepts broadcasted by the other agents. Whenever the world model is updated, it computes which are the best zones in the graph and send this information to the other agents. The `coordinator` is also responsible for creating the organizational artifacts, in the beginning of the simulation, and for distributing the groups, roles and missions among the other agents, in order to eliminate the performance issues caused by two or more agents trying to adopt the same role in a group, or trying to commit to the same mission.

The best zone in the map is obtained by calculating for each vertex the sum of its value with the value of all its direct and second degree neighbours. The vertex with the greatest sum of values is the center of the best zone. Zones with the sum of values below 10 are not considered in the calculation[6]. The same computation is performed again to determine if there is a second, third and fourth best zone, and so on, but this time removing the vertices belonging to the first best zone from the analysis. If the number of best zones is smaller than the number of `squads`, the first best zone is designated to the subgroups without specific best zone.

4 Experiments and Results

4.1 Experiments

In the MAPC, each team plays against each other team three times, and the team that wins most matches wins the overall tournament. Each match has 750 steps and the map is randomly generated, thus from one match to another the number of vertices, edges and high-valued areas can change.

The fact that the number of high-valued areas may change leads in some cases to situations where to protect a single best area is a better strategy, while in other cases it would be better to divide the team in smaller groups to try to gain control over several areas. Therefore, we have performed some experiments to analyse how the number of `squads` in our team can impact in its overall performance.

[6] This threshold value was obtained empirically, by analyzing the results of previous editions of the contest.

The experiments consisted of four teams (TG1, TG2, TG3 and TG4), all of them with the structure shown in Figure 3, except with respect to the number of `squads` as shown in Table 2. These teams competed in three different scenarios/maps (SC1, SC2 and SC3), described in Table 3. In this table, possible zones means a certain number of neighbour vertex with high values, that are hence possible candidates for a best zone. These scenarios are also represented in Figure 4.

Table 2: LTI-USP Team Configurations.

Team	**Squad**	**soldiers**	**guardians**	**medics**	**zone_explorers**	**Agents**
TG1	1	20	1	1	1	23
TG2	1	10	1	1	1	13
	2	7	1	1	1	10
TG3	1	5	1	1	1	8
	2	5	1	1	1	8
	3	4	1	1	1	7
TG4	1	3	1	1	1	6
	2	3	1	1	1	6
	3	3	1	1	1	6
	4	3	0	1	1	5

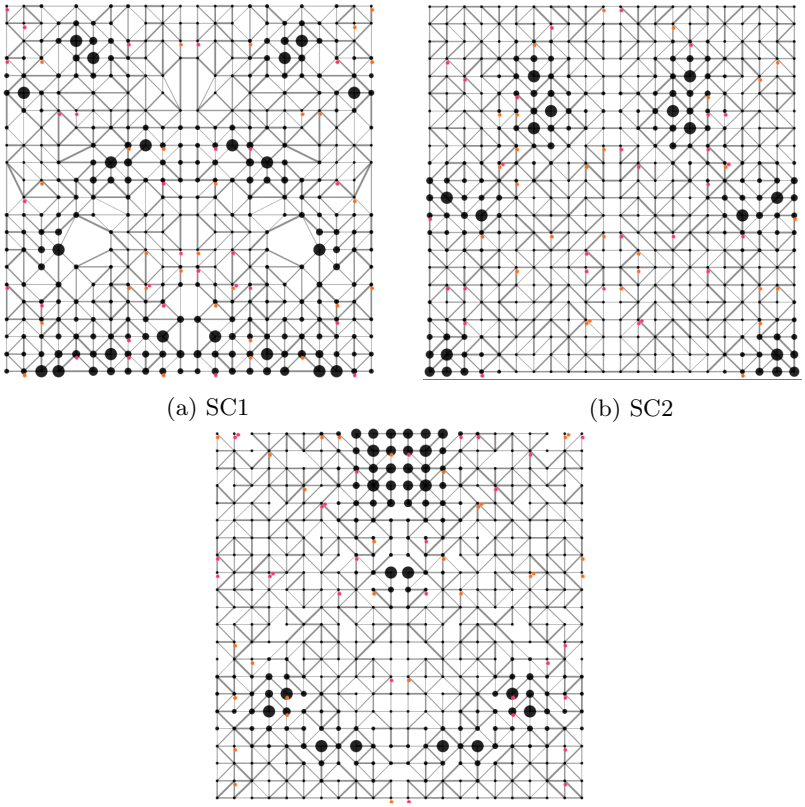
Table 3: Scenarios Properties.

	Vertex	**Edges (thinning factor)**	**Possible zones**
SC1	400	1110 (20%)	9
SC2	500	1345 (40%)	6
SC3	600	1234 (10%)	6

The number of vertices and edges shown in Table 3 were chosen according to the parameters set in the MAPC, in which the maps had from 400 to 600 vertices and the thinning factor, i.e., the number of removed edges from a complete connect graph in percent, varies from 10% to 60%.

4.2 Results

For each scenario previously described, we performed 10 simulations for each of the following matches: TG1 vs TG2, TG1 vs TG3, and TG1 vs TG4. The data collected in all simulation were: (i) the winner, (ii) the teams' final scores and (iii) the score conquered in each step for each of the two competing teams. Table 4 shows a summary of the number of wins for each team by match and scenario.



(a) SC1 (b) SC2

(c) SC3

Fig. 4: Experiment Scenarios.

Given the results, we used a hypothesis test, the *Wilcoxon T test*, to define for each match if the 10 simulations were sufficient or not to conclude that a team was better than other in a determined scenario. The *Wilcoxon T test* (also called *Wilcoxon signed-rank test*) is a non parametric test for dependent samples that can indicate with some stated confidence level if a particular population tends to have larger values than other.

The results of this analysis are shown in Table 5, where the values correspond to the *p-value* result of the *Wilcoxon T test* applied on the final score of the 10 simulations performed for each match. A *p-value* lower than 0.05 indicates that the results observed in the 10 simulations are enough to conclude that a certain team tends to obtain higher scores than other.

The results obtained for each scenario are analysed in the following subsections.

Table 4: Results Summary - Number of wins.

	TG1 x TG2	TG1 x TG3	TG1 x TG4
SC1	2 x 8	4 x 6	1 x 9
SC2	0 x 10	0 x 10	0 x 10
SC3	4 x 6	1 x 9	2 x 8

Table 5: *Wilcoxon T test*

	TG1 x TG2	TG1 x TG3	TG1 x TG4
SC1	0.02881	0.5787	0.005196
SC2	0.0115	0.002879	0.0002057
SC3	0.1655	0.06301	0.02323

Scenario 1 In the first scenario, the teams with more `squads` won most of the simulations against the TG1 (control team) and, given the *p-values* of the *Wilcoxon T test*, we can conclude that TG2 and TG4 are better than TG1, but for TG3 we can not conclude the same.

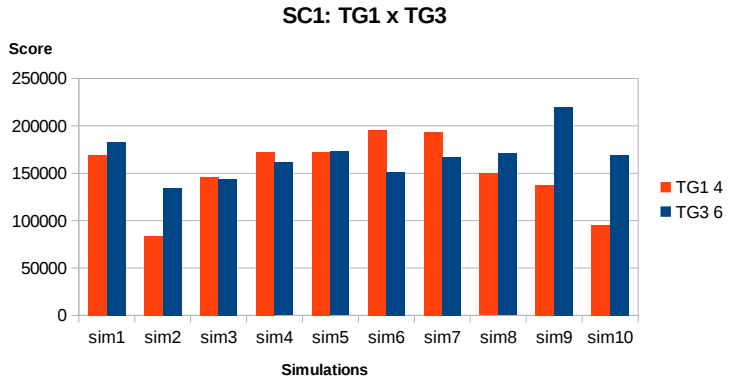


Fig. 5: Scenario 1: Final scores for TG1 vs TG3.

Figure 5 shows the final scores of the 10 simulations for the match TG1 vs TG3. Analysing the simulations where TG1 defeats TG3, we were able to identify why we have good results for TG2 against TG1, while this has not occurred when TG3 played against TG1. TG3 divides its agents in three `squads` to occupy three different zones in the map, while TG1 uses all its agents (apart from that ones used to attack the opponent and to explore the map) to try to conquer the best zone in the map. In this first scenario, there is a unique huge high valued area in the left bottom of map, which is easily conquered by TG1

since the number of agents from TG3 that will fight for the same zone is not enough to defeat the opponent. Besides that, also the score summed from the two others TG3's `squads` was lower than the score obtained by TG1 in the best zone.

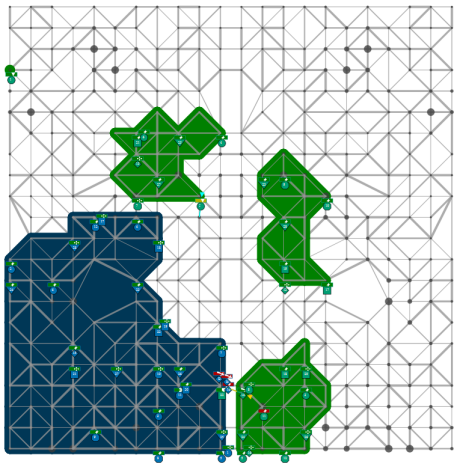


Fig. 6: Scenario 1 - TG1 (blue) vs TG3 (green).

Scenario 2 Contrasting if the first scenario, the second one does not has a huge hight valued area, and all possible best zones have almost the same value, what is good for the teams with more `squads`, as shown in Figure 7.

Scenario 3 The third scenario, as the first one, has an unique huge high valued area which now is located in the top of the map, but in this scenario TG2 did not performed as well as in the first scenario.

Figure 8 shows the results of the 10 simulations for the match TG1 vs TG2, where is possible to see that TG2 narrowly lost two simulations for TG1.

Comparing the match TG1 vs TG2 in this scenario with the first one, we were able to identify that in this third scenario, TG2 does not find in some simulations the best zone in the map, since the zone is not so spread as in the first scenario. In these cases, TG1 won when it was able to find the best zone and TG2 not, as depicted in Figure 9.

5 Conclusions

The problem of determining an appropriate or best MAS organization for a given scenario is a key problem in MAS research, and empirical approaches can be very

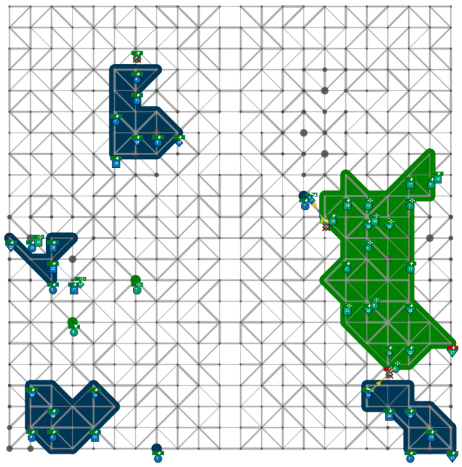


Fig. 7: Scenario 2: TG1 (green) vs TG4 (blue).

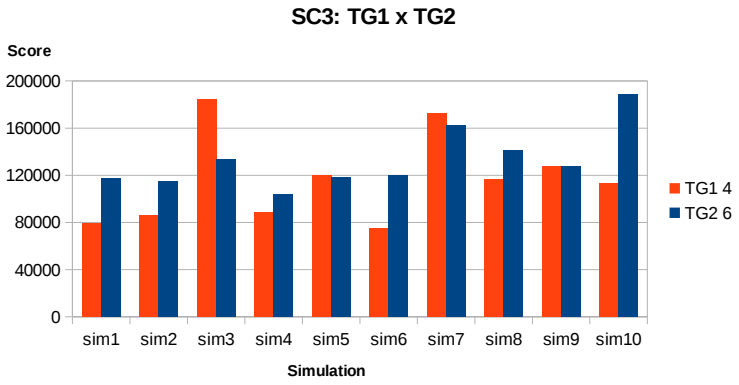


Fig. 8: Scenario 3: Final scores for TG1 vs TG2.

useful in this regard. Aiming to contribute in this issue, we presented an evaluation of different organizations over three distinct scenarios of the Multi-Agent Programming Contest case study. To validate our observations, a statistical test, the *Wilcoxon T test*, was used to detect differences in the performance of the organizations.

The results obtained by confronting the four LTI-USP teams, even though they can suggest that TG4 is the best organizational choice, are not conclusive since the number of scenarios used in our evaluation was relatively small, and the scenario can greatly impact the performance of the team as we showed in Section 4.2.

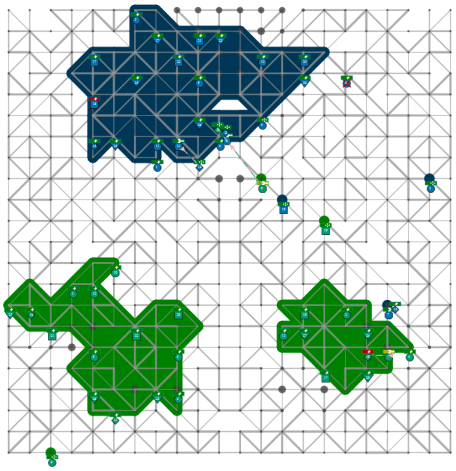


Fig. 9: Scenario 3: TG1 (blue) vs TG2 (green).

Therefore, in future work we intend to increase the number of different tested scenarios, and also evaluate different structures of organizational models, changing not only the number of `squad` but also other parameters, for instance the number of agents in charge of attacking the opponents.

Another possibility is to use the results obtained in this study to develop a team capable of reorganizing according to the characteristics of the environment. As discussed in [22], the reorganization is an important aspect in MAS, since the environment is most often not static. Therefore, MAS should be able to modify your organization to adapt to changes in the environment.

Acknowledgements

Jaime Simão Sichman is partially supported by CNPq and FAPESP/Brazil.

References

1. Gasser, L.: Organizations in multi-agent systems. In: Pre-proceedings of the 10th European Workshop on Modelling Autonomous Agents in a Multi-Agent World. IMAG, Annecy, France (2001)
2. Malone, T.W.: Modeling coordination in organizations and markets. In Bond, A.H., Gasser, L., eds.: Readings in Distributed Artificial Intelligence. Morgan Kaufmann Publishers, Inc., San Mateo, CA, USA (1987) 151–158
3. Bernoux, P.: La Sociologie des Organisations. Seuil, Paris, (1985)
4. Morin, E.: La Méthode (1): La Nature de la Nature. Seuil, Paris (1977)
5. Boissier, O., Sichman, J.S.: Organization oriented programming. Tutorial Notes, 3rd. International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004), New York, USA (August 2004)

6. Castelfranchi, C.: Guarantees for autonomy in cognitive agent architecture. In Wooldridge, M.J., Jennings, N.R., eds.: Intelligent Agents. Volume 890 of Lecture Notes in Artificial Intelligence. Springer-Verlag, Berlin, DE (1995) 56–70
7. Coutinho, L.R., Sichman, J.S., Boissier, O.: Modelling dimensions for agent organizations. In Dignum, V., ed.: Handbook of Research on Multi-Agent Systems: Semantics and Dynamics of Organizational Models. IGI Global, Hershey (2009) 18–50
8. Ferber, J., Gutknecht, O.: A meta-model for the analysis and design of organizations in multi-agents systems. In Demazeau, Y., ed.: Proceedings of the 3rd International Conference on Multi-Agent Systems, Paris, France, IEEE Computer Society Press (1998) 128–135
9. Esteva, M., Rodríguez-Aguilar, J.A., Sierra, C., Garcia, P., Arcos, J.L.: On the formal specification of electronic institutions. In Dignum, F., Sierra, C., eds.: Agent-mediated Electronic Commerce. Volume 1191 of Lecture Notes in Artificial Intelligence. Springer-Verlag, Berlin, DE (2001) 126–147
10. Decker, K.S.: TÆMS: A framework for environment centered analysis and design of coordination mechanisms. In O'Hare, G.M.P., Jennings, N., eds.: Foundations of Distributed Artificial Intelligence. John Wiley & Sons Ltd., Baffins Lane, UK (1996) 429–447
11. Dignum, V.: A model for organizational interaction: based on agents, founded in logic. Phd Thesis, University of Utrecht, Utrecht, The Netherlands (2004)
12. Horling, B., Lesser, V.R.: A survey of multi-agent organizational paradigms. Knowledge Eng. Review **19**(4) (2004) 281–316
13. Behrens, T., Köster, M., Schlesinger, F., Dix, J., Hübner, J.: The Multi-agent Programming Contest 2011: A Résumé. In Dennis, L., Boissier, O., Bordini, R., eds.: Programming Multi-Agent Systems. Volume 7217 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg (2012) 155–172
14. Behrens, T., Köster, M., Schlesinger, F.: The multi-agent programming contest 2011: a résumé. Programming Multi-Agent Systems (2012) 155–172
15. Boissier, O., Bordini, R.H., Hübner, J.F., Ricci, A., Santi, A.: Multi-agent oriented programming with JaCaMo. Science of Computer Programming (2011)
16. Bordini, R., Hübner, J., Wooldridge, M.: Programming multi-agent systems in AgentSpeak using Jason. Wiley-Blackwell (2007)
17. Ricci, A., Piunti, M., Viroli, M.: Environment programming in multi-agent systems: an artifact-based perspective. Autonomous Agents and Multi-Agent Systems **23**(2) (June 2010) 158–192
18. Hübner, J.F., Boissier, O., Kitio, R., Ricci, A.: Instrumenting multi-agent organisations with organisational artifacts and agents. Autonomous Agents and Multi-Agent Systems **20**(3) (April 2009) 369–400
19. Omicini, A., Ricci, A., Viroli, M.: Artifacts in the a&a meta-model for multi-agent systems. Autonomous Agents and Multi-Agent Systems **17**(3) (December 2008) 432–456
20. Hübner, J., Sichman, J., Boissier, O.: Developing organised multiagent systems using the MOISE+ model: programming issues at the system and agent levels. International Journal of Agent-Oriented Software Engineering (2007) 1–27
21. Behrens, T.M., Dix, J., Hindriks, K.V.: The Environment Interface Standard for Agent-Oriented Programming - Platform Integration Guide and Interface Implementation Guide. Department of Informatics, Clausthal University of Technology, Technical Report **IfI-09-10** (2009)
22. Dignum, V.: Handbook of Research on Multi-agent Systems: Semantics and Dynamics of Organizational Models. Information Science Reference (2009)