Comparison of Different Coordination Strategies for the RoboCupRescue Simulation

Sébastien Paquet, Nicolas Bernier and Brahim Chaib-draa

DAMAS Laboratory
Computer Science and Software Engineering Department
Laval University, Québec, Canada, G1K-7P4
{spaquet;bernier;chaib}@damas.ift.ulaval.ca

Abstract. A fundamental difficulty faced by cooperative multiagent systems is to find how to efficiently coordinate agents. There are three fundamental processes to solve the coordination problem: mutual adjustment, direct supervision and standardization. In this paper, we present our results, obtained in the RoboCupRescue environment, comparing those coordination approaches to find which one is the best for a complex real-time problem like this one. Our results show that a decentralized approach based on mutual adjustment can be more flexible and give better results than a centralized approach using direct supervision. Also, we have obtained results showing that a standardization rule like the partitioning of the map can be helpful in those kind of environments.

1 Introduction

Cooperative multiagent systems in which agents must interact together to achieve their goals is a very active field of research [13]. A fundamental difficulty faced by such systems is to find how to efficiently coordinate the actions of different agents (independent software entities) to make them help each other, instead of harming each other [3,4,6]. The coordination in multiagent systems is the process used to manage the dependencies between different activities [5].

There are three fundamental processes to solve the coordination problem [7]: mutual adjustment [11,12], direct supervision and standardization [9,10]. Mutual adjustment means that each agent is trying to adapt its behaviour to improve the coordination. Direct supervision, means that there is one agent that can send orders to other agents. Finally, standardization means that there are some social laws enforcing the coordination among the agents.

In this paper, we present some tests comparing a decentralized approach (mutual adjustment) against a centralized approach (direct supervision). We also have made another test comparing the mutual adjustment with the standardization approach. The results presented have been obtained in the RoboCupRescue simulation environment. This is a real-time environment where agents, representing firefighters, policemen and paramedics have to cooperate to save people after a natural disaster in a city [1,2].

In RoboCupRescue, the coordination is very important because there are strong dependencies between agents activities. For instance, a firefighter cannot extinguish a fire if the roads were not cleared by a policeman to enable him to reach the fire. In fact, for a complex problem like the RoboCupRescue, it is not obvious to see which coordination strategy is the most effective. It is why we have made some tests to compare different strategies and to find which one is the best for this kind of application.

This paper compares the usefulness of decentralisation on two different tasks: extinguishing fires and rescuing civilians. In addition, it studies the usefulness of partitioning the environment when the tasks are really dynamic and only one agent has to be assigned to a specific task. The partitioning can be seen as a social law [9] dictating the responsibilities of each agent. Before explaining those strategies, we firstly present our test environment, the RoboCupRescue simulation.

2 The RoboCupRescue Simulation: a Complex Environment

The goal of the RoboCupRescue simulation project is to build a simulator of rescue teams acting in large urban disasters [1,2]. This project takes the form of an annual competition in which participants design rescue agents trying to minimize damages, caused by a big earthquake, such as civilians buried, buildings on fire and blocked roads. In the simulation, participants should have approximately 30 to 40 agents of six different types to manage:

– *FireBrigade*: Their goal is to extinguish fires.
– *PoliceForce*: Their goal is to clear roads to enable agents to circulate.
– *AmbulanceTeam*: Their goal is to transport injured agents and search in shattered buildings for buried civilians.
– *Center agents*: There are three types of center agents: *FireStation*, *PoliceOffice* and *AmbulanceCenter*. The only actions those agents can make are to send and receive messages. A center agent can read more messages than a platoon agent (moving agents in the simulation), thus they can have a better global view of the situation, so center agents can serve as information centers and coordinators for their platoon agents.

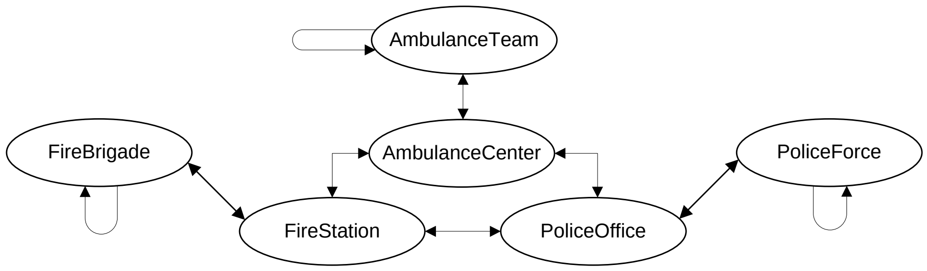

Fig. 1. Communications' organization in RoboCupRescue. Links between types of agents indicate that a communication is possible by radio between those types of agents.

Fig. 1 shows the communications' organization between the different types of agents in the RoboCupRescue simulation. This organization limits the liberty of communication between agents. As we can see, it takes at least three steps for a message to go from a *FireBrigade* agent to a *PoliceForce* agent.

The RoboCupRescue environment is complex because it imposes some hard constraints such as:

- a real-time constraint on the agents' response time because all agents have to be able to reason in less than 500 ms,
- the agents' perceptions are limited,
- the number of messages that an agent can send and hear is also limited.

In the simulation, each individual agent receives visual information of only the region in its surroundings. Thus, no agent has a complete knowledge of the global state of the environment. Therefore the RoboCupRescue domain is in general, collectively partially observable [8]. This means that even if agents are putting all their perceptions together, they will not have a perfect perception of the environment. This uncertainty complicates the problem greatly. Agents will have to explore the environment, as it would not be enough to limit themselves on the visible problems. They will also have to communicate to help each other to have a better knowledge of the situation.

Another difficulty in the RoboCupRescue simulation comes from the fact that the agents are heterogeneous, they cannot do everything by themselves. Therefore, they need to cooperate in order to accomplish their goal efficiently.

Furthermore, agents have to be really careful about the messages they send, because it is really easy to loose messages due to the limitations on the number of messages an agent can hear and also due to the organization presented in Fig. 1.

3 Centralized Against Decentralized Decision Making

In this section, we present two different tasks and for each of those two, we present a centralized and a decentralized decision process. First, we define the *FireBrigades*' task that consists in choosing which fire to extinguish. Then, we present the *AmbulanceTeams*' task that consists in choosing which civilian to rescue.

3.1 FireBrigade Agents

The main task of the *FireBrigade* agents is to extinguish fires. Therefore, the main decision that those agents have to take is to decide which fire to extinguish among all visible fires. To address this, we have developed two different decision processes: one decentralized among all *FireBrigade* agents and one centralized in the *FireStation*.

Decentralized Decision Process. In the decentralized process, each *FireBrigade* chooses the fire it wants to extinguish by itself, so all decisions concerning fire extinguishment is taken in a distributed manner, locally by each *FireBrigade* agent.

Each *FireBrigade* agent is maintaining a list of all the buildings on fire which it knows about. This list is then sorted according to a certain function estimating how good the choice of each building is. With this function, *FireBrigades* are prioritizing certain categories of building. Empirically, we have chosen some weight to define the relative importance of fires' characteristics to make agents prioritize :

- the early fires, with an intensity of 1, because they are generally easy to extinguish and at the border of a zone on fire;
- the fires that put the biggest area in danger, to protect the biggest area;
- the closest fires, because it is faster to reach such fires;
- the smallest fires, because they are easier to extinguish;
- the fires with *FireBrigades* already on it, because it is faster to extinguish a fire in group;
- the buildings that are not concrete reinforced, because those are harder to extinguish.

Centralized Decision Process. In the centralized decision process, practically all decisions are made by the *FireStation* agent. Therefore, the decision process is centralized in one agent. The *FireStation* is informed of the situation, i.e. where the fires are, by messages sent by all agents in the simulation. At each turn, the *FireStation* agent uses a function, similar to the one described in the decentralized approach, to sort all fires according to their importance.

Afterwards, the *FireStation* agent sends a message to each *FireBrigade* containing the two best fire it has identified. Those two fires are seen as orders by the *FireBrigades*, so they obey and try to extinguish the first fire on the list. When the first one is extinguished, they try to extinguish the second one.

The *FireBrigade* agents blindly obey the *FireStation*. The only time they choose by themselves which fire to extinguish is when they do not receive a message from the *FireStation*.

Results. To test those two different coordination approaches, we have made some test runs on three different maps. To study the effectiveness of our agents at accomplishing the extinguishing task, we have recorded the surface burned in each situation. In Fig. 2, we can see two graphics representing the average surface burned during 20 runs on two different cities. In Fig. 2 a), the decentralized approach gives better results than the centralized approach, but on Fig. 2 b), it is the opposite.

Those results show that there is not one approach that is better than the other in all situations. To explain that, we should note that one of the big difference between the two simulated cities is that at the beginning, the roads were more blocked in city a). This has an impact because the center in the centralized approach does not consider the agents' positions, so if the roads are blocked, there is a good chance that the agents have some difficulties to reach the building on fire. Since each agent in the decentralized approach is considering its distance from the fire, it has more chance to reach the chosen fire as it is normally closer.

With the centralized approach, if the roads are not too blocked, the results are better because the center has the best global view and consequently it can choose the best fire to extinguish and the *FireBrigades* are able to reach the building more easily. When the roads are more blocked, the decentralized approach becomes better because each agent chooses closer fires, so it has more chance to reach them.

That shows that the best approach would be to combine those two extremes by having an approach where the decision is made in a centralized way in some situations and in a decentralized way in other situations. The combination and the identification of those situations will be the subject of future research.

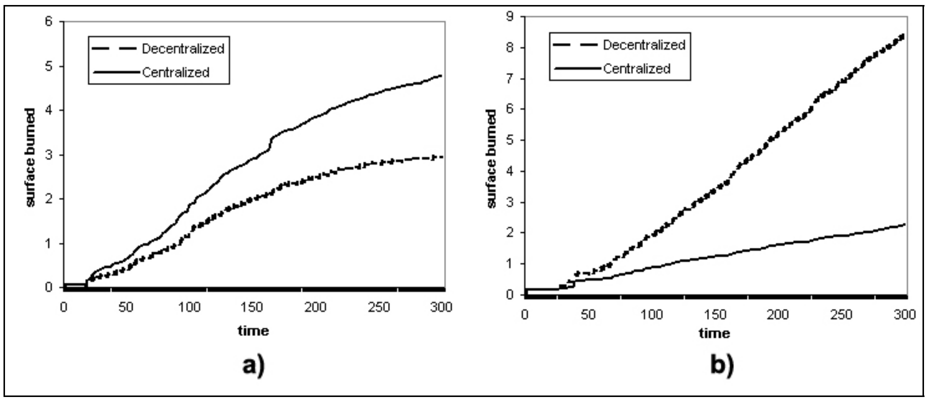

Fig. 2. Average surface burned during 20 runs of two different situations.

3.2 AmbulanceTeam Agents

The main task of the *AmbulanceTeam* agents is to rescue civilians. Therefore, the main decision those agents have to take is to decide which civilian to rescue between all known buried civilians. As for the *FireBrigades*, we have developed two different decision processes: one decentralized among all *AmbulanceTeam* agents and one centralized in the *AmbulanceCenter*.

Decentralized Decision Process. In the decentralized process, all *AmbulanceTeam* agents choose which civilian to rescue by themselves. They use a greedy planning algorithm to try to maximize the number of civilians that could be saved. Each task, corresponding at saving a civilian, has a time length giving the necessary time to save the civilian. This time is the sum of: (a) the travel time to go to the civilian location, (b) the rescuing time and (c) the time to transport the civilian to the refuge. Each task also has a deadline representing the expected death time of the civilian.

The chosen civilian is the one that maximizes the number of civilians that could be rescued after him. To do that, we count for each civilian how many other civilians we would have time to save before its expected death time. The “best one” is chosen by the *AmbulanceTeam* to be rescued. After this civilian has been rescued, the *AmbulanceTeam* chooses the next one with the same algorithm.

Centralized Decision Process. In the centralized decision process, all decisions are made by the *AmbulanceCenter*. At each turn of the simulation, this center agent sends the ordered list of civilians to rescue to the *AmbulanceTeams*. The center agent determines which civilians to save and in which order by using the same algorithm described before for the decentralized process. The only difference is that the center estimates the traveling time to be two times the time to reach a refuge from the civilian location. It has to do an estimation because it doesn't know the position of each *AmbulanceTeam*, so it does not know the distance from the *AmbulanceTeam* to the civilian. At each turn, the center agent determines the two best civilians to rescue and sends this information to each *AmbulanceTeam*. Since all *AmbulanceTeams* receive the same messages, they are always rescuing the same civilian. This is a good behavior because they work faster when they work together. For example, if one *AmbulanceTeam* agent is rescuing a civilian, it could take 30 minutes, but if the are two rescuing the same civilian, it could take 20 minutes and with three it could take 15 minutes. When they are in group, they can dig faster to find the civilian trapped in the building.

Results. To test those two approaches, we have executed the simulation 20 times on three different maps and we have recorded the total *HP* of all agents in the simulation. The *HP* represents the healthiness of an agent. If the *HP* is 0, the agent is dead.

Fig. 3 shows the average of the total *HP* for the centralized and the decentralized approaches. As we can see, the two approaches are quite similar. In our two other maps, the results are even more identical. One reason for this could be that the *AmbulanceTeam* agents have practically the same vision of the situation as the *AmbulanceCenter*. This is because they receive almost all the messages concerning the civilians. Therefore, since they are choosing the civilians based on the same function they have a good chance to choose the same civilian, thus generating the same behavior as for the centralized process.

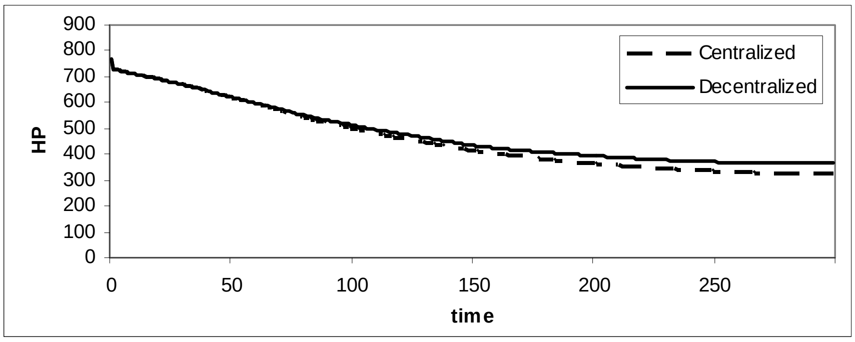

Fig. 3. The average total *HP* for the centralized and decentralized approaches.

The little difference in favor of the decentralized process could be explained by the better individual choice of civilians at the beginning. In the decentralized way, each *AmbulanceTeam* agent chooses a closer civilian, so it can have more chance to be able to reach it if the roads are blocked. This distinction is rapidly lost because after a short time, the *AmbulanceTeam* agents can move more freely and since they all have approximately the same view of the situation, they almost choose the same civilian every time, thus generating the same behavior as in the centralized process.

4 Partitioning the Environment

In this section, we present our results showing the impacts of partitioning the environment. In fact, this simplifies the coordination, because each agent has a region for which it is responsible for. To illustrate that, we present some results obtained with our *PoliceForce* agents in two different settings: with sectors or without sectors.

PoliceForces are playing a key role in the rescue operation by clearing the roads, thus enabling all agents to circulate. Without them, some actions would be impossible because all other agents would be indefinitely blocked by roads' blockades. Therefore, it is really important for them to be fast and efficient. For those agents, we have developed two strategies: one where each agent is free to execute all tasks, no matter where they come from, and an other strategy where some agents are responsible for a specific sector assigned to them at the beginning of the simulation.

4.1 Partitioning in Sectors

For this strategy, we have divided the map in nine sectors to help agents with the task allocation problem. So, at the beginning of the simulation, when the agent receives the information of the city map, it begins by dividing the map in nine homogeneous sectors and sends its position to the *PoliceOffice*. When the *PoliceOffice* has received all the positions of the *PoliceForce* agents, it assigns a sector to the nine agents that are closer to the center of a sector. Thus, there is one and only one *PoliceForce* affected to a sector and this agent has the responsibility of this sector. By doing so, we have divided our *PoliceForces* in two groups: those with a sector and those without a sector. Therefore, even in the strategy with sectors, we have some agents that do not have a sector because there are more agents than sectors. Those agents without a sector are in charge of clearing the roads around the closest fire. Here is a list of the strategies, in their priority order, that *PoliceForce* agents follow:

1. Unblock other agents in its sector.
2. Clear roads between fires and refuges in its sector.
3. Clear roads around refuges in its sector.
4. Clear all the roads in its sector.

Thus, the highest priority task of a *PoliceForce* agent with a sector is to help the other agents in its sector. This is very important, because *FireBrigades* and *AmbulanceTeams* would not be able to do their tasks if they are blocked. Therefore, when an agent is blocked it sends a message to the *PoliceForces* indicating its position. When it receives the message, the *PoliceForce* responsible for this sector adds it to the list of roads to clear. When a *PoliceForce* agent has to make a decision, it chooses the closest road to clear on this list.

When the preceding list is empty, the *PoliceForce* agents with a sector try to open the roads for the *FireBrigade* agents by clearing all roads between a fire in the sector and the closest refuge. Notice that the refuge can be in its sector or not. This is a proactive action to help the *FireBrigades*, because there is a good chance that those agents would ask for a road clearing. *FireBrigades* have a good chance to use those roads because they have to refill their tank at the refuge, so they are often going from a fire to a refuge and in the other direction too. This strategy helps to reduce the communications and the agent movements.

Afterwards, *PoliceForce* agents are clearing roads around refuges. They clear all roads in a perimeter of 40 meters around the refuge, if it is in their sector. This is also a proactive action, because refuges are intensively used by the *FireBrigade* and the *AmbulanceTeam* agents.

When the three preceding tasks have been done, *PoliceForces* clear all the roads in their sector. They first calculate the best path to visit all the roads in their sector. After this, they will follow this path and clear all blocked roads on their path. Thus, when all those tasks are done, all roads on the map have been cleared.

4.2 Without Partitioning

In this case, *PoliceForce* agents have only two tasks. The first one is to clear roads to help the other agents. When they receive a message asking for a clearing, they add it to the list of roads to clear. From this list, they choose the closest road to clear. Since they do not have any sector, they try to help all agents, no matter where they are on the map.

The second task is to clear roads on the path from a fire to a refuge. Unlike the other type of *PoliceForces*, they are not restrained to a specific sector, so they choose the closest fire. By clearing the path from a fire to a refuge, they clear the roads around fires and around refuges. This is interesting since they are really important spots to clear to help the other agents to move freely in the city.

4.3 Results

To test those two approaches, we have executed the simulation 20 times on three different maps. We have recorded the width of each road that is blocked in the simulation and the number of time that an agent is asking to clear a road. The graphic of Fig. 4 a) indicates that the *PoliceForce* agents with sectors gave better global results. In this graphic, the score is the total score given by the RoboCupRescue simulation. There is not a big difference, but in average, the partitioning of the map in sectors helped those agents.

On the Fig. 4 b), we can see that the *PoliceForces* with sectors are more efficient at clearing roads, because there are less roads blocked in average at the end of the simulation. The Fig. 4 c) shows that the number of clearing messages with or without the partitioning strategy is the same. One explication might be that *PoliceForce* agents with sectors are clearing all the roads in their sector, but they may clear roads that are not used by other agents. That could be why the number of clearing messages is the same even if there are less roads blocked when the agents are using sectors.

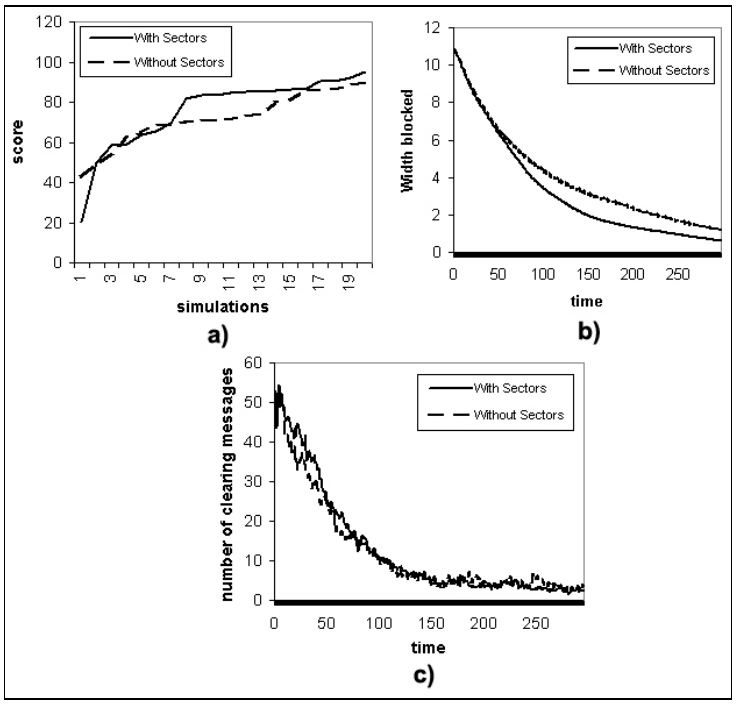

Fig. 4. Comparison of the results obtain for the *PoliceForce* agents with and without sectors. a) the total score for the 20 simulations. b) The average roads' width blocked during the 20 simulations. c) The average number of clearing messages during the 20 simulations.

5 Conclusion

In this paper, we have compared a centralized approach with a decentralized approach on two different tasks. Our preliminary results show that the decentralized approach is better when the situation is more chaotic because the center agents do not know the position of each of their agent. Since platoon agents have some problems reaching the goal of the center agent, they obtain better results when they are deciding by themselves, because they can take their distance from the objective more into account.

In the case of a centralized approach, we obtain better results when the agents are free to move, because in those situations, the distance from the goal is not important. Thus, the better global vision of the center can help to choose the best goal.

For the task of rescuing civilians, we have seen that the difference is not significant between the centralized and the decentralized approach. This happens because the information concerning civilians is less dynamic and the messages are received by all agents. So, every *AmbulanceTeam* agent has sensibly the same view as the *FireStation* agent. Therefore, *AmbulanceTeam* agents often make the same decisions as the center agent.

We have also tested in this paper the usefulness of partitioning the environment in sectors. As we have demonstrated with our results, the partitioning of the map is helpful for a task, like clearing the roads, that is very dynamic. In the RoboCupRescue environment, *PoliceForce* agents can receive a lot of messages asking to clear a road and the partitioning help in assigning a specific clearing task to an agent. Each agent knows the tasks that it is responsible for without any communication. So, there is always one and only one agent executing a specific task. The coordination is assured with those sectors and this can be seen as a coordination by social laws.

In short, we have shown in this paper that when a task is complicated, it is useful to use a decentralized decision making process and also if possible to partition the environment in sectors and assigning an agent to be responsible for each sector.

References

1. Kitano, H., Tadokor, S., Noda, H., Matsubara, I., Takhasi, T., Shinjou, A., and Shimada, S.: Robocup-rescue: Search and rescue for large scale disasters as a domain for multi-agent research. In *Proceedings of the IEEE Conference on Systems, Man, and Cybernetics (SMC-99)* (1999)
2. Kitano, H.: Robocup rescue: A grand challenge for multi-agent systems. In *Proceedings of ICMAS-2000* (2000)
3. Lesser, V., Decker, K., Wagner, T., Carver, N., Garvey, A., Horling, B., Neiman, D., Podorozhny, R., NagendraPrasad, M., Raja, A., Vincent, R., Xuan, P., Zhang, X.Q: Evolution of the GPGP/TAEMS Domain-Independent Coordination Framework. In *Autonomous Agents and Multi-Agent Systems (To appear),* Kluwer Academic Publishers (2003)
4. Lizotte, S., and Chaib-draa, B.: Coordination in CE Systems: An Approach Based on the Management of Dependencies Between Agents, *CERA: Concurrent Engineering: Research and Applications*, Vol. 5, number 4 (1997) 367-377
5. Malone, T. W., and Crowston, K.: The Interdisciplinary Study of Coordination. *ACM Computing Surveys*, Vol. 26, number 1, March (1994)
6. Martial, F. V.: Interactions among autonomous planning agents. In Demazeau, Y. and Muller, J.-P., editors, *Decentralized AI*. North Holland (1990) 105–119
7. Mintzberg, H.: *The Structuring of Organizations*. Englewoods Cliffs (1979)
8. Nair, R., Tambe, M., and Marsella, S.: Team Formation for Reformation in Multiagent Domains like RoboCupRescue. In Kaminka, G.; Lima, P.; and Roja, R., eds., *Proceedings of RoboCup-2002 International Symposium*, Lecture Notes in Computer Science. Springer Verlag (2003)
9. Shoham, Y., and Tennenholtz, M.: On the synthesis of useful social laws for artificial agent societies (preliminary report). In *Proceedings of the National Conference on Artificial Intelligence*, San Jose, CA. (1992) 276–281
10. Stone, P., and Veloso, M.: Task Decomposition, Dynamic Role Assignment, and Low-Bandwidth Communication for Real-Time Strategic Teamwork. *Artificial Intelligence* (AIJ), Vol. 100, number 2, June (1999)
11. Sugawara, T., and Lesser, V. R.: Learning to Improve Coordinated Actions in Cooperative Distributed Problem-Solving Environments. *Machine Learning*, Vol. 33 (1998) 129-153
12. Tambe, M.: Towards flexible teamwork. *Journal of Artificial Intelligence Research*, Vol. 7 (1997) 83–124
13. Wooldridge, M.: *An Introduction to MultiAgent Systems*. Wiley (2002)