Multi-Agent Task Allocation for Robot Soccer

Khashayar R. Baghaei and Arvin Agah

Department of Electrical Engineering and Computer Science
The University of Kansas, Lawrence, KS 66045 USA

ABSTRACT

This paper models and analyzes task allocation methodologies for multi-agent systems. The evaluation process was implemented as a collection of simulated soccer matches. A soccer-simulation software package was used as the test-bed as it provided the necessary features for implementing and testing the methodologies. The methodologies were tested through competitions with a number of available soccer strategies. Soccer game scores, communication, robustness, fault-tolerance, and replanning capabilities were the parameters used as the evaluation criteria for the multi-agent systems.

KEYWORDS

Multi-agent systems, task allocation, distributed robotics, robot soccer.

1. INTRODUCTION

The work presented in this paper is a study of dynamic task allocation methodologies for mobile multi-robot systems (Cao *et al.*, 1995; Arkin & Balch, 1998) in a soccer environment. Soccer is chosen as the test-bed because it provides a very complex, dynamic and, at times, hostile environment. This dynamic environment is a fast paced environment that

Reprint requests to: Khashayar R. Baghaei, Department of Electrical Engineering and Computer Science The University of Kansas, Lawrence, KS 66045 USA; e-mail: kbaghaei@hotmail.com

allows limited time for each robot to act. Successful robotic teams must rapidly adapt to environmental changes, for which both individual robot behaviors and team strategy should be adaptive. The evaluation process was implemented as a collection of simulated soccer matches using the TeamBots software (TeamBots, 2007). Game scores, communication (Corkill, 1991; Werner & Dyer, 1991; Yanco & Stein, 1993; Yanco, 1993), robustness, fault-tolerance, and replanning required for dynamic techniques were used as the evaluation criteria.

2. MULTI-ROBOT SYSTEMS

One of the important recent trends in robotics is the study of teams of multi-robot systems. Research performed under such titles as distributed robotic systems, swarm robotics, socio-robotics, decentralized robotics, multi-agent robotics, and cellular robotics has focused on the investigation of the issues and applications of systems composed of groups of robots. The general idea is that teams of robots, deployed to achieve a common goal, can only perform tasks that a single robot cannot but also can outperform systems of individual robot in terms of efficiency and quality. In addition, groups of robots provide a level of robustness, fault-tolerance, and flexibility, as the failure of one robot does not result in the unsuccessfulness of the mission, as long as the remaining robots share the tasks of the failed robot. Examples of tasks appropriate for robot teams are large area surveillance, environmental monitoring, autonomous reconnaissance, large object transportation, planetary exploration, and hazardous waste cleanup (Agah & Tanie, 1997; Fredslund & Mataric, 2001; Sukthankar, 2000; Sukthanker & Sycara, 2001).

The most significant concept in multi-robot systems is cooperation. Only through cooperative task performance can the superiority of robot groups be demonstrated. The cooperation of robots in a group can be classified into two categories of implicit cooperation and explicit cooperation. In implicit cooperation, each robot performs individual tasks while the collection of these tasks is toward a unified mission. For example, when multiple robots are engaged in collecting rock samples and returning them to a common place, the team is accomplishing a global mission while cooperating

implicitly. This type of group behavior is also called asynchronous cooperation, as it requires no synchronization in time or space. Explicit cooperation is when robots in a team work synchronously with respect to time or space to achieve a goal. One example of such cooperation is the transportation of heavy objects by multiple robots, each having to contribute to the lifting and moving of the object. This task requires the robots to be positioned suitably with respect to each other and to function simultaneously. Regardless of the type of cooperation, the goal of the team must be transformed into tasks to be allocated to the individual robots.

Multi-robot teamwork is a complex problem (Lerman, 2000; Lerman & Gastyan, 2001; Sgorbissa & Arkin, 2003; Werger & Mataric, 2001) consisting of task division, task allocation (Shehory & Kraus, 1998), coordination, and communication. Dudek *et al.* (1993) present a general taxonomy to characterize multi-agent systems, consisting of the number of agents, communication (range, bandwidth and topology, reconfigurability, processing mechanism, and differentiation).

3. EVALUATION OF TASK ALLOCATION METHODOLOGIES

One main issue in task allocation is the division of the tasks into homogeneous versus heterogeneous tasks. Another main issue in task allocation is the study of multi-robot systems in hardware with small population sizes (e.g., under 20), versus the study of issues in multi-robot systems in simulation with large population sizes. Task allocation methodologies for multi-robot systems include the following: Murdoch Publish/Subscribe System (Gerkey & Mataric, 2000, 2002a, 2002b), Broadcast of Local Eligibility (BLE) using Port Arbitration Behavior (PAB) (Werger & Mataric, 1999, 2000a, 2000b), A Free Market Architecture for Distributed Control of a Multi-Robot System (Dias & Stentz, 2000; Stentz & Dias, 1999), Auction Algorithm (Bertsekas, 1992), Alliance (Parker, 1993, 1994, 1996, 1997a, 1999, 2001), Task Acquisition using Multiple Objective Behavior Coordination (Pirjanian & Christensen, 1998), Functionally-Accurate Cooperative (FA/C) Distributed Problem Solving (Lesser, 1991), Distributed Multi-Robot Task Allocation for Emergency Handling (Ostergaard *et al.*,

2001), Team Formation-Based Task Allocation (Stone & Veloso, 1999), Ants Algorithms (Dorigo & Gambardella, 1996; Dorigo *et al.*, 1996), and Territorial Task Division (Schneider-Fontan & Mataric, 1998).

Any evaluation study of multi-robot systems must be expressed quantitatively using suitable criteria. Because many approaches to multi-robotic systems are focused on a special domain and application, there are no unified evaluation methods for comparing these approaches. However, a few patterns occur in all of these systems, which provide some mean to compare and justify each approach. These evaluation criteria can guide the selection of the most appropriate methods for new application domains for robot teams.

Researchers (Balch & Arkin, 1994; Balch & Parker, 2000; Balch & Ram, 1998) have categorized four metrics for comparing the effects of communication on performance of multi-robot systems in different applications:

- Cost: It minimizes the number of deployed robots and the cost of building of each of them.
- Time: The time required to complete the mission.
- Energy: The total energy consumed by the robots.
- Reliability/Survivability: The ability of team to survive and provide reliable performance.

Parker (2001) reported a new set of metrics, which used eight parameters to evaluate a multi-robot system operating in a specific environment, performing a known set of tasks such as moving objects and observing targets. Such metrics introduced to evaluate multi-robot systems are not general purpose and cannot be applied to a wide range of applications.

1. Time
2. Energy
3. Task per time
4. Velocity (distance/ time.
5. Number of objects moved per unit time
6. Average number of targets observed
7. Quantity of earth moved per unit time
8. Cumulative formation error

Goldberg and Mataric (1997, 2000, 2002) used the three application-based criteria for evaluating and comparing different multi-robot systems' performance:

1. Inter-robot collision
2. Distance traveled by each robot
3. Time

One of major characteristics of multi-robot systems is the heterogeneity versus homogeneity the distributed system. A number of factors have great influence on making a team of robots heterogeneous, such as sensor tuning, hardware design, software design, system wear-ability, or even different experiences of robots capable of learning (Parker, 2001). A team may include different number of robots of different types. A team made of different robots can benefit from the diversity of robots' capabilities.

The heterogeneous robot teams can be divided in two major categories of functional heterogeneity, where all team members are not capable of performing all the require tasks and some expert robots exist, and behavioral heterogeneity, where the robot behave differently. Another concept is pseudo-heterogeneity, where during task allocation some tasks are assigned based on different initial condition of the robots such as their current location or speed. Additionally, by assigning tasks based on the priority of the robot or its ID, one can design a simple heterogeneous system composed of many robots.

Goldberg and Mataric (2000) defined robustness as the ability of completing the global task in presence of partial or complete failure of parts of the group. This is not a directly measurable quantity. One way for relating robustness to a measurable quantity is to compare time of completing a predefined task with or without the presence of partial failure for a variety of different architectures.

4. ROBOTIC SOCCER

Robotic soccer, a challenging and difficult task, is a suitable domain for experimenting with new methodologies and concepts in multi-robotic systems. The important characteristics of robotic soccer are (1) its

deterministic character that provides measurable feedback about performance in contrast to vaguely defined domains, (2) its complexity, and (3) its dynamic features. Robotic soccer integrates works in research areas such as computer vision, intelligent control, interaction and cooperation, robot mechanics, power systems, real-time systems, machine learning, planning and plan recognition, and competition all in one domain. Many researchers choose to compete in robotic soccer simulator leagues, since physical robots can be expensive and complicated to design and build. Simulation of robots prior to their physical construction can help to evaluate, analyze, modify, and improve the designs. Since the focus of the work presented in this paper is to evaluate different dynamic task allocation methodologies for multi-robot applications, soccer simulation provides a very suitable environment to experiment and analyze new ideas in multi-robot systems. Research areas in soccer simulation include the following:

- Dynamic resource allocation for heterogeneous robots. Given the different skills that the robots possess, and given that robots use resources such as energy, how should they best divide their tasks among them?
- Multi-robot modeling. How can one recognize its own team's and the opponent's strengths and weaknesses? How can the behaviors be modeled?
- Machine learning (environment modeling). The system tries to model the behavior of other robots through observation and examination. The simulator can use a coach robot to build a model of the behavior of a team from observations of the team.
- Machine learning (skills). How can a robot improve its performance using its own experiences gained through interactions with the environment?
- Adjustable autonomy. How do robots decide on interpreting and acting on the coach's advice or supplied information?
- Teamwork and coordination. How can a group of robots work together, collaborating and coordinating, as an effective team? How do small sub-teams form dynamically, and how can they be made effective?
- Adversarial planning. How can the robots plan to react to the opponents' behavior?
- Robot architectures. What robot architectures are useful for dynamic, complex, multi-robot systems?

4.1 RoboCup

RoboCup (2007) is an international research event to support research in artificial intelligence and robotics. RoboCup is a game of soccer in which robots are players. Teams that pass the qualification exam and enter the actual game show a great deal of pioneer technology in at least one of the following areas: new materials, sensors, artificial muscle, artificial intelligence, intelligent robotics, highly efficient battery, energy saving systems, control, multi-agent cooperation and coordination, etc. Communications between the robots is wireless and typically uses dedicated commercial FM transmitter and receiver units. RoboCup includes four different competitions, including the simulation league, small-size robot league (field the size of a ping-pong table), middle-size robot league and the Sony legged robot league (four-legged robots). To compete in RoboCup, the robots must possess two general attributes of robustness and safety. This attribute ensures that the robots are not in danger of being damages as the result of collisions with legal objects in the field, and that the robots do not damage other robots. Global vision systems are not allowed and all components of the sensing system must be onboard the robots, and the robots cannot place any landmark on the field.

One RoboCup competition is the simulation league, a league of simulated soccer matches. The official simulation software for RoboCup is called Soccer server. In Soccerserver, teams have 11 players, each has a specified robotic controller to provide its characteristic and to support the team's strategy, and every simulated team actually consists of a collection of these programs. Many computers are networked together for this competition to take place. The online coach competition is part of Soccerserver, for which coaches cooperate and guide different teams utilizing the standard coaching language (Werger & Mataric, 2001). The goal is to provide the required information for more adaptive teamwork by opponent modeling. The coaches have access to the history of each game played in the past and can analyze and observe them.

4.2 FIRA

FIRA (2007) is an international event for robotic soccer that includes different leagues, such as MiroSot (micro robot world cup soccer tournament),

NanoSot, HuroSot, and KheperaSot, in which each league indicates the different sizes of the robots participating. The simulation league is used for providing game training and strategy learning environment and for testing the feasibility and advancement of the game strategy. Simulation software uses kinematics and dynamics to simulate movements of the robots and the soccer ball.

4.3 TeamBots

TeamBots (2007) is a popular soccer simulation software. TeamBots is a collection of packages for mobile robot and multi-robotic programming. Its simulation software is written in Java and its source code is available. An important aspect of TeamBots is its capability to provide an interface similar to real robots' control system so that a program written for the simulation can run on actual mobile robots with minimal change. TeamBots applications can run on a variety of robots including the Nomadic Technologies' Nomad 150 robot and Personal Robotics' Cye robots. The simulation environment can embed heterogeneous multi-robots and can run their control system simultaneously. TeamBots is portable and it can run on a variety of platforms such as Windows, Linux, MacOS.

TeamBots was selected and used for the simulation of the experiments reported in this paper, because of its hardware interface that makes the code independent of low-level sensor fusion and motor control. In addition, it has the most similarity to real robotic development, which has resulted in TeamBots being a popular simulator in robotics research.

TeamBots provides a group of Java classes called *Clay* that encapsulates the functional requirements of writing behavior-based control systems. Clay inherits the advantages of Java that enables it to combine, abstract, and mix behaviors. Clay is a powerful and flexible package for creating a broad range of complex controls. Clay was used to program robot controllers for the teams of simulated robots used in this paper. Perceptual schemas embedded in motor schemas, were developed to process sensory data.

SoccerBots, a part of the TeamBots software distribution, is a research tool for multi-robot application in a soccer environment. SoccerBots is an attempt to integrate the software and hardware aspects of robotics technology. The software is also an attempt to separate robotics software control (behaviors, learning, etc.) from robotics hardware architecture.

4.4 Other Soccer Simulation Systems

A number of other soccer simulation systems provide the users with the ability to play simulated soccer.

MissionLab, developed at the Georgia Institute of Technology (Atlanta, GA, USA., provides unique features via its software components, including a console-like program for monitoring the progress of runs, a graphical tool for building robot behaviors, compilers, and a hardware server that directly controls all the robot hardware and provides a standard interface for all the robots and sensors. In addition, new concepts such as Q-learning and specialized reinforcement learning are supported. One interesting feature of this simulation is its real robot interface, which allows generated missions of simulation to compile and transform to C++ codes. The C++ programs are in a structure that can run on common robots such as ActivMedia Pioneer AT, RWI Urban Robot, and Nomadics Technologies Nomad 150 and 200.

Stage is a scaleable multiple robot simulator, developed at the University of Southern California interactive lab (Los Angeles, CA, USA). Stage simulates a collection of mobile robots moving in and sensing a two-dimensional bitmapped environment. Various sensor models are provided, including laser rangefinder, sonar, odometer, and pan-tilt-zoom camera. Slight change is required to move from simulation to hardware, and back. Several controllers designed in Stage have been demonstrated to work on real robots.

Swarm, developed at the Santa Fe Institute (Santa Fe, NM, USA) using Objective C, is used for the simulation of complex systems. Its architecture allows simulating a set of robots interacting with others simultaneously (Swarm, 2007).

TCA (Task Control Architecture), developed at CMU (Carnegie Mellon University, Pittsburgh, PA, USA), provides a general-purpose framework for building robot systems that combine deliberative and reactive control (TCA, 2007).

5. RESEARCH APPROACH

This section addresses the research approach reported in this paper and tackles the questions defined by Lesser (1995), with respect to the important aspects of multi-agent systems:

1. How to formulate, decompose, and allocate the problem?
2. What is the communication mechanism provided as part of a solution?
3. How to provide effective teamwork and how to solve potential conflicts?

Selected research approaches to generate cooperation in multi-robot soccer teams using dynamic task allocation methodologies are described in this section. Noteworthily, two decisions regarding the architecture in the design of multi-robot systems were made. At the group level, distributed architecture was chosen because the nature of the selected application is distributed and requires more than one robot player. At the robot level, control was divided into two architectures. The schema-based approach was used for low-level parallel controls, and the behaviors as collection of simple behaviors using subsumption architecture were selected. Because each robot behavior can be viewed as an independent agent, behavior-based robotics (Brooks, 1986, 1990; Pirjanian, 1998a, 1998b) facilitates developing robotic controls to study multi-agent task allocation systems in robotic environment. This concept was used in the implementation of the selected task allocation systems.

5.1 Robotic Architectures

Robotic architectures are structures for developing robot controls that impose restrictions on the solution domain by limiting it to an architectural definition. One of the main approaches to robot control is the traditional deliberative strategy, a top-down design that utilizes a central model of the world. Each robot uses this model to process sensory input data and produce actions based on it. As the result of the actions, the robot updates its world model to match the new changes in the environment (Moravec & Cho, 1989). Dynamic environments such as real world applications are too uncertain, and

changes are too common so that replanning is always necessary. Additionally shown is that they cannot scale very well in dynamic applications (Brooks, 1991).

There is another mechanism for robotic control, called reactive. Unlike deliberative strategy, reactive controls are bottom-up designs that do not use world representations. Such designs are different from other artificial intelligence programs because they do not perform any search. However, reactive systems are designed to use a set of condition-action pairs, which is similar to some expert systems. Reactive systems use pairs of data that map sensory input to actuator output. This mapping can have different structures but they serve the same purpose (Brooks & Connell, 1986; Connell, 1990). The major weakness of the reactive control is its inflexibility. Complex and dynamic environments and even a complex robot control can overwhelm the adaptability and flexibility of its hard-coded rules (Mataric, 1997). Indeed, a trade-off between on-line computation and stored information distinguishes these two approaches from each other.

There is a third approach called hybrid, which benefits from both reactive and deliberative methods. Hybrid uses a deliberative method for high-level tasks and uses reactive method for low-level and urgent tasks. This architecture has many sub-control systems like RAP (reactive planning). (Arkin, 1987), contingency plans (Connell, 1992), schemas (Balch & Arkin, 1998), and so on. Behavior-based control, and more specifically schema-based architecture, was chosen because of its unique features that are explained in later sections.

5.2 Behavior-Based Robotics

Behavior-based robotics was selected as the robotic control paradigm in this paper. First introduced by Brooks (1986, 1990, 1991), behaviors are basic units for control, representation, and learning for abstracting robot architecture. These units of control law must satisfy a set of constrains to achieve and maintain their goals. Because of its close relation with multi-agent systems, this approach facilitates developing robotic controls that enable the study of multi-agent task allocation systems in a robotic environment. This relation is rooted in the knowledge that each behavior can

be viewed as an independent agent. This concept has been used in the implementations of selected task allocation systems. Behavior-based control is an extension of reactive architecture, which also has certain features of the deliberative approach. It is possible to store world presentation and state in a behavior control. Nevertheless, that does not mean that it uses a central model or representation. Although, it has some reactive components, behavior-based control is usually more than just a table look-up like reactive methods.

One of the main features of behavior-based control is the occurrence of emergent behaviors. Emergent behaviors are resulted as interaction of different lower-level behaviors. They are usually unpredicted and can be collective tasks. The conceptual approach of behavior-based robotic suggests a hierarchical and distributed control to solving behavior arbitration. The same concept was used in behavior-based approaches by running behavior validation tests in a top-down manner.

5.3 Motor-Schema

Researchers in the field of psychology have originated the concepts of schemas, but in the present paper, the definition of schema from application of schemas in brain theory and robotics is used. Schemas are adaptive control systems that use their sensory data to update the systems behaviors. Arkin (1987) defines a schema as the basic unit of behavior from which complex actions can be constructed. A schema consists of the knowledge of how to act or perceive as well as the computational process by which it is enacted. Motor-Schema is a control architecture that has biological roots; it is a higher-level abstraction for robot control that is similar to subsumption architecture (Arkin, 1989). Schemas define behaviors as concurrent robot controls and they are stored as pairs of (sense, react) rules. According to Arkin, (1987, 1989), schema behaviors are large grain abstractions that can be reused as needed. Additionally, schema behaviors act distributed and concurrently and have cognitive support.

Motor schemas are different form other behavioral controls in the following manners:

- Behavior output is computed as vectors using potential fields approach.
- Coordination can be achieved as a result of vector manipulations.

- Behaviors do not have a priori hierarchy.
- Each behavior plays a rule to produce a global output. This share or strength is a scalar multiplier that is applied to the output vector.

Perceptual schemas provide environmental data for individual behaviors and are usually included in motor-schemas. Perceptual schemas are defined recursively. It means that each of them is made of other sub-perceptual schemas. Integrated actuator output or global robot action is made of motor-schemas running concurrently. Each schema's output (vector) is multiplied by a gain. Gain is a scalar value, which describes the importance of each behavior in the context of global robot action. All multiplied behaviors sum up to generate the global output of the robot.

The schema output summation function was used in all the selected approaches in this paper. Except that, some of the task allocation methodologies required some changes in schema features. Behavioral assemblages were developed, which are groups of motor-schemas to produce behaviors that are more complex. These are explained in later sections.

6. SELECTED TASK ALLOCATION METHODOLOGIES

It was attempted to solve a complex, high-level, distributed and team-oriented task by proposing a few low-level local control systems. A few robot architectures were proposed for individual robots that could collectively perform complex tasks at multi-agent level. All the methodologies for task allocation have been adapted to the hostile and dynamic environment of a soccer match. Figure 1 shows the behaviors used in higher-level architecture behavior-based and in lower-level motor schema and subsumption architecture. Three methodologies, namely, Alliance, Territorial, and Reactive were chosen for soccer teams and are described in the following sections.

6.1 Alliance

Alliance is a fully distributed and fault-tolerant robotic architecture that allocates loosely coupled tasks to a team of heterogeneous robots (Parker,

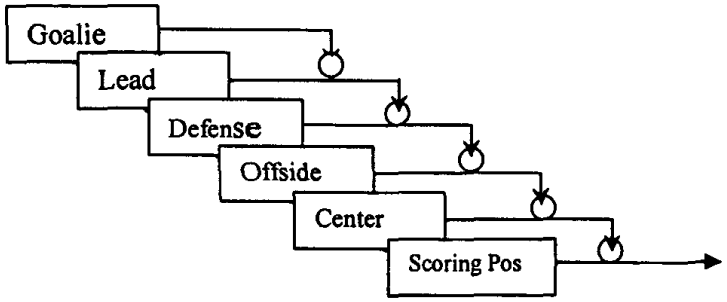

Fig. 1: Architecture and basic behaviors for methodologies

1993, 1994, 1996, 1997a, 1998, 1999). This approach is designed for small-to-medium size teams. In addition, Alliance is a behavior-based architecture that assigns tasks dynamically. Its behavior-based controller uses different groups of behaviors, called behavior-set, for different tasks. Each of them represents a functionality required to finish the task. Each robot in the team has to run an Alliance process parallel to its original controller to cooperate with other robots in the team. The robots communicate explicitly and globally. An extended approach called *L_Alliance* (Parker, 1995, 1997b) incorporates the applications of learning in Alliance.

The selection of a suitable task is based on a concept called motivation. Motivations in Alliance are mathematically modeled using two functions of impatience and acquiescence. Each robot has a partial knowledge of its own and other robots' state. This partial knowledge plus impatience and acquiesce is used to calculate the level of activation as a probability value computed for suitability of actions. Impatience happens when a robot perceives that another robot (considering its effect on the environment) has not achieved enough. Acquiesce happens when a robot understands its incapability to complete a task using its sensory feedback.

Alliance was selected as one of the task-allocation methodologies because it enables a team of robots to react systematically to the changes of environment and to the modifications of the team. However, there are a few differences between Alliance in the contexts that the developers defined it and its use in soccer. It was necessary to adapt and incorporate additional features into the methodology. One difference between original Alliance algorithm and this implementation is the availability of free robots. The developers

assume that robot R1 will take over an uncompleted task T1 when robot R2 acquiesces it and R1 has already completed its own task. Therefore, it was assumed that R1 is idle and waiting for the result of R2. But in robot soccer, robot R1 is never idle and therefore is never available to take over a task. In this implementation, a customized definition for motivation was used to adjust to this dynamic and competitive environment. Additionally, soccer simulation systems provide a sensible environment for vision sensors by using a global vision application. There are advantages to using this feature, which out-performs the incomplete and local sensory data used by developers.

6.1.1 *Methodology and representation.* The concept of motivation is based on distinguishing other agents' behaviors and responsibilities, and therefore a blackboard algorithm was implemented to provide sufficient data to team members via communication. Communication mechanism was made simple where messages had a simple format. Message contents included four pieces of information, namely, which robot is sending the message, what task it does, the current time, and when it starts the task. The first thing each robot checks is its perceptual inputs. If the perceptual schemas confirm some actions, then other criteria will be checked in order to select a task. When a robot activates one of its behavior sets, it must inhibit all other tasks by changing their motivational impatience value to zero.

When robot R1 is doing task T1 for some time less than the threshold time, then robot R2 must calculate its impatience parameter for doing task T1 using a slow rate. Robot R2 is aware of this fact because it received a communication message indicating such. Other wise robot R2 must use a fast rate. In this modified version, a very aggressive version was used and fast rate passes the motivation threshold in two simulation time slices. Slow rate passes the threshold in less than eight time slices. If robot Ri receives the first message from another robot indicating the starting of a task, then Ri must reset the associated impatience value. A robot gives up its task when it has worked on it for some time (more than required limit) and another robot is willing to do the job, or if the robot has worked on a task for a very long time.

To react robustly and coherently, teams of robots must provide two general strategies to win the game: defense and offense. In this implementation of Alliance, motivations were mathematically modeled to

express acquiescence and impatience based on these two situations: scoring and preventing the opponent from scoring. These situations have the top two priorities, while other activities support these main goals.

The defense strategy is illustrated in Figure 2. The left (light) team defends its goal against the opponent, which is to the right (dark). The illustrated strategy is called man-2-man defense. Each agent blocks an opponent. The ball is in the opponent's control.

In modeling the acquiescence, in a case where robot R1 has tried unsuccessfully for some time to stop an opponent attacker that controls the ball, R1 sends a signal claiming that it acquiescence the task of stopping the attacker. Yet, as this task has the top priority in the defense situation, it continues to do the task at least to slow the attack. Robot R2, the closest teammate, must also acquiescence its task to join R1 to stop the danger. The term closest can be defined in different forms depending on its concept. In defense closest means a point between the goal and the ball. Each robot based on the priority of its task must acquiescence its task and start the higher priority task. This chain continues until the least significant task will be left undone.

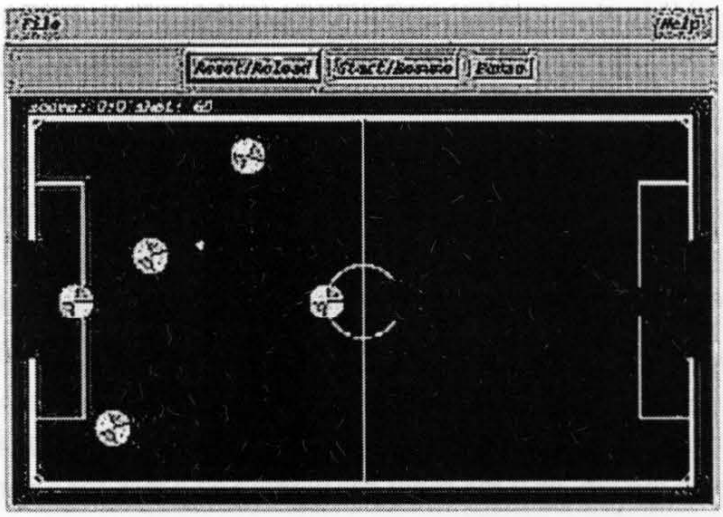

Fig. 2: The left (light) team is in a defensive position, since the soccer ball is in their field and is closer to an opponent player

In modeling of the impatience, when robot R1 has tried unsuccessfully for some time to stop an opponent attacker that controls the ball, robot R2, which has the best situation to defend the goal, becomes impatience. Robot R2 acquiescence its task and tries to stop the attacker, while R1 will continue to stop the attacker.

A typical offensive strategy is illustrated in the Figure 3, where the left (light) team is on the offensive against the opponent, which is the dark team. The ball is in light team's control. If another teammate has a better chance to score and it is possible to pass, then do the robot would not drive the ball (acquiescence) and instead steers toward the receiver and kicks the ball, i.e., passes the soccer ball. If a robot cannot dribble and drive the ball forward unless it places the control of the ball in danger, then the robot becomes acquiescence and safely passes it back to another teammate. If the robot is not driving the ball but the ball is in the team's control and the robot is not doing an important part of the offensive plan like blocking the opponent's goalie, then the robot gets close to a good receiving point to be ready to receive a ball from the agent who controls the ball. In offensive position, a receiving position is called close if it is close enough to receive a clear and safe pass and drive ahead.

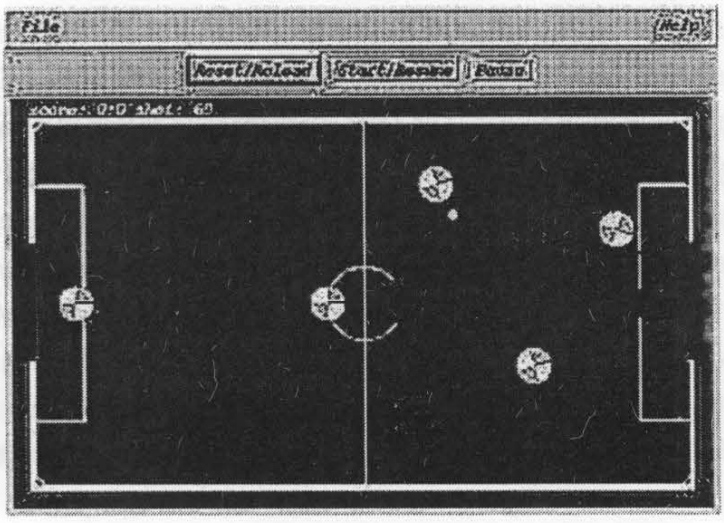

Fig. 3: The left (light) team is in an offensive position, since the ball is in the opponent's field and is closer to a teammate

6.2 Territorial Task Division

Schneider-Fontan and Mataric (1998) present a task allocation methodology to reduce robot interference by separating the robots' workspace. This technique introduces a territorial task allocation that enables the control system to assign an individual territory to each robot. The separated territories can be resized by adding and removing robots dynamically, which provides fault-tolerance capability for this methodology. This method can be applied to task allocations in soccer games. Indeed, in a real soccer game, each player has a very specific and spatially restricted play environment. This space-related characteristic separates goalie, defender (back), mid-field (halfback), and forward from each other because each of these tasks has a pre-assigned spatial definition. For instance, the goalie is closer to the team's own goal and the forward is closer to the opponent's goal. This mapping between tasks and territories allows for a more systematic implementation. This methodology is founded based on the concept that interference is a key factor that prevents mobile robots from gaining their maximum performance when operating as a team. Abating interference and maximizing synergy are the foci of this selected approachto allocating tasks.

6.2.1 *Methodology and representation.* A homogeneous team of robots was used to implement the concept of territorial task division, in which each robot was assigned equal space. The soccer field was divided into five territories, namely, goal, back, halfback, mid-forward, and forward. These territories are shown in Figure 4. The total workspace is rectangular area that is bounded by (xmin, ymin) and (xmax, ymax). Each robot has an individual a priori workspace. This workspace defines the allocated task for the player.
Each robot tries to go behind the soccer ball in its own Working Area to get the control of the ball. When it gets the control of the ball, the robot drives the ball to the Attacking Area, passes it to Positioning Area, and then returns to its Working Area. The Working Area is the area that a robot tries to gain the control of the ball and move it to the Attacking Area. The Attacking Area is one working area closer to opponent's goal and provides a better scoring chance for the team compared with the Working Area, unless the Working Area is immediately in an opponent's goal area. The scoring position is the

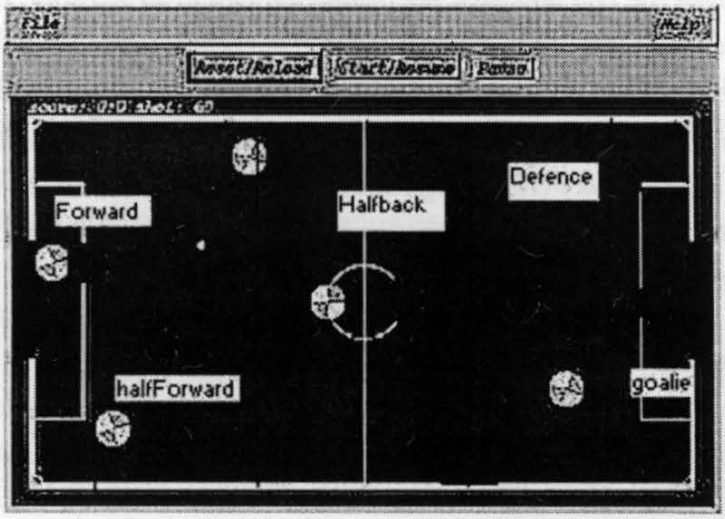

Fig. 4: A Possible territorial division and the tasks allocated to each related territory

place that a teammate can receive a pass in its own Attacking Area from one; or a place that allows one to score.

All the workspaces were assigned the same length, which was | Ymax – Ymin | but the width of each workspace depended on the number of robots. This dependency on the number of robots allows the control system to apply a dynamic workspace to each robot to handle faulty situations. The workspace width was calculated as:

$$\Delta x = \frac{|X_{\max} - X_{\min}|}{\#robots}$$

The geography of each behavior was changed, and each behavior was closely related to the robot's position. Therefore, the strategy that was used for Alliance was not suitable for Territorial approach. Additionally, in Alliance a heterogeneous team was implemented while the Territorial method required a homogenous team. The algorithm consisted of a sequence of commands: While the robot does not have the ball, it gets behind the ball. If the robot has the ball, if it can score, it scores otherwise drives it to next Working Area. If the robot is in the next Working Area and has the ball, it plays safe (passes it safely to the teammate) or leaves it in the Attacking Area position. If a robot can score, it scores. The robot returns to its own Working Area. This approach uses only one behavior. Each robot reacts to its environment by using this behavior. First, it uses a perceptual schema to find its territory. After that, it

tries to follow the ball and even get behind the ball in its territory. If the ball is not in its territory, the robot just tries to be in the best location inside its territory that can receive the ball. If the ball is inside the robot's territory, then the robot gets behind it. If the robot manages to get the control of the ball, it drives the ball to the next territory (Attacking Area) and kicks the ball in the best direction, meaning that if the ball is inside the forward area, the robot shoots it toward the goal, otherwise the robot passes the ball to the best location.

6.3 Static and Reactive Task Allocation

Static and Reactive Task Allocation is a simple task allocation methodology that uses a static and reactive compiled algorithm for a schema-based robotic soccer team. In this approach, the required tasks are mapped to robots during the design time. A team of five robots was tested in a series of soccer matches, for which where the following four tasks were used: (1) Goalie, (2) Defense, (3) Lead or attacker, and (4) Blocker. A robot was assigned to each of these tasks except the defense task, which had two robots. All robotic concepts that were discussed in previous sections were considered and implemented, including schemas and behavior-based robotics. The descriptions of the tasks are also the same as those defined in Alliance.

7. EXPERIMENTAL RESULTS

7.1 Applied Research Criteria

To evaluate task allocation methodologies described in the previous section, we used the following parameters:

- *Score.* It is usually argued that most of the time the better team is the winner. Additionally, score provide the capability to compare defensive versus offensive strategies of a team. A team that scores more has a better offensive capability; and a team that receives fewer goals has a better defensive capability.
- *Communication* is usually expensive, as it includes the hardware required to provide it and the software for processing it. The total number of messages passed during a soccer match was used as a utility parameter because this number could be calculated during a soccer game.

- *Fault-Tolerance* involved studying the effects of removing agents from a cooperative team of soccer players.
- *Number of Behavior Changes*. The average number of behavior changes for each agent was computed, as these changes could be related to cost.
- *Play against each other*. The results of teams playing against each other provide a viable parameter for comparisons.

7.2 Scores

The selected methodologies were tested in a tournament with a league of 16 soccer teams that played in a round robin fashion against each other. The calculations included the scores, number of wins, number of tie games, number of lost games, scored goals, and received goals.

First, the teams were tested in the absence of failure by assuming that all robots perform their tasks completely without failure. For preventing and eliminating random (luck-based) wins, each team was allowed to play eight times with other teams. This resulted in each team playing 128 games in the tournament. In this section, the results of the experiments are depicted, comparing different methods based on goals and points. Similar to regular soccer leagues each win results in three points, each tie has one and each loss has zero points. The results are shown in Figure 5 through Figure 10 for each methodology in the tournament.

7.3 Fault-Tolerant

The fault-tolerance and robustness of selected methodologies were evaluated by studying the effect of removing agents from a cooperative team of players. The results, as score changes, are shown. Although the intent was to remove an agent from the team randomly, it was not possible because the simulator always removed the robot whose ID was 0. This reduces the amount of randomness in static and reactive task allocation methodologies. The Alliance methodology proved that its dynamic task allocation mechanism and subsumption architecture used for defining task priorities are very efficient especially in dynamic environments such as soccer. The results of fault-tolerance experiments are shown in Figure 11 through Figure 16.

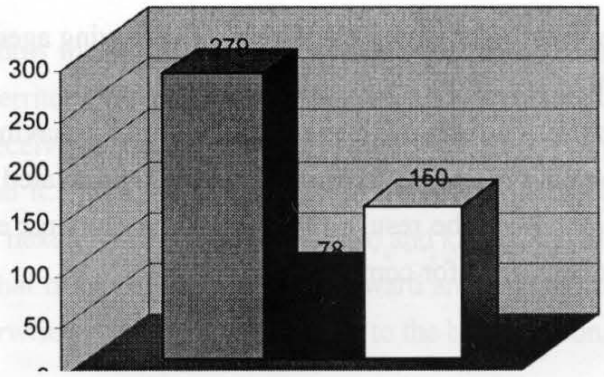

Fig. 5: Total points in the tournament

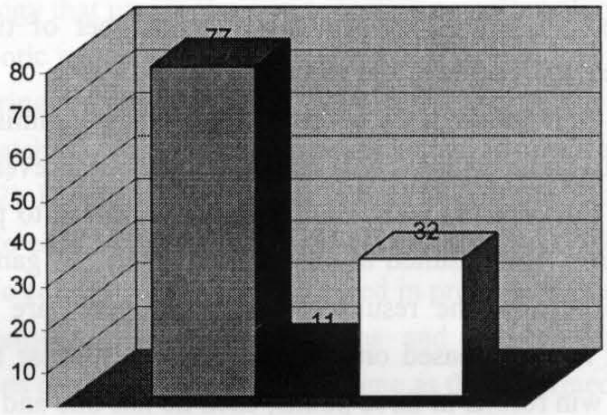

Fig. 6: Number of wins

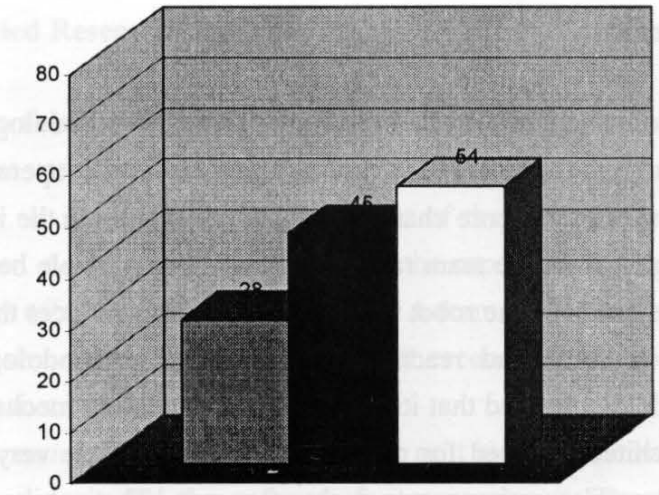

Fig. 7: Number of tie games

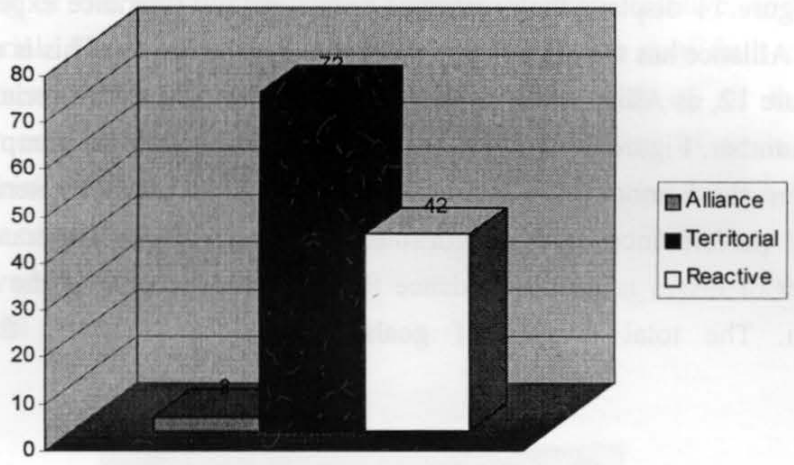

Fig. 8: Number of losses

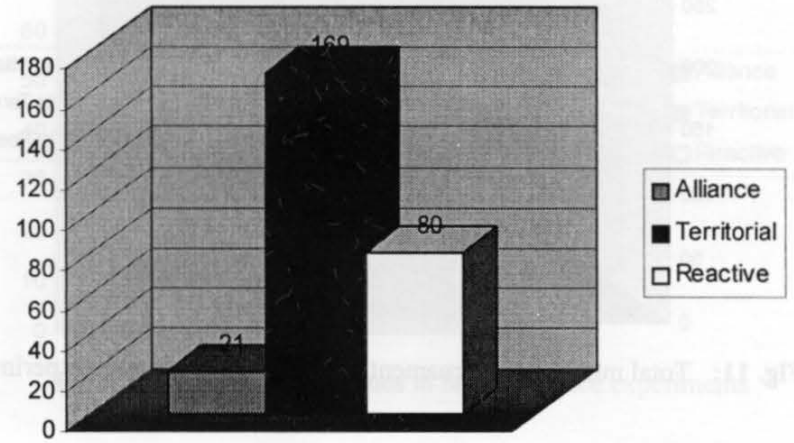

Fig. 9: Defensive capability measured as the total number of goals received

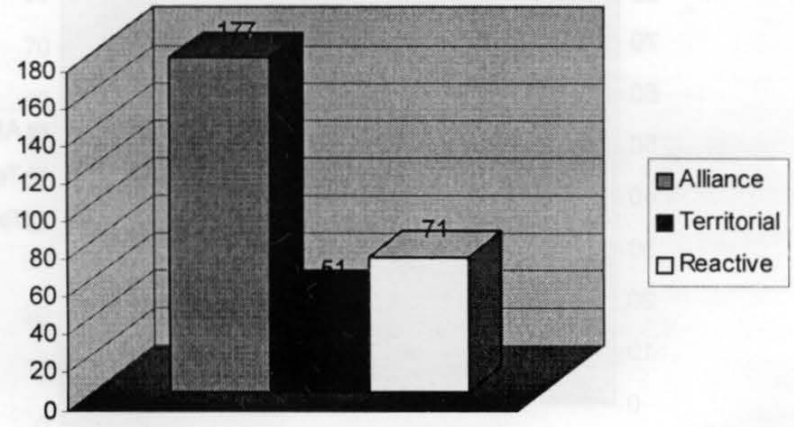

Fig. 10: Offensive capability measured as the total number of goals scored.

Figure 11 displays the number of points in fault-tolerance experiments, where Alliance has the highest and Territorial has the lowest. This is reflected in Figure 12, as Alliance has the most number of wins and Territorial has the least number. Figure 13 displays the number of tie games for completeness; however, the number of tie games by itself cannot be used to determine the overall performance of the algorithms in regards to fault-tolerance. The number of losses is further evidence for the fault-tolerance of the Alliance system. The total number of goals received also reflects the same

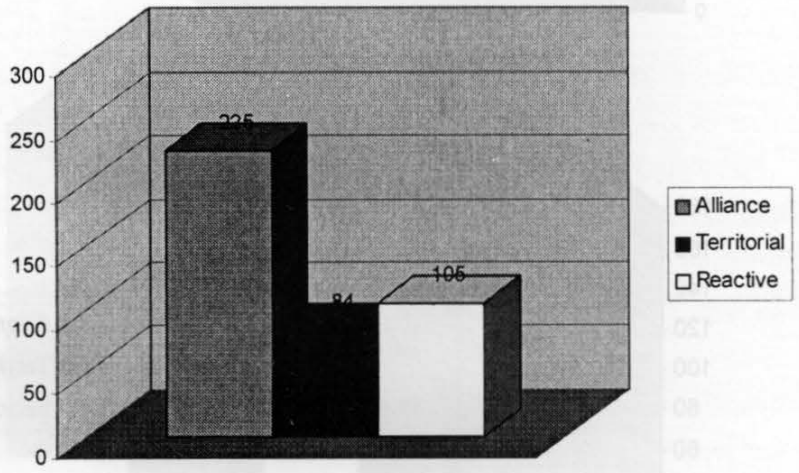

Fig. 11: Total number of tournament points in fault-tolerance experiments

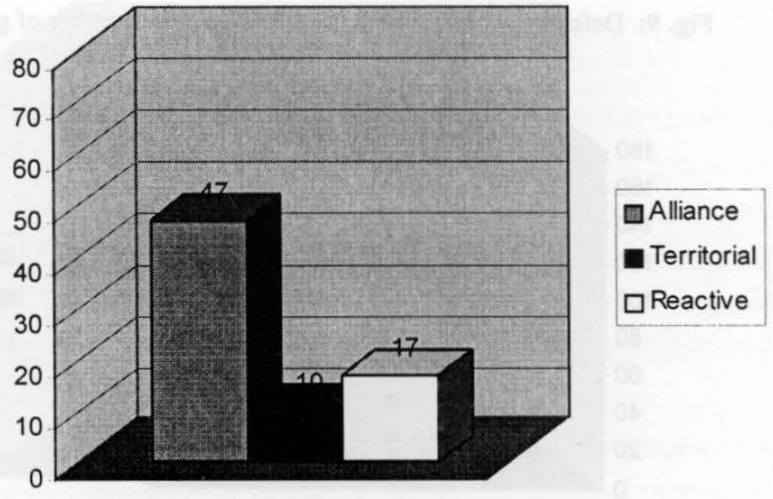

Fig. 12: Number of wins in fault-tolerance experiments

ordering of the techniques, with Alliance first, then Reactive, and finally Territorial. The total number of goals scored is the only fault-tolerance experiment where the ordering does not follow that of all the other experiments. Although Alliance has the highest number of goals scored, Territorial does not have the lowest, and instead Reactive scored the least number of goals. This may be a result of defensive versus offensive strategies employed by the techniques and how the removal of one robot agent can affect the goals scored and received.

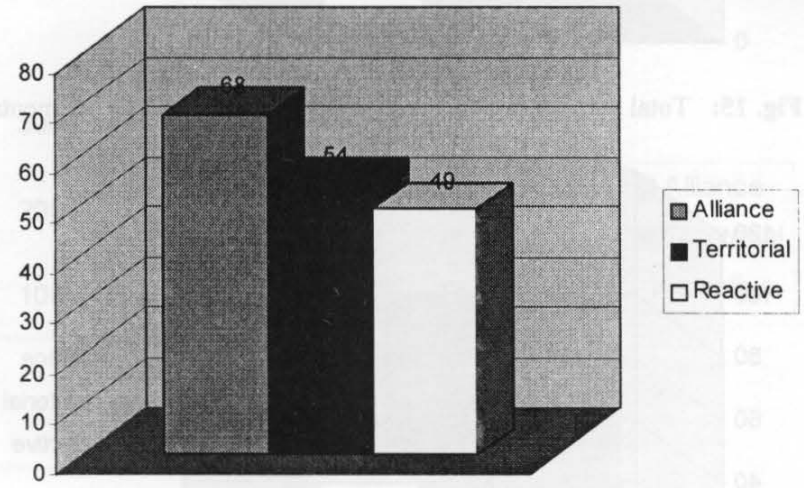

Fig. 13: Number of tie games in fault-tolerance experiments

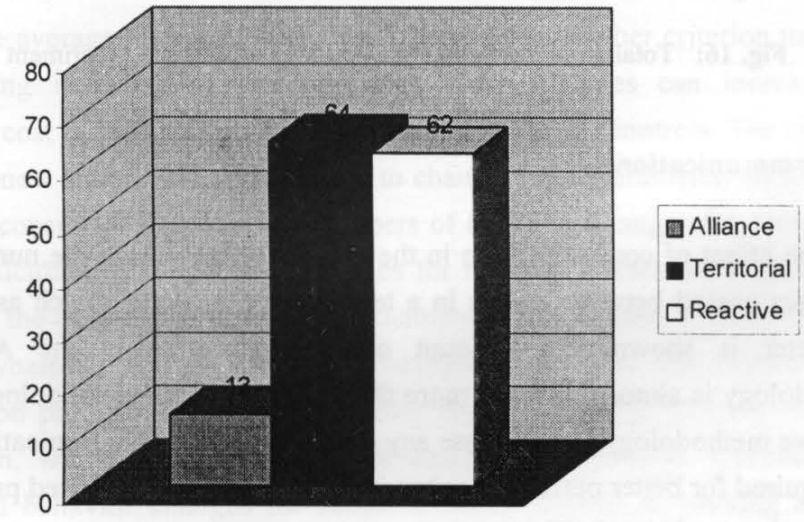

Fig. 14: Number of losses in fault-tolerance experiments

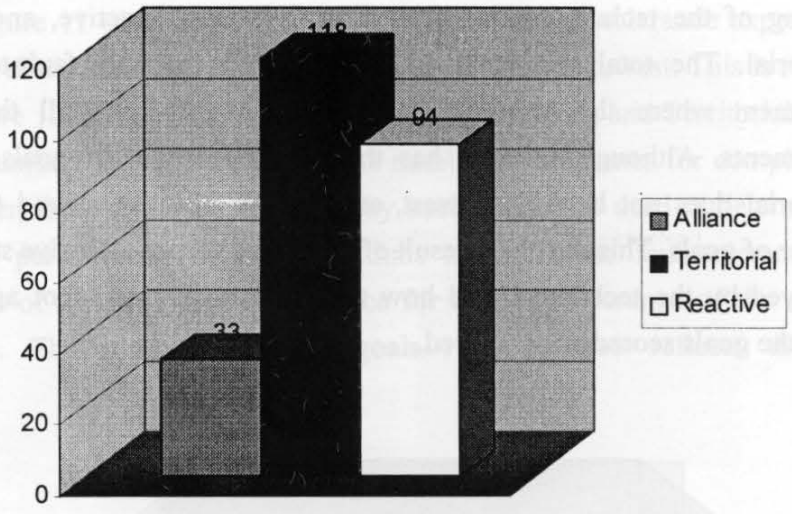

Fig. 15: Total number of goals received in fault-tolerance experiments

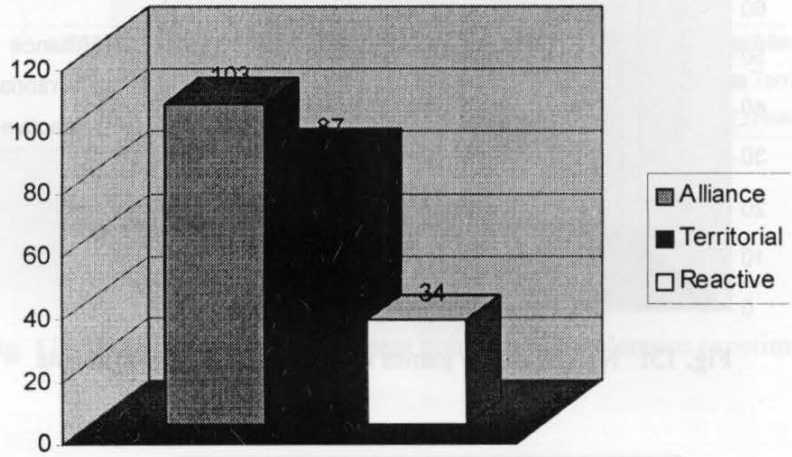

Fig. 16: Total number of goals scored in fault-tolerance experiment

7.4 Communication

The effect of communication in the context of measuring the number of messages passed between agents in a team during a soccer match as a cost parameter is shown. The amount of communication in the Alliance methodology is almost 10 times more than the Territorial methodology. The Reactive methodology does not use any communication. Communication was not required for better performance but it could be beneficial if used properly. The results are shown in Figure 17.

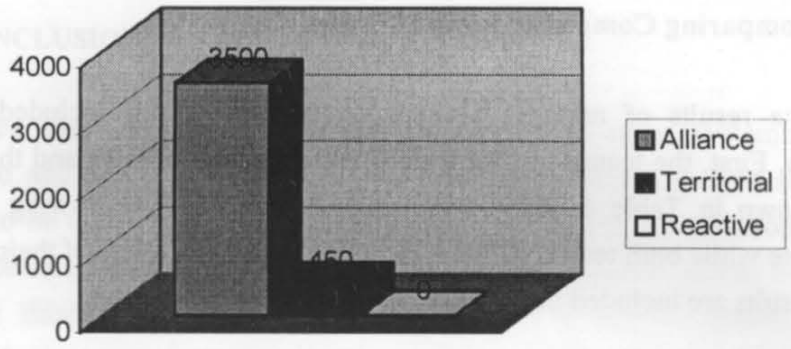

Fig. 17: Average number of messages in a soccer game

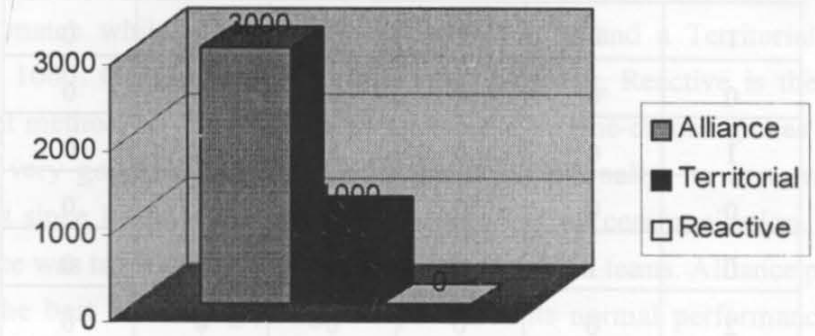

Fig. 18: Average number of behavior changes

7.5 Number of Behavior changes

The average number of behavior changes was another criterion used for evaluating implemented methodologies. Such changes can increase the general cost as they contribute to the complexity of the controls. The changes require new planning steps and need to change motor parameter, which is an energy consumer. The average numbers of behavior changes for each robot were calculated and then the averages for the whole team were calculated. Finally, the average for the entire tournament was computed. As expected, the most dynamic approach changed behaviors most often. The dynamic task allocation property of Alliance provided the advantage of flexible behavior selection, which also makes it more expensive. Territorial approach also required behavior changes for robots when they were in working area or attacking area. The results are shown in Figure 18.

7.6 Comparing Competing Methodologies

The results of matches between methodologies are included in this section. First, the teams competed in a fault-free environment and the results are shown in Table 1. Next, the teams were tested by allowing them to compete while both teams were faulty through the loss of one of their players. The results are included in Table 2.

Table 1: *Methodology capability testing*

Alliance – Territorial		Alliance – Reactive		Reactive -Territorial	
1	0	0	0	0	0
0	0	1	0	0	0
1	0	0	0	0	0
0	0	0	0	1	0
0	1	0	0	1	0
2	0	0	0	0	0
2	0	0	0	0	0
0	0	0	0	1	0

Table 2: *Fault-Tolerance adaptability testing*

Alliance – Territorial		Alliance – Reactive		Reactive – Territorial	
0	0	1	0	0	1
0	0	1	0	2	2
3	0	0	0	2	1
0	0	1	0	0	0
2	0	2	0	0	0
0	0	2	1	1	0
0	0	0	0	1	1
0	0	0	0	1	1

8. CONCLUSION

Three task allocation methodologies for multi-agent systems were evaluated and analyzed. Alliance using the adapted algorithm and strategy proved to be the winner algorithm in both normal and faulty situations. Not only it defeated both Reactive and Territorial in one-on-one games but also lost just three games of the whole 128 games it played against other methodologies. However, it was the most expensive one. The amount of communication required for an Alliance team was 800% more than a Territorial team and on average; it is 4500 messages more than a Reactive team. In addition, Alliance required 3000 times of behavior changes in each soccer match while a Reactive team needed none and a Territorial team needed 1000. Considering the results and efficiency, Reactive is the most efficient method. It just lost a game to Alliance in one-on-one games and it gained very good results in the tournament. Additionally, its cost was the smallest since it needed no behavior changes and no communication. Fault-tolerance was tested by removing an active agent from teams. Alliance proved to be the best team by gaining 225 points of its normal performance and winning 47 games. In one-on-one games, Territorial showed that its dynamic task allocation capability can close the gap between its performance and that of Reactive. Territorial even improved its performance in the presence of fault 7% while Reactive lost about 33% of its performance and Alliance lost 16% of its performance. We should point out that all the methodologies were application dependent and the results are only for the game of soccer within the described conditions, and other applications were not tested.

Although Teambots provided a close similarity to real environment, a simulation system cannot be as good as real environment. Situated and embodiment are two main characteristic of mobile robot research that cannot exist in simulation systems.

Because methodologies were application dependent, they had to be changed in order to adapt to their new environment. This adaptation was not the best modification and other versions may be more efficient. In addition, each methodology required different strategy based on their limitation and capabilities. Evaluation could be more precise if there was a canonical strategy that uniquely could be applied to all of them.

ACKNOWLEDGMENT

This work was sponsored in parts by a grant from the Kansas NASA EPSCoR Program (KNEP).

REFERENCES

Agah, A. and Tanie, K. 1997. Robots playing to win: evolutionary soccer strategies, *Proceedings of the IEEE International Conference on Robotic and Automation,* Albuquerque, New Mexico, USA, April 1997, 632-637.

Arkin, R.C. 1998. Cambridge, Massachusetts, USA, *Behavior-Based Robotics.* MIT Press.

Arkin, R.C. 1989. Towards the unification of navigational planning and reactive control, *Proceedings of the AAAI Spring Symposium on Robot Navigation,* 1-5.

Arkin, R.C. 1987. Motor schema-based mobile robot navigation, *Proceedings of the IEEE International Conference on Robotics and Automation,* 264-271.

Arkin, R.C. and Balch, T. 1998. Cooperative multi-agent robotic systems, *Artificial Intelligence and Mobile Robots: Case Studies of Successful Robot Systems,* edited by D. Kortenkamp, R.P. Bonasso, and R. Murphy, Cambridge, Massachusetts, USA, AAAI Press/The MIT Press, 277-296.

Arkin, R.C. 1989. Motor schema based mobile robot navigation, *International Journal of Robotics Research,* **8**, 92-112.

Balch, T. and Arkin, R.C. 1998. Behavior-based formation control for multi-robot teams, *IEEE Transactions on Robotics and Automation,* **14**, 926-939.

Balch, T. and Arkin R.C. 1994. Communication in reactive multi-agent robotic systems, *Autonomous Robots,* **1**, 1-25.

Balch, T. and Parker, L. 2000. Guest editorial. *Autonomous Robots,* Special Issue on Heterogeneous Multi-Robot Systems, **8**, 239-267.

Balch, T. and Ram, A. 1998. Integrating Robotic Technologies with JavaBots, *AAAI 1998 Spring Symposium,* 1-3.

Bertsekas, D.P. 1992. Auction algorithms for network flow problems: a tutorial introduction, *Computational Optimization and Applications,* **1**, 7-66.

Brooks, R.A. and Connell, J.H. 1986. Asynchronous distributed control system for a mobile robot, *Proceedings of SPIE's Cambridge Symposium on Optical and Optoelectronic Engineering,* Cambridge, Massachusetts, USA, 77-84.

Brooks, R.A. 1991. Intelligence without representation, *Artificial Intelligence,* **47**, 139-160.

Brooks, R.A. 1986. A robust layered control system for a mobile robot, *IEEE Journal of Robotic and Automation,* **2**, 14-23.

Brooks, R.A. 1990. Elephants don't play chess, *Robotics and Autonomous Systems*, **6**, 3-15.

Cao, Y.U., Fukunaga, A.S., Kahng, A.B., and Meng, F. 1995. Cooperative mobile robotics: antecedents and directions, *Proceedings of the IEEE/RSJ International Conference on Intelligent Robotics and Systems*, Pittsburgh, Pennsylvania, USA, 1, 226-234.

Connell, J.H. 1990. *Minimalist mobile robotics: A colony-style architecture for an artificial creature*, San Diego, California, USA, Academic Press.

Connell, J.H. 1992. SSS: a hybrid architecture applied to robot navigation, *Proceedings of the IEEE International Conference of Robotics and Automation,* Nice, France, 2719-2724.

Corkill, D. 1991. Blackboard systems, *AI Expert*, **6**, 40-47.

Dias, M.B. and Stentz, A. 2000. A free market architecture for distributed control of a multi-robot system, *Proceedings of the Sixth International Conference on Intelligent Autonomous Systems*, Venice, Italy, 115-122.

Dorigo, M., and Gambardella, L.M. 1996. Ant colonies for the traveling salesman problem, *Universit'e Libre de Bruxelles,* Belgium, IRIDIA, Technical Report TR/IRIDIA/1996-3.

Dorigo, M., Maniezzo,V., and Colorni, A. 1996. The Ant System: Optimization by a Colony of Cooperating Agents, *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, **26**, 29-41.

Dudek, G., Jenkin, M., Milios, E., and Wilkes, D. 1993. A taxonomy for swarm robots, *Proceedings of the IEEE/RSJ International conference on Intelligent Robotics and Systems*, Yokohama, Japan, 441-447.

FIRA. 2007. http://www.fira.net.

Firby, R.J. 1987. An investigation into reactive planning in complex domains, *Proceedings of the Sixth National Conference on Artificial Intelligence,* Seattle, Washington, USA, 202-206.

Fredslund, J., and Mataric, M.J. 2001. Robot formations using only local sensing and control, *Proceedings of IEEE International Symposium on Computational Intelligence in Robotics and Automation*, Alberta, Canada, 308-313.

Gerkey, B.P., and Mataric, M.J. 2000. Murdoch: publish/subscribe task allocation for heterogeneous agents, *Proceedings of the Fourth International Conference on Autonomous Agents*, Barcelona, Spain, 203-204.

Gerkey, B.P., and Mataric, M.J. 2002a. Pusher-watcher: an approach to fault-tolerant tightly-coupled robot coordination, *Proceedings of the IEEE International Conference on Robotics and Automation*, Washington, DC, USA, 464-469.

Gerkey, B.P., and Mataric, M.J. 2002b. Sold!: auction methods for multi-robot coordination, *I EEE Transactions on Robotics and Automation, Special Issue on Advances in Multi-Robot Systems*, **18**, 758-786.

Goldberg, D., and Mataric, M.J. 1997. Interference as a tool for designing and evaluation of robust behavior-based controllers. *Proceedings of the AAAI-*

97, Providence, Rhode Island, USA, 637-642.

Goldberg, D., and Mataric, M.J. 2000. Robust behavior-based control for distributed multi-robot collection tasks, *USC Institute for Robotics and Intelligent Systems*, Technical Report IRIS-00-387.

Goldberg, D., and Mataric, M.J. 2002. Design and evaluation of robust behavior-based controllers for distributed multi-robot collection tasks, In *Robot teams: from diversity to polymorphism*, edited by Balch, T., and Parker, L.E., Natick, Massachusetts, USA, A K Peters Ltd, 315-344.

Lerman, K., 2000. Design and mathematical analysis of agent-based systems. *Lecture Notes in Artificial Intelligence*, Berlin, Germany, Springer-Verlag.

Lerman, K., and Gastyan, A., 2001. A general methodology for mathematical analysis of multi-agent systems, *USC Information Sciences,* Technical Report ISI-TR-529.

Lesser, V.R. 1991. A retrospective view of FA/C distributed problem solving. *IEEE Transactions on Systems, Man, and Cybernetics*, **21**, 1347-1362.

Lesser, V.R. 1995. Multi-agent systems: an emerging subdiscipline of AI, *ACM Computing Surveys*, **27**, 340-342.

Mataric, M.J. 1997. Behavior-based control: examples from navigation, learning and group behavior, *Journal of Experimental and Theoretical Artificial Intelligence*, **9**, 323-336.

Moravec, H.P., and Cho, D.W. 1989. A Bayesian method for certainty grid, *Working Notes of ARAI 1989 Spring Symposium on Robot Navigation,* 57-60.

Shehory, O., and Kraus, S. 1998. Methods for task allocation via agent coalition formation, *Artificial Intelligence*, **101**, 165-200.

Ostergaard, E.B., Mataric, M.J., and Sukhatme, G.S. 2001. Distributed multi-robot task allocation for emergency handling, *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, Maui, Hawaii, 1226-1231.

Pirjanian, P. and Christensen, H.I. 1998. Satisfying action selection, *Proceedings of the SPIE Conference on Intelligent Systems and Advanced Manufacturing*, Boston, Massachusetts, USA, 153-164.

Parker, L.E. 1993. Designing control laws for cooperative agent teams, *Proceedings of the IEEE International Conference on Robotics and Automation*, 582-587:

Parker, L.E. 1994. An experiment in mobile robotic cooperation, *Proceedings of the ASCE Specialty Conference on Robotics for Challenging Environments*, 131-139.

Parker, L.E. 1995. *L-ALLIANCE: a mechanism for adaptive action selection in heterogeneous multi-robot teams*, Oak Ridge National Laboratory ORNL/TM-13000 Technical Report, Oak Ridge, Tennessee, USA.

Parker, L.E. 1996. On the design of behavior-based multi-robot teams, *Journal of Advanced Robotics*, **10**, 547-578.

Parker, L.E. 1997a. Cooperative motion control for multi-target observation,

Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, 1591-1598.

Parker, L.E. 1997b. L-ALLIANCE: task-oriented multi-robot learning in behavior-based systems, *Advanced Robotics, Special Issue on Selected Papers from IROS'96*, **11**, 305-322.

Parker, L.E. 1998. ALLIANCE: an architecture for fault-tolerant multi-robot cooperation, *IEEE Transactions on Robotics and Automation*, **14**, 220-240.

Parker, L.E. 1999. Cooperative robotics for multi-target observation, *Intelligent Automation and Soft Computing*, **5**, 5-19.

Parker, L.E. 2001. Evaluating success in autonomous multi-robot teams: experiences from ALLIANCE architecture implementations, *Journal of Theoretical and Experimental Artificial Intelligence*, **13**, 95-98.

Pirjanian, P. 1998. Multiple objective action selection and behavior fusion using voting, Ph.D. Dissertation, *Department of Medical Informatics and Image Analysis,* Aalborg University, Denmark.

Pirjanian, P. 1998. Multiple objective action selection in behavior-based control, *Proceedings of the Sixth Symposium for Intelligent Robotic Systems*, Edinburgh, Scotland, 83-92.

RoboCup. 2007. http://www.robocup.org.

Russell, S., and Norvig, P. 1995. *Artificial intelligence: a modern approach,* Upper Saddle River, New Jersey, USA, Prentice Hall.

Schneider-Fontan, M.J., and Mataric, M. 1998. Territorial multi-robot task division, *IEEE Transaction on Robotics and Automation,* **14**, 815-822.

Sgorbissa, A., and Arkin R.C. 2003. Local navigation strategies for a team of robots, *Robotica*, **21**, 461-473.

Stentz, A. and Dias, M.B. 1999. *A free market architecture for coordinating multiple robots,* Carnegie Mellon Robotics Institute Tech Report CMU-RI-TR-99-42, Pittsburgh, Pennsylvania, USA.

Stone, P., and Veloso, M. 1999. Task decomposition, dynamic role assignment, and low-bandwidth communication for real-time strategic teamwork, *Artificial Intelligence*, **110**, 241-273.

Sukthankar, G. 2000. Team-aware multirobot strategy for cooperative path clearing. *Proceedings of the Seventeenth National Conference on Artificial Intelligence*, 1095.

Sukthanker, G., and Sycara, K. 2001. Team-aware robotic deminig agents for military simulation, http://www-2.cs.cmu.edu/~softagents/iaai00/iaai00.html.

Swarm. 2007. http://www.swarm.org.

TCA. 2007. http://www.cs.cmu.edu/~TCA/tca.orig.html.

TeamBots. 2007. http://www.teambots.org.

Werger B.B., and Mataric M.J. 1999. *Exploiting embodiment in multi-robot teams*, IRIS Technical Report IRIS-99-378, Los Angeles, California, USA, University of Southern California.

Werger, B.B., and Mataric, M.J. 2000. Broadcast of local eligibility: behavior-based control for strongly cooperative robot teams, *Proceedings of the*

Fourth International Conference on Autonomous Agents, 21-22.

Werger, B.B., and Mataric, M.J. 2000. Broadcast of local eligibility for multi-target observation, In *Distributed Autonomous Robotic Systems 4*, edited by Parker, L.E., Bekey, G.A., and Barhen, J., Springer, 347-356.

Werger, B.B. and Mataric, M.J., 2001. From insect to Internet: situated control for networked robot teams, *Annals of Mathematics and Artificial Intelligence*, **31**, 173-198.

Werner, G.M. and Dyer, M.G. 1991. Evolution of communication in artificial organisms, *Proceedings of the Second Interdisciplinary Workshop on Synthesis and Simulation of Living Systems*, 659-687.

Yanco H., and Stein, L. 1993. An adaptive communication protocol for cooperating mobile robots, In *From Animals to Animats: International Conference on Simulation of Adaptive Behavior*, edited by Meyer, J.-A., Roitblat, H., and Wilson, S., Cambridge, Massachusetts, USA, MIT Press, 478-485.

Yanco H. 1993. Talking about the world: cooperative robots that learn to communicate, In *Learning Action Models: Papers from the 1993 AAAI Workshop*, AAAI Technical Report WS-93-06, Washington, DC, USA, 52-56.