Learning and Adaptation in Physical Heterogeneous Teams of Robots

Josep Lluis de la Rosa and Israel Muñoz

***Abstract*—In this paper we present a novel approach to assigning roles to robots in a team of physical heterogeneous robots. Its members compete for these roles and get rewards for them. The rewards are used to determine each agent's preferences and which agents are better adapted to the environment. These aspects are included in the decision making process. Agent interactions are modelled using the concept of an ecosystem in which each robot is a species, resulting in emergent behaviour of the whole set of agents. One of the most important features of this approach is its high adaptability. Unlike some other learning techniques, this approach does not need to start a whole exploitation process when the environment changes. All this is exemplified by means of experiments run on a simulator. In addition, the algorithm developed was applied as applied to several teams of robots in order to analyse the impact of heterogeneity in these systems.**

***Index Terms*—Heterogeneity, physical agents, robots, ecosystems, robocup.**

I. Introduction

MULTI-ROBOT Systems have developed extensively in the last years and have drawn the attention of many researchers worldwide. A considerable number of real applications of these systems have been applied to real-life problems ranging from cleaning tasks and foraging to soccer [1].Multi-Robot Systems can be classified as either homogeneous or heterogeneous. Heterogeneity in these systems can be found at two different levels: behavioural and physical. Robots can differ in how they are programmed [2], in other words, how they act depending on the information provided by the sensors. Robots can also be different in their physical features, that is, in their actuators, sensors, shape, size, etc. From the point of view of physical heterogeneity robots may differ in which tasks they are able to accomplish and in how efficiently they perform the same task [3]. Most of the work done so far has focused on homogeneous teams of robots and teams of identical robots with heterogeneous behaviours. However, in recent years heterogeneous robot systems have attracted more and more interest. Teams of heterogeneous robots are often assembled by combining several robots created previously for different purposes. Efforts so far have focused on deploying these systems, but little attention has been paid to the benefits of using these systems and how their full potential might be exploited.

Josep Lluis de la Rosa is with EASY - Centre de la xarxa IT del CIDEM University of Girona, E17071 Girona. E-mail: peplluis.delarosa@eia.udg.es

Israel Muñoz is with Institute for Agro-Food Reserach and Technology (IRTA) and a part-time lecturer at the University of Girona (UdG). E-mail: israel.munoz@irta.es

This paper attempts to address some of the key points of this emerging field. We focus on learning and adaptation in these systems and also analyse the benefits/drawbacks of heterogeneity. The structure of this paper is as follows: section 2 stress the motivation of heterogeneity, section 3 introduces the experimental environment, section 4 contains the ecosystem based model for physical agents as robots, section 5 explains the set of experiments which are analysed in section 6. Finally conclusions are set in section 7.

II. Motivation

Heterogeneous teams of robots raise several questions not posed by homogeneous ones. These issues are:

A. How to exploit the full potential of heterogeneity

In the literature, the potential of these systems is generally exploited through decision making. Most of the systems deployed in real applications assign tasks to robots based on the individual capabilities of each agent. One of the most widespread methods is auctions. This is the case for MURDOCH [4] and ETHNOS [5], in which each robot is aware of their individual capabilities and the decisions are taken considering the capabilities of the various robots available to complete a given task. Tasks are assigned based on a utility value created by each robot using information about their capabilities. These co-operation strategies have two main problems. On one hand, robot capabilities can change over time. Therefore, decisions may be taken based on outdated information, except if the robot is able to update their own physical capabilities over time. This feature is implemented by L-ALLIANCE [3], where robots performing a given task are observed by other team-mates, so that robot capabilities can be evaluated and updated. On the other hand, distributing tasks based on the most appropriate robot for the task do not guarantee the best team performance when there are several tasks to be performed at the same time. One possible way to solve this problem is by allowing agents to learn which is the best task distribution based on the overall team performance. However, the literature on heterogeneous systems does not contain any work on learning in Multi-Robot Systems. In order that robot capabilities do not overlap the decision making goal, the different elements of the system need to be coordinated efficiently. For example, in [6] three robots in charge of assembling a structure co-operate through explicit coordination using a layered architecture. In [7] the movement of two robots with different features is coordinated using the information on their controllers.

B. Benefits of heterogeneous teams over homogeneous ones

The question of the benefits of heterogeneity in robot teams does not have a clear answer. On one hand, the benefits of heterogeneity are evident when all capabilities needed for a specific task can not be built into one robot. However, when robot capabilities overlap, these questions cannot be answered. Only [3] was able to determine, in a group of robots with overlapping capabilities performing several tasks, which performance decreased as heterogeneity increased. The answer to these two questions should help designers create and exploit these systems. This paper addresses the first question in depth by explaining and testing a new algorithm for learning in a heterogeneous multi-robot system. The second question is extensively analysed by summarising the results of applying the algorithm detailed in the first part of the paper to a set of homogeneous and heterogeneous robot teams.

III. Experimental domain

For this research robotic soccer has been selected as the test-bed for several reasons. According to [8] RoboCup is an attempt to foster AI and robotics research using a soccer game as a representative domain in which a wide range of technologies can be integrated and new technologies can be developed. The framework of the RoboCup Physical Agent Challenge provides a good test-bed to see how physical bodies play a significant role in carrying out intelligent behaviours. In addition, we are familiar with this domain, as we have participated over the past years with a team of real robots in several competitions around the world. The experiments presented in this paper have been run on a simulator called JavaSoccer that emulates the Small Size League in RoboCup.

IV. An ecosystem based model for physical agents

The literature on multi-robot and multi-agent systems represents a large amount of research aimed at enabling a group of robots to learn and adapt (see [9] for a survey). However, most of the approaches are not conceived for dealing efficiently with heterogeneous members and, in particular, for learning how to complete a joint task efficiently. One of the particular features of these systems is the size of the state space. As the components are different, the number of possible solutions to the problem increases dramatically. One of the most interesting works related to this research was developed by [10]. In this work, sets of physical components (pump, heater...) learn to participate in the negotiations to execute a given plan based on their physical capabilities. Although this work presents some interesting issues, it does not deal with the complexity of the problem completely. This section leads to the selection of the algorithm for this task.

Quite often researchers have turned to natural phenomena to solve complex problems. One of the most popular sources of inspiration is insect societies. The research done in this area is also known as swarm intelligence and has been applied to solve a huge number of problems (from telecommunications routing to robot control) [11]. This research builds on two different aspects: on one hand, a natural phenomenon that allows task partition in animal and insect societies through competitions among members [12] and, on the other, the work done by [13] on chaos in distributed systems and the model developed as a result.

Heterogeneity is an intrinsic feature of many insect and animal societies. Insects usually live in groups or colonies and work co-operatively towards the same goal. In these societies tasks are partitioned according to the physical abilities of each member of the colony (see [14] for detailed examples). One of the phenomena that allows tasks to be partitioned is competition among the members. Insects and animals (including humans) that live in social groups establish a social structure called dominance hierarchy within their groups. This hierarchy serves to maintain order, reduce conflicts and promote co-operation among group members. Each member of the group has a position within the dominance hierarchy based on the outcomes of interactions with other members. The tasks that each member carries out depend on the position in the hierarchy. In many cases, the resulting hierarchy is affected by the physical features of the group members. This phenomenon offers an example of how nature solves the problem of task allocation in a heterogeneous system. A model has to be defined to emulate this process for the purpose of learning in heterogeneous multi-robot systems. Hogg and Huberman [13] developed a model to study heterogeneous collections of agents competing for the use of bounded resources. Agents select resources depending on the number of agents already using them and compute a perceived pay-off for their use. The number of agents of a given type also increases depending on how well each type of agent is performing with respect to other types of agents. Although this model was developed to study computational ecologies [15], it can be reformulated so that it emulates the competition for responsibilities and tasks in a group of agents depending on how well each member is doing. The next section details how this model has been adapted to deal with the problem of learning in heterogeneous multi-agent systems.

A. Ecosystem description

The main aim of this model is to define how agents change their preferences for performing a given type of task and how they co-operate. This model is also referred to as the ecosystem, as it models some phenomena observed in natural systems. In this ecosystem a set of agents $S = s_1, s_2, s_3, s_4, s_5$ competes for a set of roles $R = r_1, r_2, ..., r_{10}, r_{11}$ (representing a subset of responsibilities). In this explanation subindex r stands for roles and subindex s for agents. Agents in this ecosystem have different physical dynamical features and roles are roles in a soccer team. These agents use the information about the roles in the form of rewards to decide which roles to use. The way these preferences and rewards are combined is represented in a modified model of [13]. This approach also allows the fitness (how well each member is doing when compared with other members) of each species of the ecosystem to be modelled. This fitness affects how agents interact with each other, as well as each agent's preferences.

B. Role definitions

In this domain tasks are roles on a soccer team. Each task contains a limited number of high level actions $(kick_ball, move_ball, defend_goal, cover_goal, regain_ball...)$, some of which are shared by several roles. Rules propose actions to agents depending on the position of the ball and the player on the field who is taking the decision. Each condition is a fuzzy variable, whose values are defined by a fuzzy set. Each rule has a certainty (φ) associated with it $(\phi \in [0,1])$. This value depends on the level of activation of each condition and the operators (*AND/OR*). Conditions and operators are combined together using fuzzy algebra and implemented by means of a possibilistic approach. The roles are: goalie, fullback, left/right defender, defensive midfielder, left/right midfielder, centre forward, left/right winger and central striker. Two or more agents can use one role at the same time, as long as it consists of more than one action. Each agent gets rewards when these roles are used.

C. Creating heterogeneous physical agents

Every agent in this ecosystem has different physical and dynamical features. From experience, big robots are heavier, as they carry more batteries and more material is needed to assemble them. This affects the speed of the robot and its dynamical behaviour. From our experience controller design, controllers tend to be more accurate when they are designed to control a robot that moves slowly, and this feature is also affected by the accuracy of the vision system. On the other hand, when a controller is designed to control (smaller) robots with faster speeds, it tends to be more inaccurate. Bigger robots can also have enough room to contain a powerful kicking device, while this is not so easy for small robots, whose kicking devices are not usually so powerful. Using these ideas, heterogeneous players are created by building robots ranging from fast, inaccurate, and with small kicking devices to small, accurate and with powerful kicking devices (see Table I).

The criteria applied to building the robots allow us to use robots that can contribute in different ways to the team's overall performance, as each robot has unique features.

D. Rewarding actions

Agents use the rewards they get from the action they select to compute their preferences and the fitness of each agent. Rewards are given by taking into account several aspects:

- The player is situated properly on the field according to the role it is fulfilling
- Ability to prevent the ball from going into the goal
- Goals scored
- Goals made
- Dribbling ability
- Ability to regain control of the ball
- Ability to move the ball forward.

  As each agent can use any role at any given time, every tw seconds (in this work 100 seconds) agent s adds all rewards obtained over this period of time for each role r (R_{rs}). Using additional information this agent tries to estimate the possible rewards (G_{rs}) if the given role had been used at each decision step. Several parameters are applied to compute this expected value.
- Each agent keeps track of its decisions and every t_w computes the number of times that role r has been used (P_{rs}), and then uses this value to compute p_{rs} (normalising P_{rs} for agent s).
- μ_{rs} is a confidence parameter that depends on the amount of information available to calculate G_{rs}, see 1. This means that $\mu_{rs} = h(P_{rs})$. $\mu_{rs} \in [0,1]$.

$$G_{rs} = \frac{R_{rs}}{p_{rs}} * \mu_{rs} \tag{1}$$

G_{rs} is also affected by the team's performance. If the goal margin is increasing with respect to the previous results, G_{rs} increases proportionally to each role usage. Otherwise, the values of G_{rs} decrease. This is modelled by the function learn.

$$G_{rs} = learn(G_{rs}, P_{rs}) \tag{2}$$

Next, G_{rs} is rescaled and if the value is under a threshold, values are not considered significant of agent preferences. This is achieved by means of the function *sclp*, which assigns a constant value to G'_{rs} if G_{rs} is below a given value; otherwise, the value G'_{rs} is proportional to G_{rs}.

$$G'_{rs} = sclp(G_{rs}) \tag{3}$$

This function filters out rewards and discards those under a given value, as they are not considered representative of agents' preferences.

E. Modelling agent

Preferences Once each agent knows its rewards it can start the process of updating the model parameters, which are the model agent preferences and the fitness values.

Preferences $-f_{rs}$

Given G'_{rs}, each agent s can compute its current preferences P_{rs} for each role r. This value is used to update $\int_{rs}$, which can be understood as a weighted average of each agent's preferences over time. The average is helpful for several reasons:

- Agents should be able to interact several times with the same roles to evaluate the rewards they get.
- The preferences for each role are not only determined by the physical features of the players and the opponents, but also by the preferences of the rest of the team. This is a dynamic process in which each agent decision affects the rest of the team.

On the other hand:

- Agent preferences should be able to be updated quickly when they change as a result of a change in the environment.

This value is used for the decision making process. High values of $\int_{rs}$ mean a high preference of agent s for role

Agent	Radius(cm)	Translation Velocity(m/s)	Turning Velocity(rad/s)	Kick Speed(m/s)
A1	7.0	0.25	5.28	0.50
A2	6.5	0.30	5.78	0.35
A3	6.0	0.35	6.28	0.30
A4	5.5	0.40	6.78	0.30
A5	5.0	0.45	7.28	0.20

TABLE I
AGENT FEATURES

r resulting in a higher likelihood of using this role. These preferences can also change as a result of the changes in the value N_s.

Fitness $-N_s$

N_s is used to compare each agent's individual performance with respect to the whole team and can be understood as the *fitness* of each agent in the group. Above average values of N_s mean that agent s is better adapted to the environment and getting more rewards from it. This value is updated depending on η_s , the current *fitness* using the latest rewards obtained by each agent over t_w. The motivation for using this average is the same as the one for updating preferences (see Equ. 4,5, 6 and 7).

$$\frac{d\int_{rs}}{dt} = \alpha(N_s p_{rs} - \int_{rs}) \tag{4}$$

$$\frac{dN_{rs}}{dt} = \gamma(\eta_s - N_s) \tag{5}$$

$$p_{rs} = \frac{G'_{rs}}{\sum_{r=1}^{R} G'_{rs}} \tag{6}$$

$$\eta_s = \frac{\sum_{r=1}^{R} G_{rs}}{\sum_{s=1}^{S}\sum_{r=1}^{R} G_{rs}} \tag{7}$$

This model deals with the way decision making is done at two different levels. Each agent s tends to use the roles with the highest preferences ($\int_{rs}$). When two agents think that role r is best for them, then N_s plays a decisive role. This parameter increases/diminishes the values of $\int_{rs}$, so agents with higher N_s tend to win in these conflictive situations, resulting in a hierarchical structure of the system, in which some agents are dominant with respect to others in conflictive situations. This aspect is detailed in depth in the next section: decision making. F. Decision Making Agents co-operate in order to accomplish their goals. This co-operation technique focuses on reducing the number of conflicts between several agents-for instance, preventing two or more agents from taking the same decision. This consensus technique has been used in soccer robotics [16] to reduce the amount of conflicts. The procedure is the following: initially each robot takes the decision with the highest certainty associated with it according to the revision process imposed by the consensus technique for fuzzy rule certainties. If two or more agents intend to take the same decision, the one with the highest certainty wins. This process is repeated until each agent has taken a different action. Agents use their communication capability to exchange information and reach agreement on their decisions. The revision process is based on two parameters, *Prestige (P)* and *Necessity (N)*, which modify the value of certainties (φ). P is related to each agent's preferences. Both parameters have unique values depending on each agent. Here both values are used without a subindex for the sake of the explanation (see later on in this section for more details).

Prestige performs a linear transformation over φ , as described in (8).

$$\varphi' = P(\varphi) = P * \varphi \tag{8}$$

Then, *Necessity* performs a non-linear transformation over φ' as described in (9).

$$\varphi'' = N(P(\varphi)) \tag{9}$$

Prestige can be understood as the confidence that an agent has in one role. Higher values of Prestige mean high confidence values for these roles. Prestige is a conservative parameter. If an agent has a low confidence in a role, the Prestige value will be very low. Here this value is assimilated to the preferences of each agent.

Necessity is a parameter that increases φ' according to the necessity of the information source that is being revised. This is a non-linear parameter that prevents agents from settling in one/several roles too quickly.

In order to deal with the idea of preference for one role, the value of $\int_{rs}$ is used to assign the parameter Prestige (P) for the certainty revision process (see (10)).

$$P = \frac{1}{1 + \exp(\frac{\frac{1}{N_p layers * N_r ols} - \int_{rs}}{k})} \tag{10}$$

where k is a constant value.

The values of Necessity are computed depending on the use of roles. We use this parameter to make sure that the robot initially uses all the roles. As the agent interacts with the roles and starts to show more preference for one or more roles, the effects of this parameter tend to disappear (see (11)).

$$N_{rs} = \exp(-\delta * \int_{s}^{total} * \frac{p_{rs}}{\int_{rs}}) \tag{11}$$

where δ is a constant value and $\int_{s}^{total} = \sum_{r=1}^{R} \int_{rs}$

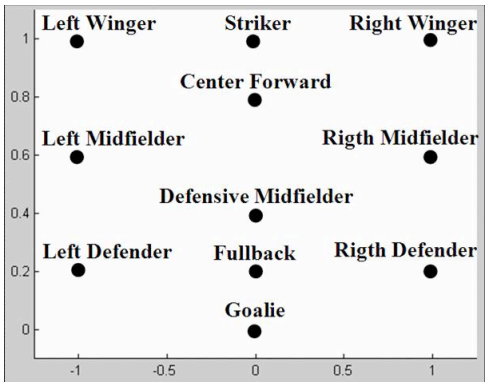

Fig. 1. Team representation

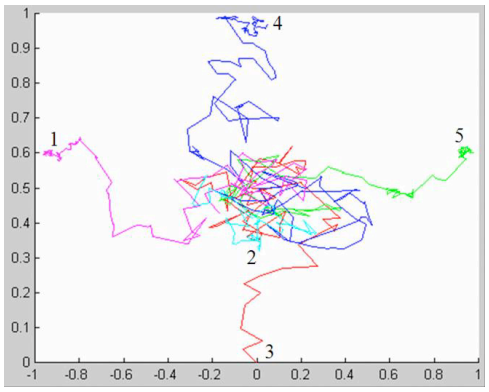

Fig. 2. Role Usage

V. Experimentation

This section describes how the algorithm works in two different situations: learning and adaptation. These two processes are analyzed through several examples and the results are presented.

One of the most interesting features of this algorithm is that it not only allows a group of heterogeneous components to learn how to perform a joint task co-operatively, it also allows a group of agents to adapt to changes in the environment while working as a team. In order to illustrate the explanations a simplified view of the robot decisions has been developed. Robots are represented according to their role usage, and each role has a position on the field. An average position on the field is determined based on the role usage of each robot. Fig. 1 displays the typical positions defined for each role.

A. Learning

Learning takes place through a self-organising process, in which agents interact with the environment and other team-mates. Team-mates interact through co-operation and competition to fill roles. The result is an emergent process in which agents converge to roles.

Fig. 2 shows how the five robots used in the experiments carry out the roles. Two different phases can be perceived in this learning process: exploration and consolidation of agent preferences. During the first phase agents interact with the environment and tend to fill all the roles several times, which can be observed in the middle field (in order to have a clearer view of the results, the information represented is the average of several periods). As initially, preferences are very similar, the exploration of the environment is determined by Necessity, one of the parameters that defines the consensus process. Using this parameter ensures that all agents interact several times with all roles. At this point in the process the team has chaotic behaviour, as agents start using a given role but, a few seconds later, switch to a different one. Agents start to focus on a small number of roles, even though they are still exploring the environment. After a given time, they usually focus on two or three different roles, as their preferences for them are slightly higher than for the rest of the roles. One of the things that usually takes place at this point in the game is that several agents may become interested in the same role. This is due to the same factors detailed previously. This gives rise to competition for roles, which emulates the competition for limited resources that can be observed in animal societies. Agents that have a similar interest in the same role tend to fill the same role in turn, as the Necessity parameter allows agents to explore other roles when the preferences are still very similar. The result of this process depends on the following factors:

- *Opponent's skills*. The disorganised behaviour of the system may keep the team from reaching attacking positions, for instance, if the opponent has very good attacking skills. This may force agents to focus on defensive/midfield roles at the beginning.
- *Physical features*. Each robot's physical features determine the number of rewards that each player can obtain; the skills needed to fill a role usually depend on the opponent.
- *Other team-mates*. Other team-mates often have a similar preference for the same role. This situation can make it more difficult for other agents to use a given role, as the number of conflicts increase (two or more robots decide to choose the same action in the same field area). This may decrease the probability that other players select this role.

During the last phase in the learning process, agents tend to increase their preferences for the role that they were using most of the time at the end of the previous phase. In some cases, some players may finally switch to a different role as a result of the increased order in the system. At the end of this process a set of agents is assigned to different roles. The whole process leads the team to develop a team formation adapted to the opponent with very efficient role distribution among the heterogeneous players. Results show that the result of the learning process is that the team performs, on average, around 90 % of the best hand-coded solution. However, this result can be obtained in about 30 minutes of interaction, while the hand-coded solution may take several days, as the designer needs to know which robots are best suited to a given position and a given opponent. If the opponent changes the designer has to start the whole process again.

Fig. 3 shows how the score for a learning team and a hand-coded team changes over time. The learning team is

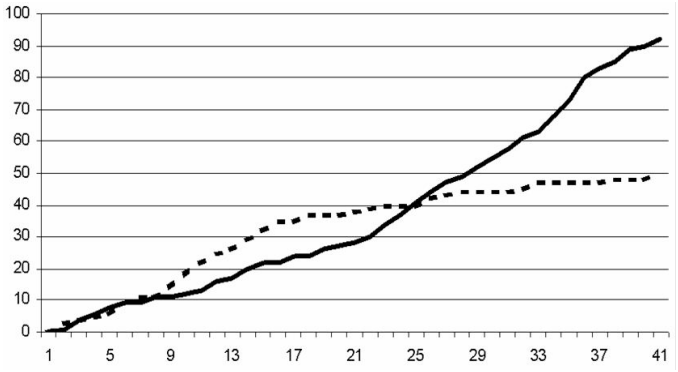

Fig. 3. Learning process

represented by the black line and the opponent by the dotted line. Although initially the opponent performs better than the learning team and the goal margin is increasing, the learning team is able to self-organise and increase the number of goals scored, while the opponent is only able to score a few goals after step 20.

VI. Adaptation

One of most interesting features of this algorithm is that it allows a team of agents to adapt/develop when it is working and performing reasonably well. The key for this feature is the concept of fitness (N_s) and the function *sclp*. As the environment evolves (changes in the opponent's behaviour or in physical features of a robot), the amount of rewards that each agent obtains from the environment also changes, thus modifying the preference and fitness for each agent. In some cases, agents do not get enough rewards to keep their current role preferences (due to the filter imposed by *sclp*). In this situation, these agents start to explore the environment again, while the other members continue to fill the same roles. They tend to settle into a different role, thus improving the team performance in the new environment. Fitness contributes to adaptation, especially when the physical features of one of the members change as a result of a mechanical failure.

The following example analyses how the team is able to adapt when one of the team components breaks down, as in example A1. A1 is filling the role of striker when its speed falls from 0.45 m/s to 0.15 m/s and its rotational speed from 7.28 rad/s to 5.28 rad/s. Its performance falls from a goal margin of 80 goals to 30 goals. A3 and A2 are filling the roles of left and right midfielder, while A5 fills the role of goalie and A4 that of defensive midfielder (see Fig. 4).

After the robot failure the team starts to evolve thanks to the change in fitness. A2 tends to move towards more offensive positions, while the other agents, except A3 which also plays in attacking positions, keep their current positions. The reason for these changes is that A1 is not able to play its striker position efficiently any more, as a result of which its rewards decrease, and thus its preferences for this role and the fitness. This allows A2, and also A3, to participate more often in attacking positions. This change in the team configuration results in a better team performance, which jumps to a goal margin of 60 (see Fig. 5).

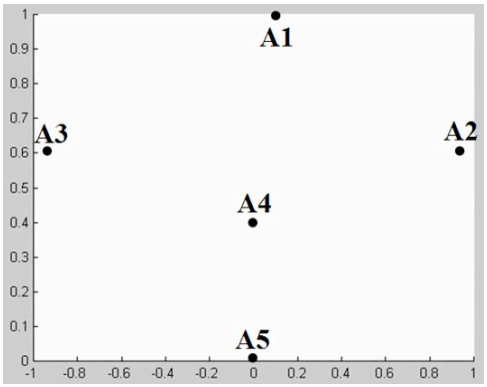

Fig. 4. Team adaptation 1

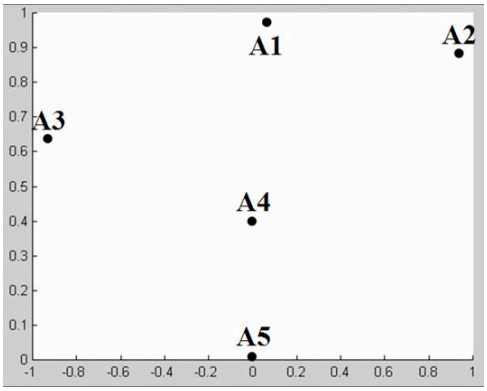

Fig. 5. Team adaptation 2

Finally A2 replaces A1 in the role of striker and A1 fills the role that A2 was filling. As A2 is damaged, A4 helps it by playing on the right side of the field. As a result of these changes in team configuration, the performance jumps to a goal margin of 76 goals (see Fig. 6). This example shows that the team is able to adapt while playing efficiently.

If the same fitness value is considered for each robot during the entire process, the system is not able to fully adapt, that is, the performance improves only slightly. This is explained by the fact that the decrease in A1's fitness allows other robots to participate in A1's tasks in attacking positions, as the other members tend to win more conflicts with A1, in this example

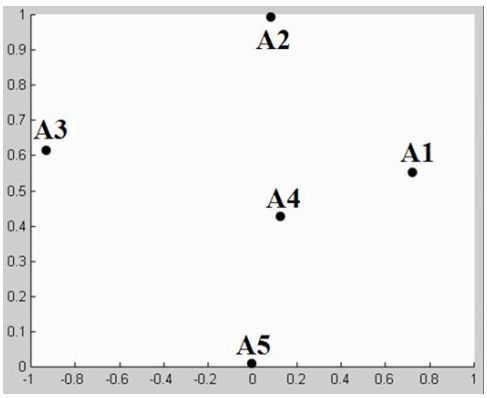

Fig. 6. Team adaptation 3

A2. As this robot tends to fills offensive roles more often, the overall team performance increases, thus increasing the rewards for A2 in the attacking roles.

VII. Heterogeneity analysis

The algorithm developed has been applied to a set of homogenous and heterogeneous teams to analyze the benefits/drawbacks of heterogeneity. Five homogenous and five heterogeneous teams were built. Heterogeneous teams were composed of a combination of the robots detailed in Table I, while homogenous teams were composed of 5 identical robots of each of the types described. These teams played against 3 hand-coded teams that represent 3 different levels of difficulty (easy, intermediate and difficult). Eighty simulations were done for each case. Table II details the results of the simulations as a goal margin. The best team against a given opponent is assigned a 1 and the worst team a 0.

Results show that heterogeneous teams generally perform better than the homogeneous ones: 4 out of 5 heterogeneous teams perform better than the homogeneous teams. Analysing each case individually, in two cases a homogeneous team performed better than any heterogeneous team. Although H5 is the best team against the third team, it is the worst against the first and second teams. Heterogeneous teams tend to perform more uniformly than homogenous ones. Their performances show a considerably smaller variance. An in-depth analysis of the results shows the following:

- *Task specialization*. Physical features demanded for a given role depend on the tasks that must be fulfilled. Heterogeneous teams present a wider range of capabilities. Thus, a correct distribution of these capabilities may result in better efficiency when the team performs the different tasks.
- *Flexibility*. One of the features observed in the results is that role distribution in a team of heterogeneous components changes depending on the opponent. This means that the features demanded for each role also depend on the opponent. Thus, heterogeneous teams, on average, tend to perform better because they can combine their capabilities in different ways. Another feature observed in heterogeneous systems is the surprising results of combining skills when the components of the team cooperate.
- *Skills combination*. There are several examples (HT1 and HT2, HT3 and HT4) where replacing a robot that individually performs worse against the same opponent results in a poorer team performance.

VIII. Conclusions and future work

This paper has presented a novel approach to allow a team of heterogeneous members to select the best roles for them based on the other team-mates and the environment. In addition, the algorithm detailed in this paper allows the team of robots to adapt while still performing efficiently. Finally, this algorithm was applied to several teams of homogeneous and heterogeneous robots. The results have shed some light on the question of heterogeneity in multi-robot systems. We will extend this algorithm to other domains and other problems. One of these new domains is Rescue [17], in which mainly heterogeneous teams of robots are deployed to rescue people after a catastrophe. The algorithm can also be improved so that tight co-operation can be learnt. This model will be tested using other decision-making processes.

Acknowledgment

This work was supported by the Spanish MCYT project DPI2007-66872-C02-01, Arquitecturas Multiagente de Control e Interaccion de Robots Móviles en su Aplicación al Rescate de Supervivientes de Catástrofes con Agua and by EU project No. 34744 ONE: Open Negotiation Environment, FP6-2005-IST-5, ICT-for Networked Businesses.

References

[1] L.E. Parker, "Current State of the Art in Distributed Autonomous Mobile Robotics", Distributed Autonomous Robotic Systems 4, Lynne E. Parker, George Bekey, and Jacob Barhen (eds.), Springer, 2000: 3-12.
[2] T. Balch, "Behavioral Diversity in Learning Robot Teams", PhD Thesis Georgia Tech 1998.
[3] L. E. Parker, "Heterogeneous Multi-Robot Co-operation", PhD Thesis MIT, May 1994.
[4] B.P. Gerkey and M.J. Mataric, "Murdoch: Publish/Subscribe Task Allocation for Heterogeneous Agents", Proceedings Agents 2000, Barcelona, Spain, June 2000.
[5] C. Castelpietra, L. Iocchi, D. Nardi, M. Piaggio, "Communication and Coordination among Heterogeneous Mid-size Players: ART99", In Proceedings of the 4th International Workshop on RoboCup, pages 149-158, Melbourne, Australia, August 2000.
[6] R. Simmons, S. Singh, J. Ramos and T. Smith, "Coordination of Heterogeneous Robots for Large-Scale Assembly", Proceedings of the International Symposium on Experimental Robotics (ISER), Hawaii, December 2000.
[7] L. Chaimowicz, T. Sugar, V. Kumar and M.F. Campos, "An Architecture for Tightly Coupled Multi-Robot Cooperation", Proceedings of the 2001 IEEE International Conference on Robotics and Automation, pp. 2292-2297. Seoul - Korea, May, 2001.
[8] H. Kitano, P. Stone, M. Veloso, S. Coradeschi, E. Osawa and H. Matsubara, "The RoboCup Synthetic AgentChallenge 97", XV IJCAI-97 International Joint Conference on Artificial Intelligence, Vol. 1, pp.24-29, Nagoya, August 1997.
[9] P. Stone and M. Veloso, "Multiagent Systems: A Survey from a Machine Learning Perspective", Autonomous Robotics, volume 8, number 3, July 2000.
[10] M.V. Nagendra, V.R. Lesser, and S.E. Lander, "Learning organisational roles in a heterogeneous multi-agent system" In Adaptation, Coevolution and Learning in Multiagent Systems: Papers from the 1996 AAAI Spring Symposium, pages 72-77, Menlo Park, CA, March 1996. AAAI Press.
[11] E. Bonabeau, M. Dorigo and J-L Theraulaz, "Swarm Intelligence: From Natural to Artificial Systems", Oxford University Press.
[12] Bonabeu, Theraulaz and Deneubourg, "Dominance orders in animal societies: the self-organisation revisited", Bulletin of Mathematical Biology, 1999.
[13] T. Hogg and B.A. Huberman, "Controlling Chaos in Distributed Systems", IEEE Transactions on Systems, Man, and Cybernetics, Vol. 21, No. 6, November/December 1991.
[14] F.L.W. Ratnieks and C. Anderson, "Task partitioning in Insect Societies", Insectes Sociaux, 46 (1999) pp 95-108.
[15] B.A. Huberman and T. Hogg, "The Emergence of Computational Ecosystems", 1992 Lectures in Complex Systems, pp185-205, SFI Studies in the Sciences of Complexity, Volume V, Addison-Wesley, 1993.
[16] J. Ll. de la Rosa, A. Oller, J. Veh and J. Pujol, "Soccer Team based on Agent-Oriented Programming", Robotics and Autonomous Systems. Ed. Elsevier, Vol. 21, pp.167-176, October 1997.
[17] H. Kitano, S. Tadokoro, I. Noda, H. Matsubara, T. Takahashi, A. Shinjou and S. Shimada, "RoboCup Rescue: Search and Rescue in Large-Scale Disasters as a Domain for Autonomous Agents Research", Systems, Man, and Cybernetics, 1999. IEEE SMC '99 Conference Proceedings, Vol. 6, 1999 Page(s): 739 -743

Team	**Performance 1 Team**	**Performance 2 Team**	**Performance 2 Team**	**Average**
H1	0.21989529	0.78571429	0	0.3352
H2	0.76963351	0.71428571	0.07476636	0.5195
H3	1	0.39285714	0.4953271	0.6293
H4	0.54973822	0.46428571	0.8411215	0.6183
H5	0	0	1	0.3333
HT1	0.7591623	0.91071429	0.8411215	0.8369
HT2	0.62827225	0.98214286	0.70093458	0.7704
HT3	0.68062827	0.39285714	0.80841121	0.6272
HT4	0.58638743	1	0.91121495	0.8325
HT5	0.68062827	0.71428571	0.57943925	0.6581

TABLE II
SIMULATION RESULTS