Applying Dynamic Conditions to an Auction Behavior-Based Robotic Architecture

Bradford A. Towle Jr., and Monica Nicolescu
University of Nevada, Reno, Nevada, United States of America
towle@cse.unr.edu, monica@cse.unr.edu

Keywords: Behavior-Based, AI, Distributed, Auction, HRI

ABSTRACT

Robotic systems in the real world will work in dynamic environments and have to choose between multiple, and occasionally conflicting goals. In order to facilitate these two requirements an Auction Behavior-Based Robotic Architecture (ABBRA) was developed that allowed different behaviors to compete by bidding for control of a robot. Each behavior would bid with an activation level calculated from metrics based on the environment. This paper introduces two new features to the Auction Behavior-Based Robotic Architecture, which are the dynamic addition of a goal at run time and the ability to set time constraints on more than one behavior. Dynamic situations may require these new features in order to ensure a strict critical timing and adaptability to the environment. Coupling these two features together with the existing ABBRA system increased the goal selection performance across six test scenarios.

1. Introduction

Robots in a real-world environment can face several difficult problems. The real world is a dynamic environment that may impose time constraints, allow asynchronous input from humans and have situations where a robot may encounter multiple conflicting goals. Because of these issues, we developed an auction behavior-based robotic architecture (ABBRA)[1], which currently can adapt to most of these situations. Not only will ABBRA adapt to dynamic situations but it also attempts to choose the best goal in order to minimize execution time. To accomplish this, each behavior (module) must compete, or auction, with one another for control of the robot. Thus, this system uses a competitive winner-take-all action selection mechanism.

In addition to arbitrating between goals ABBRA can handle multiple conflicting goals requesting control simultaneously. Only a small number of robotic architectures can handle multiple conflicting goals without a static priority configuration such as the use of inhibition signals between behaviors where one behavior will deactivate another [2, 3].

Despite the above capabilities of ABBRA certain dynamic situations required the addition of two new dynamic features to the previously proposed auction behavior based robotic architecture [1]. The original architecture allows individual behaviors to challenge each other for control by bidding with an activation level. The activation level represents the importance of a certain goal. Metrics from the outside world derive the activation level, thus, the architecture can make the most opportunistic decision based on the most recent known values from the environment [1].

The two new features are as follows: i) the ability to allow temporal components into more than one behavior and ii) the ability to request multiple goals during runtime. Allowing the architecture to consider temporal constraints for more than just one behavior incorporates scheduling into the robotics decision making. The robot can now judge behaviors based on how much time they have left and determine whether they are critical or not. This also gives the robot the ability to postpone a lesser time-critical task in lieu of more opportunistic tasks. The second component, dynamic addition of a task during runtime, makes the architecture more robust against unknown changes and allows for a non-static architecture. Most behavior-based architectures require a recompilation or at least reset after adding a new module due to the usually static interconnectivity of the behaviors. However, the ABBRA system will simply add the new module into the system, and start running it on the next program cycle.

The paper is structured as follows: Section II provides related work. Section III gives a more in depth look at the robotic architecture and the new capabilities added to it. Section IV presents the results taken from six different tests. Section V evaluates the findings. Section VI provides a brief look into the future work of this project followed by the conclusion.

2. Related Work

There are several major action selection mechanisms for robotic decision-making. Deliberative architectures plan and create a ideal solution for the known world [4, 5]. Problems arise when these systems encounter dynamic environments, because the robot must re-plan whenever it detects the situation has changed. As an alternative, voting allows each behavior choose the best action for itself. This works well when behavior outputs are similar enough to estimate which action is closest to each behavior's desired goal. To this end, the architecture must also select which actions will be voted on in the first place [6]. Arbitration is another option in which a robot has multiple behaviors, but must choose only one to execute. However, when multiple goals that conflict [2] arise, the architecture must prioritize [7] or select which behavior is most applicable [8].

The ABBRA project uses a winner-take-all method. In this approach, the robot must choose between a set of possible behaviors and choose the one it wishes to perform. The problem arise when handling conflicting goals and allowing the architecture to change goals when it is opportunistic for the robot. Activation Networks [2] solve conflicting goal by allowing behaviors to promote other behaviors by injecting "activation energy". The behavior that has the most activation energy will win control. Although similar, ABBRA does not use inter-behavior communication to promote activation for a certain behavior. Instead, ABBRA uses the environment to determine which behavior is most efficient to run. This follows the standard behavior based paradigm where data from the environment provides state information [9]. Generally, behavior based paradigms will use this information to prioritize goals [7] or use inhibition signals to prohibit conflicting goals [3]. ABBRA extends this concept, instead of simply prioritizing goals, it will allow them to compete and dynamically change their priority.

Market-based approaches are widely used in multi-agent robotic systems. Since the seminal paper [10] the number of market based robotic papers has increased dramatically increased [11-20]. However, these papers focus on multi-agent (multi-robots) and solve a different problem than ABBRA. Here are some key differences: 1) The first difference is that multi-agents systems involve robots competing for task where as ABBRA deals with behaviors competing for control over actuators or other robotic resources; 2) Multiple agents bid for tasks whenever they becomes available where as in ABBRA the bidding occurs continually; 3) Multiple agent systems must monitor the robot who won the tasks to ensure that it is performing well [13] - in ABBRA if a task does not perform well another task will out-bid it on the next cycle; 4) Multi-agent systems must worry about external conflicts between robots, where ABBRA resolves conflicting goals by allowing environmental and temporal metrics to influence which behavior has won [21]; 5) Conversely, a lot of research has been done to allow individual agents to cooperate with each other – in ABBRA behaviors that are simultaneously running will automatically be capable of helping other behaviors [14]. Because of these differences, the problem ABBRA solves and the domain of multi-agent systems are substantially different.

3. The Architecture

ABBRA allows multiple behaviors to challenge each other for control of an actuator in order to accomplish their specific objectives. Each behavior calculates an activation level that describes its priority to run. The activation level is calculated by multiple metrics form the environment. These metrics represent the "closeness" of the desired goal state to the current state of the robot. This ensures that the activation level describes the most up-to-date environment as possible [9]. Each metric is divided by the maximum value of that particular metric's type. All active metrics in a behavior are then summed and averaged resulting in a normalized activation level that can be compared to all other behaviors. Equation 1 (next page) details this calculation.

Subtracting from one inside the summation allows the user to specify that a lower metric value is more desirable. Without it a higher metric value would yield a higher activation level. After calculating the activation level, each behavior will bid for control of a mutex, which grants access to various robot actuators or any other limited resources. The highest bid wins control for that round. This causes an emergent behavior where the robot will try to handle the easiest/most important tasks first [1] yet it is still capable of changing behaviors if necessary.

$$\text{Activation Level} = \frac{\sum_{All\ metric\ types} 1 - \frac{BehaviorsMetricValue}{Max\ value\ for\ Metric\ type}}{Number\ of\ used\ Metrics}$$

Equation 1: Normal calculation for activation levels.
BehaviorsMetricValue - Value of a particular metric for a particular behavior wishing to bid for an actuator.
MaxValueforMetricType - The maximum value of the particular metric for all behaviors challenging.
NumberofUsedMetrics - The number of metrics used in the particular challenging behaviors.

It is desirable that the robot using ABBRA could handle multiple goals with varying degrees of time constraints. For example, behavior "A" must end in five minutes, but behavior "B" must end within 35 seconds, the robot should choose behavior "B" with all other metrics being equal. However, if two goals have large time constraints while only requiring a small amount of time, there should be very little contribution from the time constraints to the activation level. Thus, equation 1 does not work. Consider a scenario where two goals both take five minutes to complete: one has a time constraint of 30 minutes and the other has a time constraint of an hour. The time constraint is almost pointless in this scenario. However, if the time constraint is closer to zero then the it should be important even if the difference between another goal is only a couple of minutes. In order to handle this situation the architecture uses a different fitness function for the temporal metric. The percentage of time remaining applied to Equation 2 (next column) gives the appropriate non-linear response to the activation level. Originally, the time-to-finish was to be used. However, many behaviors' completion time are difficult to estimate. Therefore, we used the time-to-start metric where the time before a behavior must execute can be estimated and used in the activation level calculation.

$$ContributionFT = e^{-10*(\mathrm{PRS})}$$

Equation 2: Calculating the contribution for time remaining
PRS - The percentage of time that remains before the task MUST start.
ContributionFT = This the value that will be added to the summation in equation 1 for time instead of equation 1.

Another newly added component of the ABBRA is the capability for the architecture to add a new goal into the system after the system started running. This allows the architecture to add new goals for situations it did not know existed. This process was straightforward for the ABBRA system. The user, or another program, adds a new goal, which in turn adds its own behaviors into the architecture. The next bidding cycle calls these behaviors as if they have been there since the robot had started running, and then the new behaviors begins to compete for control. Currently no inter-connecting inhibition signals are required because ABBRA does not use inhibition signals.

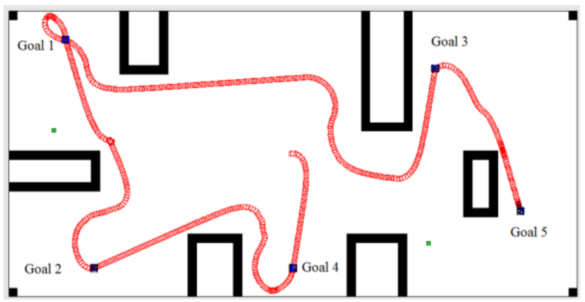
4. Experimental Results

Six scenarios were designed to test the new features. There are six goals in each test, each containing five locations that the robot must navigate using a "Go-To-Point" behavior. The sixth goal uses "Center-on-Green object", which centers the robot at a green object when one is found. The first tests provided a simple control run containing no time constraints, and no goals were added at run time. The second scenario tested the possibility of having two known goals (locations five and six) with a strict time constraint. Scenario three provided the same analysis as scenario two except that this time the location of the goal was unknown because it gave "center on green object" a stricter time constraint. Tests four and five dynamically added a new goal during runtime. This demonstrated what would happen if the architecture added a goal with an urgent time constraint. Test four added "Center-On-Green-Object" midway through the scenario execution, likewise scenario five added "Go-To-Point" on goal five during the execution. The sixth scenario added the same goal as scenario five however, no time constraints were placed on the new goal. These tests were run on the player/stage package simulating a Pioneer 3 robot.

4.1 Experiment 1: No Constraints

The first test demonstrates a typical run where the robot must travel to five goal locations and center itself and its camera on a green object, whose location is unknown. Figure 1 details the path the robot took to reach the goal.

This first run contained no time constraints, thus no goal had a smaller scheduled time for completion than any other goal. Therefore, the robot takes a "shortest distance first" approach and centers itself on the first green object it can find. Figure 1 (bottom) details the order in which the processes ran. Without any interruptions or time constraints, the robot simply executed the closest goal first interrupting Goal 1 only to center on the green object that it detected.



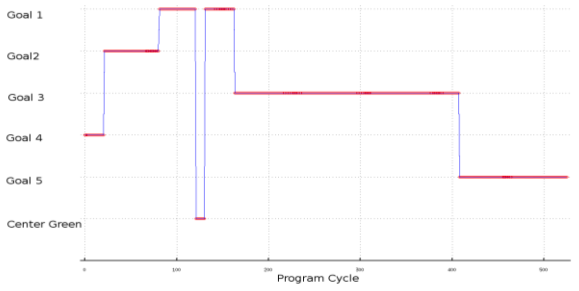
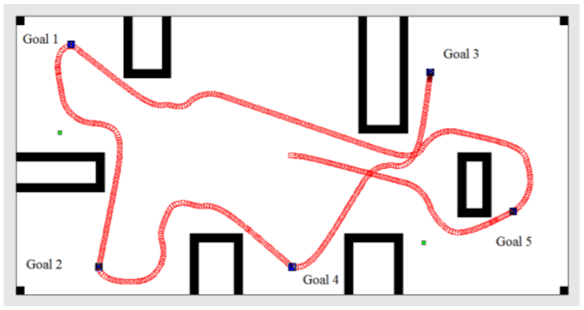


Figure 1: Path taken by on the first run (Top), Order of goals taken (Bottom)

4.2 Experiment 2A : Time Critical Constraint for Center on Green Object

The second set of tests focused on goals with time constraints. The first time-critical test configures Goal 1 and Goal 5 with a very urgent time constraint. The time constraints are set so it reaches zero immediately. In real life, this situation would be unlikely but it would be equivalent to having two emergencies occurring at once.

The following images show the order in which the processes of Test 2A executed. The robot immediately moves to finish Goal 5 and then goes toward Goal 1. The robot did not visit Goal 3 although it came in close proximity to this goal. This demonstrates the importance the robot put on Goal 1.



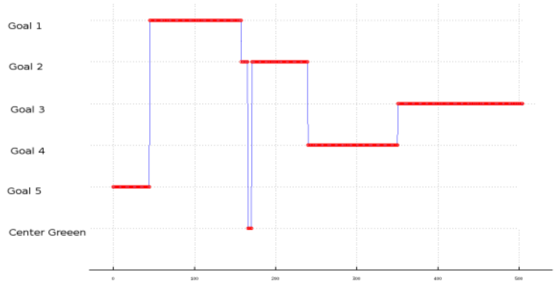


Figure 2: Path for Test 2A (Prev. Page), Order for Test 2A (Top)

4.3 Experiment 2B: Time Critical Constraint for Goal 5

Part B of the second test was setting an unknown goal, such as “Center-On-Green-Object”, to have a time critical requirement. In the following figure the “Center-On-Green-Object” is running instead of “Go-to-Point”, therefore, it uses a slightly different obstacle-avoid procedure designed to wander a maze. The robot did encounter Goal 3 - However, this was purely coincidental. Notice the robot avoids Goal 5 until finally discovering the green object in the right hand corner. Figure 3 (next page top) details the path of the robot and Figure 3(next page bottom) shows the order of goal fulfillment.

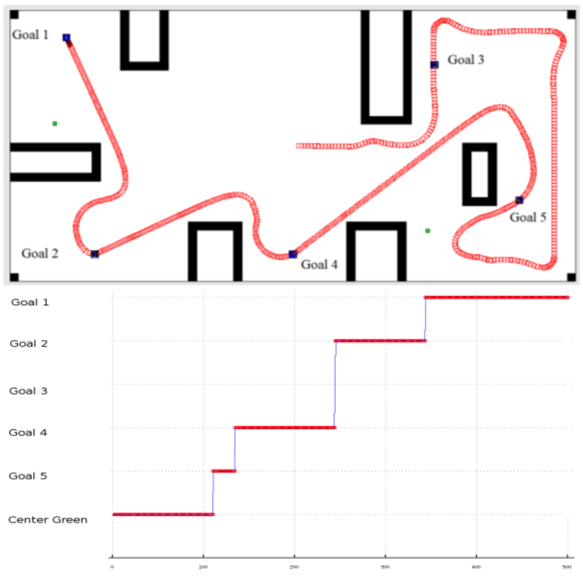


Figure 3: Path for Test 2B (Top), Process order for Test 2B (Bottom)

4.4 Experiment 3A: Dynamically Adding Center on Green Goal with Time Constraint

The third test introduces a new goal with an urgent time constraint during run-time. A common problem in robotics is that once an architecture has started to run it cannot dynamically add modules to it. The ABBRA system handles it as a new competitive module. The following figure details the path taken by the robot: As it is rounding the corner for Goal 1 the "center on green object" behavior is added, thus the behavior is spawned. The process order of the third test shows where the "Center-On-Green-Object" started and when it interrupted the first goal. Note on Figure 4 (Bottom) the blue "X" marks the step in which "Center-On-Green-Object" dynamically added itself to the architecture.

4.5 Experiment 3B: Dynamically Adding Goal 5 with Time Constraint

The second portion the third test repeated the same test, but this time it would dynamically add Goal 5. The results for this are more dramatic. As the robot goes around the corner to reach goal one, Goal 5 with a short time constraint is created. This forces the robot to abandon the current goal and travel across the map to Goal 5 (Figure 5).

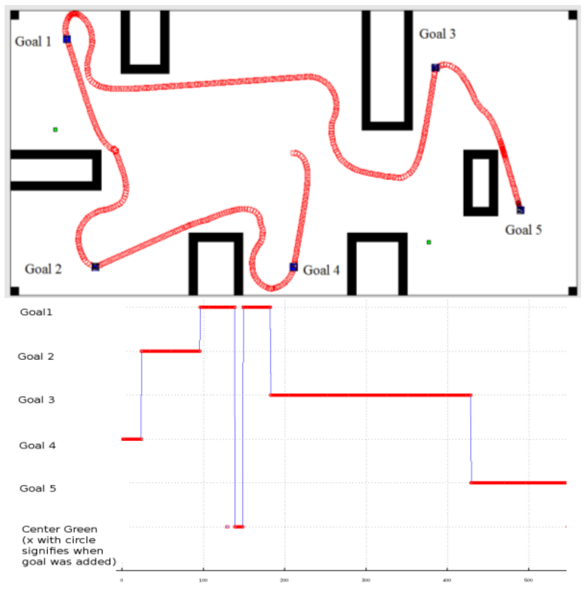


Figure 4: Path for Test 3A (Top), Order of goals for 3A (Bottom)

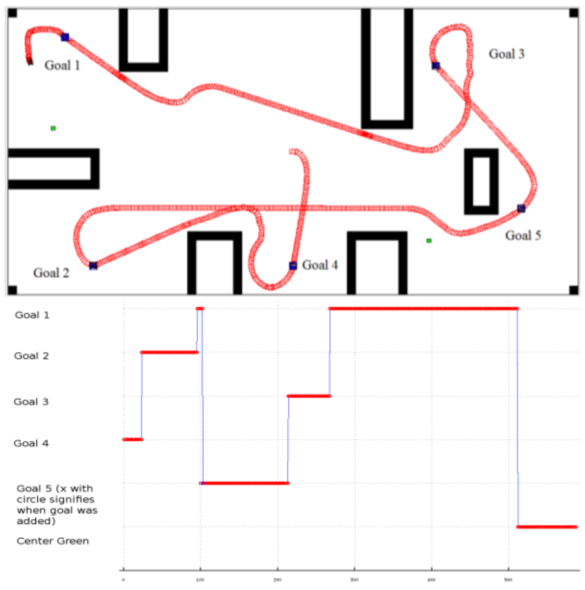


Figure 5: The path taken by the robot for Test 3B (Top), Order of goals for Test 3B (Bottom)

The blue "x" close to the 100 mark on Figure 5 (Bottom), mark indicates the step in which Goal 5 was added to the system. Since the behavior associated with goal five has an urgent time constant, it immediately takes over and begins running.

4.6 Experiment 4: Dynamically Adding a Goal without Time Constraint

In this test, Goal 5's addition occurs at the same time as in the previous test; however, Goal 5 has no time critical component. This test demonstrated that the addition of a goal with relative equal temporal priority would have very little effect on the system. As the results show, Goal 5 has no impact when added with no time constraints (Figure 6). Figure 6 (top) details Goal 5 being added and no real change occurring to ABBRA's order of goal fulfillment.

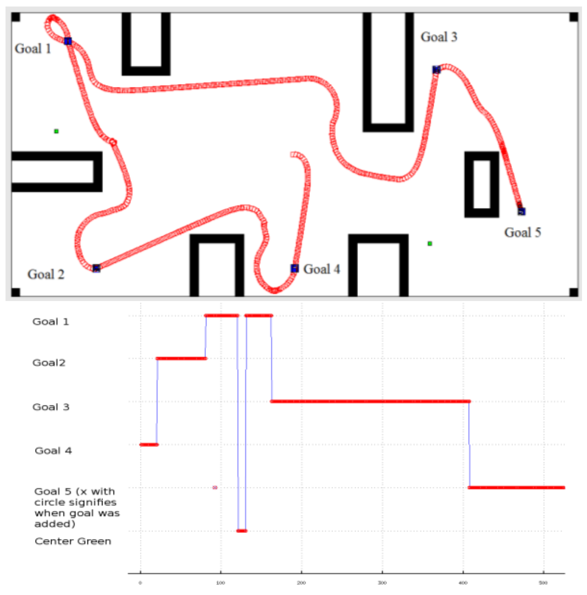


Figure 6: Path taken for Test 4 (Top), process order for Test 4 (Bottom)

5. Discussion

The data shown above supports the following two claims: first, allowing the robot to consider temporal constraints gives the robot a much more adaptive ability. For example, the robot may change the time priority of a goal in case an emergency arises. The time constraint could also allow the robot to have safety thresholds. For instance, if a robotic arm dealing with chemicals required the addition of extra chemicals with accurate timing, the architecture could consider this. Allowing multiple time constraints gives the user the ability to specify that a goal can execute whenever there is no other time critical goals challenging. The user could also ask the robot to do something when it was convenient - if the robot had pre-existing goals that contained a time critical component, it would finish those goals first.

Second, the ABBRA also allows the robot to handle additional goals given to it after startup. This is necessary in order to ensure that the robot can adapt to new challenges, requests and obstacles. One of the most common use cases for adding a goal dynamically is allowing the user the ability to add a new task requirement for a robot. This could be through any medium as long as the goal is registered and introduced into the system. The robot may also detect an obstacle that was unexpected, and overcoming this obstacle may require dynamically adding a new task.

6. Future Work

This paper demonstrates the importance of the ABBRA's adaptability. The next step of this work incorporates this system into the real world by testing it on a physical robot. Future work is also needed regarding human robot interaction (HRI). If humans interact with the robot, the architecture must choose between performing a task and interacting with the human. A way of dealing with this consists of using teamwork between the robot and human [25-30]. However, the user must still interact with the robot in terms of behaviors and goals; this requires the architecture to fuse low level behaviors into a level of abstraction users want to interact in [31]. Incorporating SLAM (Simultaneous Localization and Mapping) increases the accuracy of the architecture substantially. If the robot can identify real distance to goals versus straight-line distance, the architecture will have a more accurate understanding of its current state in the environment.

7. Conclusion

In conclusion, the ABBRA gives the robot the ability to adapt to dynamic environments. By allowing temporal constraints in the behavior's activation level calculation, the robot can prioritize a task based on temporal requirements. This paper demonstrates the ability for the robot to introduce a new goal into the architecture that was not initially running at the beginning of runtime. The robot can use these new features coupled with the older features provided by the ABBRA to adapt to a wide variety of dynamic environments as demonstrated by the scenarios presented.

8. References

[1] B. A. Towle, and M. Nicolescu, “Fusing Multiple Sensors through Behaviors with the Distributed Architecture,” in 2010 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems, Salt Lake, Utah, 2010, pp. 115-120.

[2] P. Maes, “How to do the right thing,” *Connection Science,* vol. 1, no. 3, pp. 291-323, 1989.

[3] M. Proetzsch, T. Luksch, and K. Berns, “Development of complex robotic systems using the behavior-based control architecture iB2C,” *Robotics and Autonomous Systems,* vol. 58, no. 1, pp. 46-67, Jan, 2010.

[4] R. Volpe, I. Nesnas, T. Estlin *et al.*, “The CLARAty architecture for robotic autonomy,” in Aerospace Conference, 2002, pp. 1.

[5] T. Estlin, R. Volpe, I. Nesnas *et al.*, "Decision-making in a robotic architecture for autonomy."

[6] J. Rosenblatt, and C. Thorpe, "Combining multiple goals in a behavior-based architecture." pp. 136-141.

[7] R. A. Brooks, “A ROBUST LAYERED CONTROL-SYSTEM FOR A MOBILE ROBOT,” *Ieee Journal of Robotics and Automation,* vol. 2, no. 1, pp. 14-23, 1986.

[8] J. Koseck , and R. Bajcsy, “Discrete event systems for autonomous mobile agents,” *Robotics and Autonomous Systems,* vol. 12, no. 3-4, pp. 187-198, 1994.

[9] R. Brooks, “Elephants don't play chess,” *Robotics and autonomous systems,* vol. 6, no. 1-2, pp. 3-15, 1990.

[10] R. Davis, and R. G. Smith, “Negotiation as a metaphor for distributed problem solving,” *Artificial intelligence,* vol. 20, no. 1, pp. 63-109, 1983.

[11] F. Brandt, W. Brauer, and G. Weiss, “Task assignment in multiagent systems based on vickrey-type auctioning and leveled commitment contracting,” *Cooperative Information Agents IV-The Future of Information Agents in Cyberspace,* pp. 11-44, 2004.

[12] P. Faratin, C. Sierra, and N. R. Jennings, “Negotiation decision functions for autonomous agents,” *Robotics and Autonomous Systems,* vol. 24, no. 3-4, pp. 159-182, 1998.

[13] B. P. Gerkey, and M. J. Mataric, “Sold!: Auction methods for multirobot coordination,” *Robotics and Automation, IEEE Transactions on,* vol. 18, no. 5, pp. 758-768, 2002.

[14] B. Jennings, and Å. Arvidsson, “Co-operating market/ant based multi-agent systems for Intelligent Network load Control,” *Intelligent Agents for Telecommunication Applications*, pp. 71-71, 1999.

[15] H. Jung, M. Tambe, and S. Kulkarni, "Argumentation as distributed constraint satisfaction: Applications and results." pp. 324-331.

[16] R. Krovi, A. C. Graesser, and W. E. Pracht, “Agent behaviors in virtual negotiation environments,” *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on,* vol. 29, no. 1, pp. 15-25, 1999.

[17] M. J. Matari , G. S. Sukhatme, and E. H. Østergaard, “Multi-robot task allocation in uncertain environments,” *Autonomous Robots,* vol. 14, no. 2, pp. 255-263, 2003.

[18] R. G. Smith, “The contract net protocol: High-level communication and control in a distributed problem solver,” *Computers, IEEE Transactions on,* vol. 100, no. 12, pp. 1104-1113, 1980.

[19] K. Sycara, and D. Zeng, “Coordination of multiple intelligent software agents,” *International Journal of Cooperative Information Systems,* vol. 5, no. 2, pp. 181-212, 1996.

[20] M. P. Wellman, and P. R. Wurman, “Market-aware agents for a multiagent world,” *Robotics and Autonomous Systems,* vol. 24, no. 3-4, pp. 115-125, 1998.

[21] M. B. Dias, and A. Stentz, *Traderbots: A market-based approach for resource, role, and task allocation in multirobot coordination*: Citeseer, 2003.

[22] D. Ferguson, Y. Yemini, and C. Nikolaou, "Microeconomic algorithms for load balancing in distributed computer systems." pp. 491-499.

[23] L. M. Ni, C. W. Xu, and T. B. Gendreau, “A distributed drafting algorithm for load balancing,” *Software Engineering, IEEE Transactions on,* no. 10, pp. 1153-1161, 1985.

[24] I. Ahmad, and A. Ghafoor, “Semi-distributed load balancing for massively parallel multicomputer systems,” *IEEE Transactions on Software Engineering*, pp. 987-1004, 1991.

[25] T. Fong, N. Cabrol, C. Thorpe *et al.*, "A personal user interface for collaborative human-robot exploration."

[26] T. Fong, C. Thorpe, and C. Baur, *Collaborative control: a robot-centric model for vehicle teleoperation*: Carnegie Mellon University, The Robotics Institute, 2001.

[27] T. Fong, C. Thorpe, and C. Baur, “Multi-robot remote driving with collaborative control,” *Ieee Transactions on Industrial Electronics,* vol. 50, no. 4, pp. 699-704, 2003.

[28] T. Fong, C. Thorpe, and C. Baur, “Robot, asker of questions,” *Robotics and Autonomous Systems,* vol. 42, no. 3-4, pp. 235-243, 2003.

[29] G. Dorais, R. Bonasso, D. Kortenkamp *et al.*, "Adjustable autonomy for human-centered autonomous systems."

[30] A. Fereidunian, M. Lehtonen, H. Lesani *et al.*, "Adaptive autonomy: smart cooperative cybernetic systems for more humane automation solutions." pp. 202-207.

[31] M. Nicolescu, O. Jenkins, and A. Stanhope, "Fusing robot behaviors for human-level tasks." pp. 76-81.