Adaptive Behavioral Control of Collaborative Robots in Hazardous Environments

Kevin McCarty, Member IEEE, Milos Manic, Sr. Member IEEE[1]

Abstract— **Terrain exploration carries with it significant hazards. Robots attempting to map a piece of unknown terrain must be able to make decisions and react appropriately to dynamic and potentially hostile conditions. However, because of constraints on size and cost, robots may have limited ability to store and process necessary information. In addition, knowledge discovered by others may be difficult to share. This paper proposes a system using a powerful master controller, operating from a safe environment, directing the movements of numerous robots exploring a piece of terrain. The master controller processes the information from the robots, updates the decision process and distributes these updates back to the robots. This process allows for a cooperative, effective search environment while also maintaining a small processing footprint. It also allows the robot to employ adaptive, subsumptive behavioral modification as new information is made available. A test simulation of a hazardous environment demonstrates that even robots with little intrinsic intelligence can learn complex behaviors in order to reach their goal.**

I. Introduction

For brevity's sake, this paper will consider robots, autonomous vehicles and software agents to have similar characteristics and be classified using the term "robot".

The process of robot learning proceeds as follows:

1. A robot ventures into an unknown space.
2. Within that space, the robot uses sensors to acquire information about the space.
3. The robot takes that information and applies algorithms to determine the best course of action.
4. Using actuators, the robot implements the action, then, using sensors, evaluates the result.
5. The result is then assigned some utility value which the robot uses to determine whether the action taken was "good" or "bad". "Good" results reinforce a given action or behavior, while "bad" results do just the opposite [1], [2].

A typical autonomous robot must manage a large array of sensory information to determine its environment. Each sensor provides some input about the world around the robot; that input being incorporated into a knowledge base. From this knowledge base, appropriate rules about actions taken in response to input are generated. These rules allow the robot, to interact with its surroundings in a way that hopefully achieves some goal.

However, creating and maintaining these rules, as well as gathering new data for the knowledge base poses significant challenges [3], [4]. Processes such as advanced data mining techniques for rule generation require significant computing resources in order to store, search and manage large datasets, which can be very computationally expensive. Putting such resources at risk in a hostile environment can prove difficult and costly. In addition, forcing each robot to carry such resources internally means duplicating, perhaps unnecessarily, expensive assets. However, the robot needs to be able to process and act upon any new information from its environment. Otherwise, the robot will have to rely on a static set of rules which may be inadequate if the robot's circumstances change significantly. Algorithms for climbing up stairs may do little good if the step or two is missing. Routines for dealing with a threat such as a water obstacle have little use when the water turns to ice. Without being able to process and respond to new information, a robot loses its ability to adapt.

There is also the issue that real-time computations using adaptive behavioral techniques such as neural networks also impose significant amounts of memory and processing power. Such constraints may serve to greatly limit either the operational effectiveness of a robot.

Finally there is the matter of robot size. Putting significant computing resources on a robot requires both power and a certain minimal size constraint, which may be too large to allow a robot to operate effectively.

There are other practical matters as well. Many robots have the potential to explore an area more quickly and more thoroughly than one [5], [6], [7]. In addition, losing an inexpensive robot to a hazard is much preferable to losing an expensive one; particularly if, in the case of an inexpensive robot, there are replacement robots nearby that can be directed to take its place.

This paper is organized as follows: Section II will present a problem statement. Section III describes the steps of this approach. Section IV will look at software used to run the simulation and its results. Section V will present conclusions and future work.

II. Problem Statement

Consider the example of a chemical spill. Toxic waste accumulates in a storm drain. Robots are sent in to assess the extent of the contamination by finding, obtaining, and returning with chemical samples.

While robots can be prepared for many things, it would be extremely difficult and very expensive to attempt to prepare them for everything. In a storm drain there might be rodents, insects, debris and hazards associated with the shape of the cavity or how it was constructed. Hence, the ability to adapt and make good decisions with new information is extremely important.

[1] The authors are with the Department of Computer Science, University of Idaho at Idaho Falls, Idaho Falls, ID 83402. Email: kmccarty@ieee.org, misko@ieee.org

A number of solutions have been proposed to address this problem, including the use of cooperative clusters of autonomous robots [6] which switch between modes ranging from highly cooperative to loosely cooperative, depending upon the situation. There are also human and robot cooperative teams which utilize key performance parameters combined with supervisory human intelligence to direct the behaviors of robot clusters [5].

There are also techniques to direct the actions of multiple robots working in concert [8], such as factory assembly line or multi-robot system, through the use of a central controller [9] or employing sophisticated genetic or other adaptive algorithms [10]. Behavioral subsumption architectures and evolutionary algorithms have also been used to implement adaptive, cooperative behavior among autonomous units [2], [11].

These techniques, however, have limitations in trying to implement cooperative, adaptive, autonomous behavior in dynamic environments for the following reasons:

1. In the case of cooperative clusters and robots working in concert, the actions of each robot are specialized. The loss of any one robot may mean serious degradation the capabilities of the overall system.
2. Adaptive architectures are localized; hence information from robots that might be beneficial to the overall function of the group may not be easily assimilated or shareable.
3. Updating key performance parameters, behavioral modifications and other autonomous functions depends upon human interpretation of input data along with human initiated interaction and supervision. Human intelligence is often unable to perform timely analysis and mistake-free interaction and hence is limited in its ability to make the necessary adjustments.
4. In all the above configurations, a robot's ability to incorporate and analyze large datasets and quickly propagate the results to the group may be limited by its intrinsic hardware.
5. Real-time solutions may be beyond a given robot's computational capacity.

Advanced data mining techniques [12] have been shown to be very successful in determining predictive behavioral models as well as finding hidden patterns and associations in data. Prior research [1], [7], demonstrates the effectiveness of multiple robots working together and sharing knowledge. By combing the intrinsic power of data mining with the cooperative capabilities of clusters of exploring robots a very powerful system for data gathering, analysis and real-time behavioral adaptations is possible.

This paper presents a practical implementation that, instead of relying on expensive, highly intelligent and capable robots, uses the cooperative efforts of a cluster of smaller, cheaper robots and a powerful central controller. Throughout the example, the robots possess no real intelligence. Instead, they sense their environment and collectively learn what to do about it through basic trial and error. A central controller takes the information accumulated, analyzes it, creates new models of behavior and distributes them to the robots. By letting the robots work cooperatively and exchange information via a central controller located safely away from the zone of exploration, there are a number of benefits:

1. Robots can have a much smaller footprint and be produced much more cheaply. This would allow exploration of areas deemed too hazardous or otherwise impractical for larger, more expensive robots.
2. Robots could be "sacrificed" more readily to discover unknown hazards and obstacles.
3. Information gained from "sacrificing" a robot could be transmitted to the controller and passed out in real-time to nearby robots along with instructions for dealing with the hazard.
4. Exploration could be done over larger areas with less power usage and time than otherwise.
5. Each robot benefits from the experiences of all the others and their movements can be coordinated as needed.
6. Such a "trial and error" approach can serve to uncover previously unknown hazards and behaviors to counter them. Human experimentation is also made easier as "mistakes" are less costly.

Consider a hazardous environment consisting of a small drainpipe. A chemical spill has occurred and some chemical has run down the drainpipe; but the extent of the contamination is not known. Massive cleanup operations can be very dangerous if the type and extent of the contamination is not known. It can also be very costly and should be undertaken only if absolutely necessary. The contaminated pipe contains unknown hazards such as debris and live rodents and its design is uneven so there are small ledges or trenches.

In order to determine whether the expense of a costly cleanup operation is necessary and where contaminants exists the robots must take samples inside the pipe and return them to a sampling station for analysis.

A software simulation of the environment pits an array of robots against a number of hazards, rodents, debris and ledges, along with chemical samples as shown in Fig. 1.

Fig. 1. Robots and hazards in a chemical spill

Any of these hazards can disable or "kill" the robot. The robot's goal is to find, and collect a sample and return it to a sampling station. In order for this to occur the robot must first explore the environment and then must keep alive long enough to reach its goal. It does this through intelligent application of built-in rules in response to its surroundings. Hazards must be identified and countered effectively; otherwise the robot becomes disabled and counted as a casualty.

Now consider a new type of collaborative effort using an array of robots, each with an array of behaviors, but no idea how best to employ them. The collective knowledge base is initially empty and rules are applied randomly. Somehow the robots must be able to learn what to do in the right situation in order to reach the gold. But many may “die”, or become disabled, in the attempt. However, by collectively pooling their experiences and utilizing a central controller (CC) with vast processing power and storage, the survivors will, over time, become more intelligent and better able to adapt and respond to the hazards they face. Even the robots that are unsuccessful and “die” add to the knowledge base through negative reinforcement. The knowledge is then used to develop ever more complex and appropriate behaviors.

III. Cooperative Algorithm

Pseudocode for the Hazardous Environment Simulation algorithm is as follows:

```
HazardousWorldExplore
  CentralController cc;
  while Chemical Sample Amount is below sufficient
  quantity and there are still Robots alive
     foreach Robot r in RobotArmy
         GatherSensorData
          AnalyzeData to determine situation
           Make a decision
           Do an action
           Calculate utility based upon results
             if Robot is still alive
               Update Robot utility
             Central controller adjusts behaviors rules
             Propagate new rules to all Robots
End HazardousWorldExplore
```

The algorithm is implemented in the following steps:

Step 1: Robot explores environment and gathers data.

The robot starts by exploring its environment. From this exploration it receives sensory data which is passed to the CC. For a given collection of robots R, each with N sensors, let r_i be a given sensor input and R_i be the collection of all the sensors for a given robot. An individual robot r creates an interpretation r_{int} by summing the collective input from its sensors:

$$r_{int} = \sum_{i=1}^{n} f_{int}(r_i) = f_{int}(R_i) \quad (1)$$

where r_{int} is the resulting interpretation. For example, in the hazardous world simulation, the robot, using a combination of light, camera and ultra-sonic sensors could determine the next space is a pit or an obstacle such as a boulder.

Step 2: Robot makes a decision.

The interpretation is fed into a function f_{dec} to determine the appropriate course of action. f_{dec} is the controller supplied function that dictates robot behavior.

$$r_{act} = f_{dec}(r_{intpr}) \quad (2)$$

Step 3: Robot acts on the environment and records the results.

Once a decision is made, the robot acts upon its environment using one of the many responses at its disposal. That action has consequences, which are recorded. Let r_c be a given consequence from a given action r_{act}.

Step 4: Data communication from robot to central controller.

Each robot then relays its data to the central controller (CC). The data include the environment, or situation the robot faced, r_{env}, the action or actions it took, r_{act}, and the consequences resulting from the action or actions, r_c. Some assumptions may also be implicit, such as if the robot stops transmitting, the CC will assume the robot has stopped functioning.

Step 5: Data gathering, transformation and analysis by the CC.

The central controller gathers, processes, and transforms the data for import into its knowledge base.

$$KB = \{(r_{env}, r_{act}, r_c)\ \forall r \in R\} \quad (3)$$

Because the CC is far more powerful than the robots themselves, it has the capability to do sophisticated analysis of all the data. Part of that analysis involves determining the appropriate utility for a given result. The CC takes a look at a given 3-tuple of environment/situation, action and result, assigning a utility u_i to measure the effectiveness of a given robot’s behavior.

$$u_i = f_{ut}(r_{env}, r_{act}, r_c) \quad (4)$$

As described in Section I, the learning process involves not just analyzing the data but also modifying the robot behavior in such a way as to encourage successful actions and discourage unsuccessful ones. In the Hazardous Environment Simulation successful actions generate positive utility while unsuccessful actions generate negative utility. The CC looks at the results of an action and must decide whether that action is appropriate based upon the results and adjust robot behavior accordingly. A common technique for doing this is called Q-Learning [13], [14]. Given a state space S and set of actions A, Q-Learning is designed to “teach” a robot the most appropriate behaviors through application of both positive and negative reinforcement.

Application of a learning function Q_i, over a state s_i, action a_i with a utility u_i generates a state table which can be incorporated with the existing state table st_i to generate new table st_{i+1}

$$st_{i+1} = st_i + Q_i(s_i, a_i) + u_i \quad (5)$$

By applying advanced data mining techniques (ADMT), such as a neural network classifier, k-means clustering algorithm, or decision tree generation algorithm such as C4.5, the new state table in the knowledge base produces an updated set of rules *B*.

$$ADMT(st_{i+1}) \Rightarrow \{B\} \quad (6)$$

Step 6: Behavior modification via new rule creation.

Once the analysis is complete, the central controller must adapt the behavior of the robots accordingly. The new *{B}* is then transformed by the CC into a new behavioral function, f_{dec}.

Step 7: New behavioral function transmission by CC to robots.

The robots receive the new behavioral function f_{dec}, replacing the existing one and proceed back to step #1.

IV. Hazardous Environment Simulation Test Example

The Hazardous Environment Simulation is a software simulation pitting software robots against rodents and other hazards. The goal for the robots is to find and collect chemical samples and return them to base. Along the way, they must confront and learn to overcome a variety of hazards both in order to get the samples as well as make it back to base.

The robots are equipped with a number of primitive behaviors:

1. Move forward from space to space.
2. Use the electric prod
3. Jump over a space.
4. Go around a space.
5. Collect a sample.
6. Return back to base to drop off samples or recharge the prod.
7. Avoidance behavior to stay clear of other robots or rodents

In order to make the environment more challenging there are also some complex, subsumptive behaviors required:

1. Robots prods only have 3 charges, so once they are out of charges, they won't be able to use the prod until they return to base and get recharged. Therefore, when number of charges reaches zero, the robot must employ a subsumptive decision matrix, override normal operations in order to return to base and recharge.
2. Robots can carry as many samples as they want, but they must return the samples to base in order to get credit. Again, subsumptive behavior applies as the robot must return the samples to the sample station.

The environment of the Hazardous World Simulation consists of the following:

1. The rodent. In order to "kill" the rodent, the robot must zap it with its prod or avoid it altogether. Otherwise, the robot gets eaten and dies.
2. The pit. Uneven construction means there are drop-offs to ensnare the robot. In order to defeat the pit the robot must jump over it or go around it or try to avoid it. If the robot moves forward or tries to "collect" it, the robot will fall down and die.
3. The boulder. In the environment, debris collects in mounds like a boulder. In order to defeat the boulder the robot must go around it or avoid it or move it aside using the prod. Attempting any other action means getting tangled up in the mess and being disabled.
4. The chemical sample. Any action other than a collect loses the sample.
5. The empty space. The robot must move forward onto the empty space.
6. Other robots. The robot must avoid them or it risks disabling them both.

Adding one further challenge for the robots is that for each situation there are multiple behaviors that can address it. It is important that the robot learn to maximize its utility by not only doing the right thing, but also doing the most appropriate thing. Jumping over a pit is better than going around it, though both are valid. Going around a boulder is better than avoiding it.

Finally the robot must be careful and survive. Chemical samples should be returned to base or they will be lost if the robot dies. Because the robot only receives a limited number of charges with which to zap rodents, it must learn to return for more or risk being without any charges when a rodent comes along.

Test Case 1- Individual situations

Each of the Hazardous Environment World situations was tested individually to see how quickly the robots would learn the appropriate behavior. The overall results are displayed in Fig. 2.

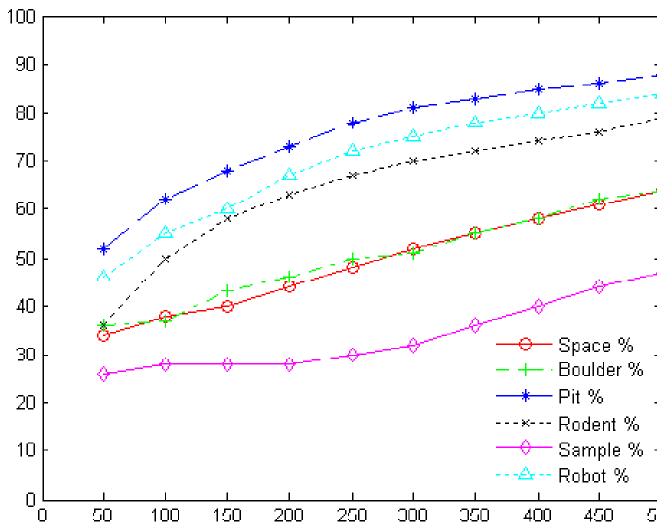

Fig. 2. Overall results of individual simulation situations

Measurements were taken of robot performance every 50 iterations. At first, the actions taken were essentially random, but as the utility was determined for each outcome and the knowledge base increased, the overall success rate for the robots increased significantly. Fig. 2 shows a gradual improvement in overall robot actions as they acquired new knowledge and learned to apply that new knowledge successfully. This is more clearly demonstrated by the graph in Fig. 3 which shows individual iteration success rates improve rapidly as new experience is gained. Prior experience provides new, more effective and accurate basis for subsequent decision making. In all instances, performance in the final iterations was significantly better than in the first, with success rates improving to near 100% in cases.

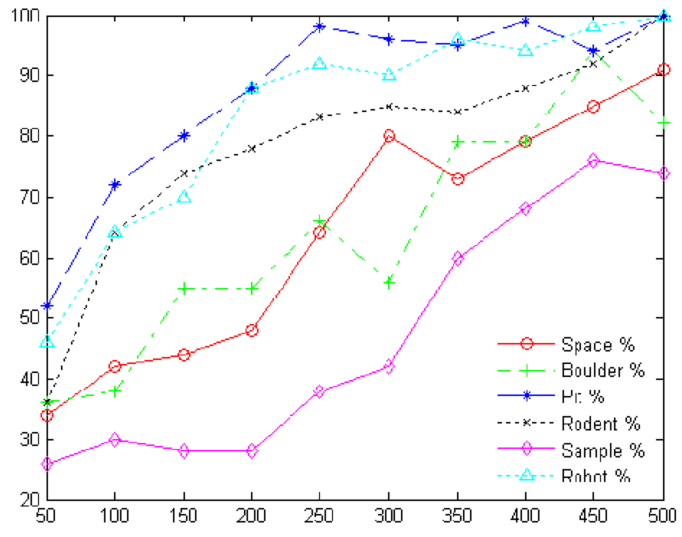

Fig. 3. Success rates, iteration by iteration

In some cases, however, success rates were achieved at a much slower pace. This was due to a utility dilution from a number of "good" but not "best" outcomes generated. In these cases, the robot was able to generate positive utility, but the utility was less than desirable. The test algorithm used does not distinguish between the any given behavior, but rather encourages behavior to the degree utility is affected. As a result, "good" behaviors are encouraged to a lesser degree, but encouraged nonetheless. While still increasing utility, this dampens the desired successful score and ability to recognize the best course of action.

Test Case 2 - Random Encounters

In the Hazardous Environment Simulation, the robots begin exploring the simulation looking for samples. A randomizer generates various hazards which the robot must successfully overcome. Fig. 4 shows how, over time, the central controller is able to "learn" the best actions to overcome any given obstacle, even those actions, such as going back to base to get missiles or drop off samples, which are combinations of primitive moves. By the end of the 2000 iterations, the success rate has moved from under 50% to over 90%.

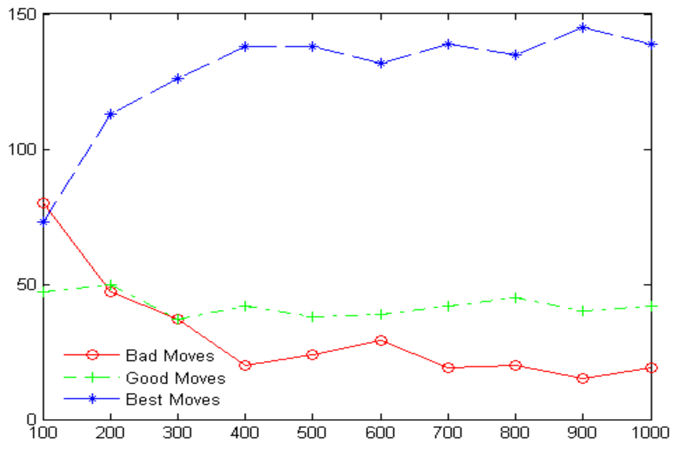

Fig. 4. Comparison of robot moves, Bad, Good, Best, during the simulation

Over time, as demonstrated by the graph, the Robots became much more successful in surviving the various hazards they encountered. Fig. 5 shows that, despite a number of alternative behaviors that lead to robot casualties, and a limited supply of charges, the robots were able to learn what to do in order to turn the tide against the rampaging rodents and even figure out how to pick up and return samples at the end. One interesting result was the long time it took for the robots to learn to collect and return samples. This behavior took much longer to learn than the use and recharging of the prod. The reason was the algorithm's applying significant negative reinforcement for rodent encounters, often resulting in robot casualties, while failing to do so for lost samples, which merely resulted in lost opportunity. These results demonstrate how much more effective a combination of positive and negative reinforcement can be over an approach that relies upon positive or negative reinforcement alone.

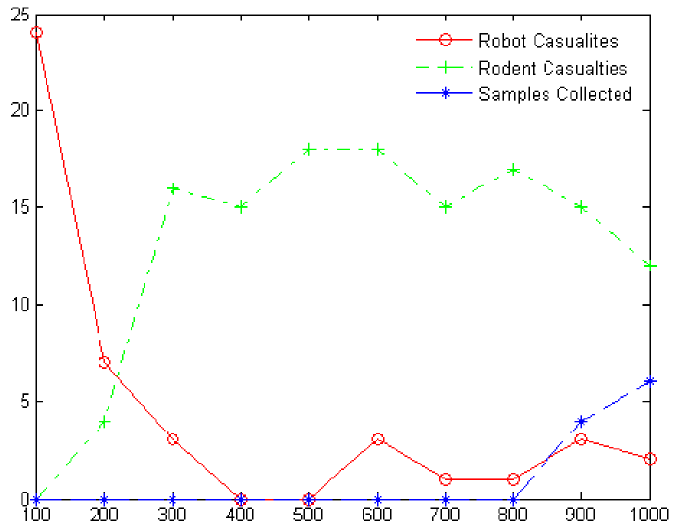

Fig. 5. Comparison of casualties, Robot vs. Rodent and Samples Collected

The results from the Hazardous Environment Simulation test examples demonstrate that cooperative learning can achieve very fast behavioral adaptations, even if the individual robots have no intrinsic behavioral intelligence to begin with. Throughout the entire simulation, the robots never possessed any preprogrammed rules for how to behave except those provided by the central controller. Random attempts to find a solution resulted in a number of early losses, but the information gained by the central controller allowed the remaining robots to more than make up for those losses. Despite initial random behavior, the controller was able to quickly learn how to best play the game through positive and negative reinforcement and pass that information along to the robots so they could achieve their goal.

IV. Conclusions and Future Work

The Hazardous Environment Simulation test example shows that use of a central controller to analyze inputs and create rules for behavior can allow even unintelligent robots to interact successfully and incrementally improve their performance across a variety of unknown situations if given enough time and resources to explore available options. A central controller with an already existing knowledge base and adaptive algorithms will serve to increase the robots' adaptive capabilities even further.

The overall configuration allowed the robots to take advantage of central controller's greater processing capability while the CC took advantage of the robots' ability to explore collectively and across a wide area. Despite having no initial knowledge base, the CC was able to gather and assess enough information to create a working and improving knowledge base very quickly and adapt robot behavior successfully. For the robots, despite no intrinsic intelligence from beginning to end, they were able to "learn" from the central controller how to successfully overcome a series of hazards and defeat the rodents. Initial losses were significant in the beginning but the robots eventually were able to obtain their respective goals and win the simulation.

Much research exists [2], [7], [10], describing advanced techniques for implementing adaptive behavior in robots. Future work consists of employing one or more behavioral techniques such as particle swarm optimizations to better direct robot behavior autonomously without requiring massive computing power. Advanced data mining techniques such as neural networks and evolutionary algorithms will also help the central controller to more quickly discover suitable adaptive behaviors for the robots. Another approach using fuzzy type 2 constructs, such as those described in CoFuH-DT [15], could prove useful in deriving contextual rules to drive robot behavior under more specific and overriding conditions.

References

[1] T. Umetani, Y Mae, K. Inoue, T. Arai, *Adaptive Relocation of Environment-Attached Storage Devices for Effective Knowledge-Sharing among Multiple Robots*, IEEE/ASME International Conference on Advanced Intelligent Mechatronics, July 2001.

[2] H. Liu, H. Iba, *Multi-Agent Learning by Evolutionary Subsumption*, IEEE International Conference on Evolutionary Computation, Dec. 2003.

[3] M. Dai, Y. Huang, *Data Mining Used in Rule Design for Active Database Systems*; 4th International Conference on Fuzzy Systems and Knowledge Discovery, June 2007.

[4] National Institute of Standards and Technology. "Tests Check Out Rescue Robots' Life-saving Vision." ScienceDaily 12 June 2008. 7 December 2008 <http://www.sciencedaily.com /releases/2008/06/080612100442.htm>.

[5] J. W. Crandall, M. L. Cumming, *Identify Predictive Metrics for Supervisory Control of Multiple Robots*, IEEE Transactions on Robotics, Oct. 2007.

[6] T. Fujita, H. Kimura, *Tight Cooperative Working System by Multiple Robots*, IEEE International Conference on Intelligent Robots and Systems, Oct. 1998.

[7] N. Kubota, M. Mihara, *Multi-Objective Behavior Coordination of Multiple Robots Interacting with a Dynamic Environment*, IEEE International Conference on Fuzzy Systems, March 2003.

[8] S. Akella, S. Hutchinson, *Coordinating the Motions of Multiple Robots with Specified Trajectories*, IEEE International Conference on Robotics and Automation, May 2002.

[9] C.K. Tsai, *Multiple Robot Coordination and Planning*, IEEE International Conference on Robotics and Automation, April 1991.

[10] X. Ma, Q. Zhang, Y. Li, *Genetic Algorithm Based Multi-Robot Cooperative Exploration*, IEEE International Conference on Control and Automation, June 2007.

[11] A. Rodrigues Neto, G. de Campos, J. de Souza, M. Roisenberg, V. Marques, *Autonomous Agents and Subsumption as Models for Simulations of Population Dynamics*, IEEE International Conference on Machine Learning and Cybernetics, July 2008.

[12] J. Han, M. Kamber; *Data Mining Concepts and Techniques*, 2nd Ed, Morgan Kaufmann Publishers, 2006.

[13] S. Chen, H. Wu, X. Han, L. Xiao, *Multi-Step Truncated Q Learning Algorithm*, IEEE International Conference on Machine Learning and Cybernetics, August 2005

[14] Y. Yang; Y. Tian, H. Mei, *Cooperative Q Learning Based on Blackboard Architecture*, International Conference on Computational Intelligence and Security Workshops, Dec. 2007

[15] K. McCarty, M. Manic; *Contextual Fuzzy Type-2 Hierarchies for Decision Trees (CoFuH-DT) – An Accelerated Data Mining Technique*; IEEE International Conference on Human System Interaction, May 2008.