An Auction Behavior-Based Robotic Architecture for Service Robotics

Bradford A. Towle Jr.
University of Nevada, Reno
Scrugham Engineering/Mines (SEM) 242
College of Engineering | University of Nevada,
Reno/0171, Reno, NV 89577-0171
Phone: (775) 784-6974
Fax: (775) 784-1877
towle@cse.unr.edu

Monica Nicolescu
University of Nevada, Reno
Scrugham Engineering/Mines (SEM) 242
College of Engineering | University of Nevada,
Reno/0171, Reno, NV 89577-0171
Phone: (775) 784-1687
Fax: (775) 784-1877
monica@cse.unr.edu

Abstract – Service robots have the potential of improving the quality of life and assist with people's daily activities. Such robots must be capable of operating over long periods of time, performing multiple tasks, and scheduling them appropriately for execution. In addition, service robots must be capable of dealing with tasks whose goals may be in conflict with each other and would need to determine, dynamically, which task to pursue in such a case. Adding to the complexity of the problem is the fact that some task requests may have time constraints – deadlines by which the task has to be completed. Given the dynamic nature of the environment, the robots must make decisions on what tasks to pursue in situations where there could be incomplete or missing information. The robots should also be capable of accepting requests for new tasks or services at run-time, while possibly working on another task. In order to achieve these requirements this paper presents the Auction Behavior Based Robotic Architecture (ABBRA) that brings the following contributions: i) it uses an auction mechanism to determine the relevance of a task to run at any given time, ii) it handles multiple user requests while dealing with potentially critical time constraints and incomplete information, iii) it enables long-term robot operation and iv) it allows for dynamic assignment of new tasks. The proposed system is validated on a physical robotic platform, the Segway RMP® and in simulation.

Keywords *Auction-based robotic architecture - Intelligent action selection - Dynamic time and environmental constraints- Behavior Based Robotics*

1 INTRODUCTION

Service robots can provide significant assistance to people in their daily activities, reducing the workload or serving as helpers for users needing assistance with achieving certain goals. The complexity of the environment in which service robots would need to operate and the number of issues that such robots have to overcome pose significant challenges for the development of a robot control architecture that can handle them appropriately.

First, a service robot must be capable of operating over long periods of time, in dynamic environments, and be continuously prepared to accept new task requests from users, even if the robot is still in the process of working on another task. Therefore, the robot's control architecture must be versatile enough to integrate new tasks at runtime and also to handle the interruptions caused by the new addition of the new tasks.

Second, the robot must be able to efficiently schedule multiple task requests for execution. This requires a task selection mechanism that can choose what task is most appropriate to run at any given time. This requirement is particularly important when dealing with tasks whose goals may be in conflict with each other. Such conflicts may occur when two or more tasks require a specific resource on the robot, in order to achieve different goals. For example, a conflict will arise if a task requires the robot to pick up an object while another requires the robot to empty the content of its gripper: the goals of the two tasks are conflicting, as each task would undo the progress of the other. A solution to this problem is the use of a pre-defined, static priority [1] that would select which task to run first. A deliberative approach could resolve conflicting goals, however it may not always be feasible in dynamic environments [2]. In practice, it is hard to decide what task should have precedence over another or to use a deliberative system, especially given that task requests are coming at run-time and their order is not a priori known. An effective solution to this problem would enable the robot to dynamically decide what task to perform given the current task workload and environmental situation.

A third challenge for a service robot is that certain tasks may need to be completed by given deadlines, thus posing time-constraints that can vary in length and importance. Some time-constraints are critical to adhere to, while others are simply preferential for the user. The robot must also handle tasks for which there is unknown or incomplete information. This reinforces the need for determining the relevance of each task dynamically and would allow the robot to adapt and switch tasks if necessary should new information be discovered in the environment.

Finally, service robots must be capable of functioning autonomously in dynamic environments. Any changes in the world could impact a robot's current plan of performing its tasks. Thus, the robot must be capable of adapting to the new situations and be able to change its task execution plan, if necessary.

In order to address the above challenges, this paper presents the Auction Behavior Based Robotic Architecture (ABBRA), which introduces the following contributions: first, the architecture uses an auction mechanism to select which task to run at any given time. In this system, each task uses environmental data to determine its relevance to run at that time and uses this relevance to compete for control of an actuator on the robot in order to achieve its goal. The auction mechanism allows the winning task control of the specific robotic resource preventing multiple tasks from attempting to access it at once.

Second, the architecture handles multiple user requests while dealing with potentially critical time constraints of the tasks and incomplete information. Using the auction mechanism for task selection, ABBRA resolves conflicting goals without statically setting the priorities for tasks or by prohibiting two tasks with conflicting goal from being active at the same time. Whenever such a case may occur, the fact that a tasks' relevance is based on environmental data, will enable the robot to selecting on a single one of these goals. As a goal state becomes closer to fulfillment, the more relevant the corresponding task will be for the architecture to run and the more likely it is that the task will win the auction, thus, reducing oscillations between different tasks.

Third, ABBRA enables a robot to dynamically accept new task requests, at run-time, without the need for any designer reconfiguration. Given that an auction cycle is executed at each time step, a newly added task begins participating in the bidding as soon as it is requested, thus becoming a part of the robot's set of tasks to be done. The architecture is based on the behavior-based paradigm [1], which allows the robot to perform efficiently in a real-world, dynamic environment and to react appropriately to unforeseen changes.

This paper covers three stages of development of the proposed architecture. The first two stages were validated in a simulated environment, while the third phase used a physical Segway RMP® mobile robot.

The first stage demonstrates the basic auction mechanism for task selection, the ability to handle a task with time constraints, and the dynamic rescheduling in changing environments, for scenarios where all the tasks are requested from the start. The second stage introduces the ability to handle time constraints on multiple or all of a robot's tasks and the capability to dynamically add new requests in the system at run-time. The

third stage incorporates two additional features: a real-time clock and mapping and localization capabilities, which allow for more accurate time to completion estimation for the tasks, and thus for a better task selection mechanism for the robot.

The remainder of the paper is structured as follows: Section 2 discusses related work, Section 3 describes the details of the architecture, Section 4 provides the results of the tests for each phase, Section 5 presents directions for future work and Section 6 gives a summary of the presented approach.

2 Related Work

The architecture proposed in this paper is related to three major areas of research: action selection, market-based robotic systems and job scheduling. This section describes related work in each of these domains and discusses how ABBRA relates to existing approaches.

2.1 Action Selection

The ability to select the most appropriate task to perform at any given time is a critical component of the architecture proposed in this paper. The issue of action selection, or behavior coordination, has been widely addressed in robotics, with two major approaches being most commonly used: behavior *fusion* and *arbitration*.

Fusion is a cooperative approach in which the outputs of multiple behaviors are combined into a single output that is sent as command to the robot. Several methods have been proposed. A first method of combining behavior outputs is using voting [2, 3]. This mechanism allows the outputs of behaviors to vote for what they determine to be the best action. Action voting in [4] uses a neural network architecture to vote on an action suitable from perceived environment. Other methods of fusion consist of fuzzy command fusion where each behavior is synthesized by a rule-base based approach [5]. This approach can be extended into multi-valued logic [6] and a fuzzy voting system such as Fuzzy DAMN [7]. Other main fusion techniques include schemas where vectors will be added in order to derive the direction the robot should be moving [8].

Arbitration, also called winner-take-all, is a competitive action selection approach, which chooses a single behavior, from a set of multiple behaviors, to send commands to the robot's actuators. ABBRA uses an auction-based approach to determine the winning behavior and allocate the robotic resources. Should the arbiter only choose a behavior once the previous behavior is finished a stable result can be expected. However, ABBRA continually processes bids regardless of whether the currently running behavior has achieved its goal or not. This causes a problem, when tasks have

conflicting goals because the arbiter must now consider if interrupting a task will cause goal-oscillation. Goal-oscillation occurs when the robot switches between two or more task that counteracts each other. When multiple conflicting goals arise [9], the architecture must prioritize [10] or select which behavior is most applicable [11] for the current situation. Another approach to arbitration is activation networks [9], which solve the issue of conflicting goals by allowing behaviors to promote other behaviors through "activation energy" injection. The behavior that has the most activation energy will win control of the desired actuator. 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 relevant to run. This follows the standard behavior-based paradigm where data from the environment provides the needed state information [12]. Existing behavior-based approaches to dealing with conflicting goals use statically prioritized behaviors [10] or use inhibition signals between behaviors [13]. Instead of prioritizing behaviors ABBRA allows them to compete and dynamically change their priority.

Another method of action selection has been proposed using neural networks [14]. This method proposes that motor and feature codes will promote different tasks and one tasks will be stronger than the rests. This method is useful for lower level motor controls, determining a motor function based on outside stimuli, while ABBRA deals with determining what generic tasks to run. Another underlying issue is the association between code and tasks must be known in advance, while in ABBRA the decision is made dynamically. Finally, the neural network must be trained, therefore, tasks outside of the scope will require re-training. This algorithm was simulated with a relatively simple scenario, which cannot handle all of the constraints and dynamic conditions that are possible using ABBRA. Another paper used a weighted ontology to detect false positives by extending the graph functionality to allow for new nodes [15]. While the results demonstrated an impressive resistance to false positives and negatives, the system was limited to locating objects based on the locality principle. This system was able to introduce more behaviors dynamically, but they were used as sub-goals to a primary goal. In addition, no other constraints were taken into account and the testing scenario had only one goal. Although this algorithm could be expanded, it does not provide the same functionality as ABBRA.

2.2 Market-Based Systems

Market-based approaches have been used to solve distributed problems [16]. This approach works

well for multi-agent robotic systems and was successfully used for task allocation across multiple robots [17-26]. However, these methods solve a different problem than proposed in this paper. First, multiple-agents involve multiple robots competing for a task, while ABBRA needs to handle multiple behaviors competing for control over actuators or other resources on a single robot. In other words, the prize for a multi-agent auction is a task, where as the task bids in ABBRA. Even if you defined each agent in a multi-agent system as an actuator, the problem is still different. In multi-agent systems, you would have actuators bidding for a chance to perform a task, where in ABBRA the task is bidding for control of the actuator.

Second, multiple-agents typically bid for tasks whenever they become available or when an agent has completed a task. Mapping multi-agents problems will usually bid more frequently but this is because they usually for re-synchronizing the map and to determine the next area to explore. However, often these algorithms will not interrupt the tasks once it has started [27]. In contrast, ABBRA continuously bids and can interrupt any task if a different task wins the auction.

Third, multi-agent systems will sometimes monitor the robot who won the tasks to ensure that it is performing well [19]. In ABBRA, if a task does not perform well in another task will out-bid it on the next cycle in a process that is performed automatically by the system.

Fourth, multi-agent systems deal with teamwork problems and ensuring robots are working together [28] [20]. ABBRA allows behaviors on different resources to work together without explicitly defining their roles.

Because of these differences, the problem solved by ABBRA and the domain of multi-agent systems are substantially different. The research closest to our approach is presented in the work by M. K. Sahota [29]. This work allows each task to bid for control of an actuator, however, it does not deal with time or dynamic addition of tasks. The architecture is composed of two parts: Executer and Deliberator, similar to hybrid robot control. The Deliberator uses environmental context and urgency to determine which bid wins. Then the deliberator then assigns a winner and provides a weighted solution to schema actions in the Executer layer. ABBRA uses environmental data and the auction mechanism to perform task arbitration, instead of a deliberative component. In ABBRA, urgency is incorporated as a time constraint that allows the behaviors to bid for full control of the actuator.

2.3 Job Scheduling

Job shop (JSSP) and flow shop (FSSP) scheduling problems take a set of tasks and attempt to maximize the throughput of the tasks. This

problem is very similar to that which is solved by ABBRA. Both problems deal with limited resources and unknown arrival of jobs that require completion within certain constraints. However, there are differences between the two problems. For example, JSSP and FSSP do not deal with resource conflict and goal degradation. JSSP and FSSP can manage jobs across multiple resources [30], however, the majority of approaches assume the job cannot be interrupted once it begins. JSSP and FSSP can break jobs down into smaller operations to streamline the tasks [31], but these smaller operations cannot be interrupted once they begin running. These problem sets deal with jobs that simply arrive and need processing. Except for the time lost, there are no detrimental effects of a job waiting in JSSP and FSSP. In contrasts, ABBRA is capable of stopping any task and allowing another task to take control if necessary. This is because in robotics if the environment changes, it may be possible to miss a valid opportunity or worse do potential harm if the appropriate behavior cannot take control of the necessary resource.

Goal degradation occurs where performing one task will undo another task. JSSP and FSSP do not encounter this problem because their tasks are independent from each other [32]. Therefore, switching between tasks will not detract from the task previously executed. For example, research in optimizing scheduling for E-Commerce servers [33] switches to a different task of higher priority whenever it is required. However, just because a server switches to a different process does not mean the previous process is undone. This assumption that tasks do not degrade each other is the largest difference between any shop-scheduling algorithm and the robotic domain. In the literature found, none addressed any potential conflict by switching goals. In contrast, a robot must be able to finish a goal unless a condition changes making another goal more relevant. Therefore, the users are expecting the robot to converge onto a single task and finish it in a timely manner.

JSSP and FSSP do not handle time estimation for task completion. The majority of the methods reviewed expected time as a parameter for the simulation and do not estimate completion time themselves [32, 34-38]. This assumption is valid since it would be difficult to determine completion time for computational tasks, especially if it required human interaction. In one approach, the scheduler used an approximate completion time, allowing it to handle times that were not exact. However, the approach itself was not estimating the completion time [39]. This is in direct contrast with ABBRA, which can estimate completion time for tasks whose objectives are at a known location.

There are many different approaches to job-shop scheduling, either by using static priorities [33] or

resembling the subsumption architecture [10]. Once again, the use of static priority prohibits the robot from taking advantage of opportunities found in the environment. Another method used the product based-approach [30], which required the entire product to be planned out ahead of time and is thus not applicable to the field of service robots. The concept of multi-agents was also applied to schedulers [40], but these methods solve problems closer to multiple-agent research than those of service robots. Lastly, other methods dealt with finding the cost of scheduling and trying to make it more efficient [41, 42] but they did not deal with conflicting goals or the estimation of time.

As discussed in Section 3.3, the scheduling problem that needs to be solved in the context of service robots is closest to the problem of scheduling with release times and deadlines, which is NP-Complete. ABBRA provides a greedy approximation solution that schedules, at every time step, the behavior most relevant to run as given by a combination of metrics extracted from the environment. These metrics and the auction mechanism are described in detail in Section 3.4.

3 Description of the Architecture

This section provides a detailed description of ABBRA, going through the three main phases of the architecture's development. This section also provides a formal definition for the problem solved by ABBRA, along with a detailed look at the auction cycle process and activation level calculations for individual behaviors.

3.1 Three Phases of Development for ABBRA

ABBRA was developed in three phases. The first phase was a proof of concept to demonstrate that behaviors bidding for control of the actuators can resolve conflicting goals and that a robot could achieve a set of requested goals using an auction mechanism as a behavior arbitrator [43, 44]. In this phase tasks with complex goals are structured as a composition of potentially multiple behaviors that achieve the overall goals. The different behaviors compete for control of the robot's actuators. Multiple behaviors with conflicting goals are allowed to compete for the same actuator, thus demonstrating there would be no goal oscillation if multiple behaviors were active. This phase of the research also demonstrated dynamic re-prioritization of behaviors. For example, if the robot discovers that it can complete one of its assigned tasks

(which was not the one currently pursued), the architecture automatically adjusts the priority of the behaviors in order to complete the new task. This enables the robot to takes advantage of opportunities that arise in order to complete all its tasks. Phase one also showed that multiple behaviors could be run simultaneously if they did not require the same actuator. The third test also demonstrated the ability to assign a time constraint on one (only one at this stage) behavior, giving it priority over other behaviors.

The second phase of research enhanced the architecture with new capabilities. In this research phase the behavior bidding system was improved to allow for multiple tasks to be requested with temporal constraints (deadlines). In this version, the timer system relied on architectural cycles in order to track time. Therefore, the architecture used the temporal metric of *"How long until behavior must start"* in order to enforce time constraints. This phase of research also introduced the ability for dynamic addition of new task requests (and of the tasks's corresponding behaviors) during runtime. ABBRA handled the newly added behaviors without any noticeable delay and respected all of the constraints put on a new behavior on the next auction cycle. These improvements allowed for the testing of several complex scenarios, which had not been possible before.

In the third phase, the improvements to the architecture consisted of introducing mapping and localization and a real-time clock to enforce the temporal constraints [45]. ABBRA was also successfully validated on a Segway Robotic Mobile Platform (RMP). The motivation behind adding mapping was to allow the robot to take a more efficient path to goals that had known locations. The use of mapping enabled the robot to determine actual distance to its goals, which could now estimate how long a behavior would take to finish and thus to reach the goal. The second improvement in this phase was the incorporation of a real-time clock to enforce time constraints. These ensured temporal constraints were upheld and made it easier for the user to specify the time constraint. With these two modifications, the architecture was successfully implemented on a Segway Robotic Mobile Platform (RMP) and ran in a dynamic environment.

3.2 Architectural Structure

ABBRA is based on three main components: **goals**, **behaviors** and **mutexes**. A **goal** represents a task, or subtask, assigned to the robot by a user. When a task is assigned to the robot a goal object that represents the task is created. This goal object will spawn one or more corresponding behavior(s) in order to complete its objective. **Behaviors** are the functional modules that compete and send commands to the robot's

actuators. Goal objects may also choose between different behaviors in the situation where there is more than one way to complete the tasks. These newly spawned behaviors may have pre-requisites that are introduced as new goals for the system. These goal/behavior can be linked together by pre-requisites in order to form behavior chains that are capable of more performing complex tasks. The behaviors gather data from the sensors and derive a set of metrics that represents how important it is for them to run. Each behavior will bid through a **mutex** for control of the specific actuator it needs by sending the set of metrics it gathered. In the case of the behavior-chains these behaviors will not start bidding until all their pre-requisite goals are achieved. Therefore, the last added behavior on this set or chain would remain running until it completed its goal. Once this occurred, the next behavior can start running. This process would repeat until the first behavior had been completed, hence finishing the desired goal that represents the desired task. It is also important to note that the entire state of these behavior chains are represented through the environment [12]. The goal objects will notify the corresponding behavior or behaviors it spawned that it is no longer necessary to run upon completion of its objective.

A mutex is an object that acts as an auction house for a specific resource on the robot and enforces mutual exclusion of the behaviors on that particular resource. The mutex will collect bids from each behavior that bids for it and then assign a winner, which gains control of the actuator for that round (Figure 1). In the event that a behavior requires more than one actuator, an abstract behavior is used. Abstract behaviors [46] are behaviors who perform no direct actions but simply control other behaviors, as described below. Abstract behaviors do not control an actuator therefore do not have to bid for control. The tasks that require more than one actuator are represented by an abstract behavior with two or more child goal-behavior pairs that use separate actuators. None of the abstract behavior's children competes against each other and all run simultaneously. When all child behaviors reach their goal states, the goal of the abstract behavior will be complete, thus completing the tasks. This mechanism allows the architecture to avoid the multiple-prize-auction problem where the architecture needs to compare a behavior trying to get multiple actuators to a behavior that only needs one. An example of such a task is the "center camera and robot on green object" task used an abstract behavior to control two child behaviors that separately. One child behavior would center the camera on the green object while the other child behavior centered robot. Abstract behaviors are also used to fuse outputs of behaviors together - in this case the abstract behavior will bid for control a single actuator.

After the abstract behavior has won control, it will invoke a set of behaviors and fuse their outputs into a single command, which is then sent to the actuator. In this scenario, the child behaviors do not have goal object they are simply modules that can be called when necessary.

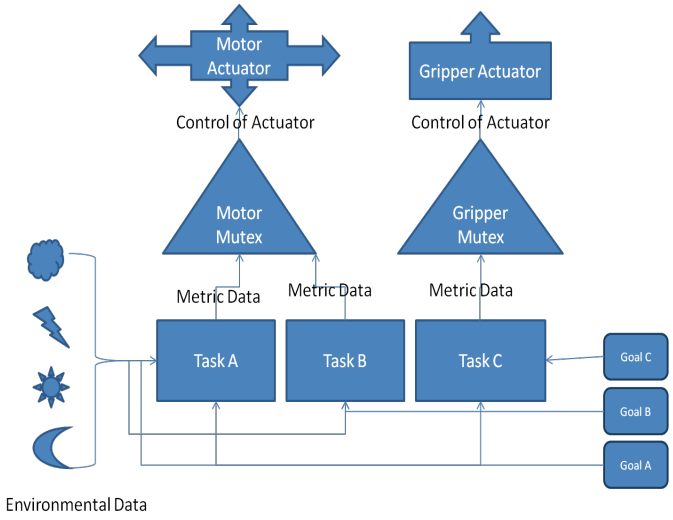

Figure 1: High-level diagram of ABBRA

3.3 Formal Description of the Action Selection Problem

This section provides a formal description of the scheduling problem solved by ABBRA's auction system. Given the following:

$R = \{R_1, R_2 \ldots R_N\}$,

a set of *N* resources available on the robot,

$B = \{B_1, B_2 \ldots B_M\}$

$TC = \{TC_1, TC_2 \ldots TC_M\}$

a set of *M* behaviors available to the robot,

a set of time constraints for each behavior (could be empty), the main objective of the architecture is to determine which behavior(s) to run at each time step in order to service all the requests efficiently (that is without missing any deadlines or minimizing the lateness for all the tasks). The resources are actuators on the robot such as the wheel motors, gripper, or swivel camera.

In the service robot domain the times when new requests are coming are not known from the beginning as users may come in with new requests at any time. Assuming that the times for the requests would be known in advance, and assuming that the robot has only one resource available, the problem of scheduling the tasks in order to minimize the lateness is an instance of scheduling with release times and deadlines which is NP-Complete. Given that for the service robot scheduling problem the request times are unknown and that there is more than one resource available, it follows that an approximation algorithm is needed to solve this optimization problem. ABBRA solves this scheduling problem as follows. Given the following sets:

$B(t) = \{B_i \in B | B_i \text{ is active at time } t\}$

$B_{R_j}(t) = \{B_k \in B(t) | B_k \text{ uses resource } R_j\}$, it is true that the behavior sets are mutually exclusive between resources (Equation 1)

$$B_{R_j}(t) \cap B_{R_k}(t) = [\,] \text{ where } j \neq k \qquad (1)$$

At any time behaviors can be introduced or deleted in the system, as a new request can come

in from a user. Therefore, the following two components need to be considered:

$$P_{R_j}(t) = \{B_k \in B | B_k \text{ has been requested by the user at time } t \text{ \& requires resource } R_j\}$$

$$Z_{R_j}(t) = \{B_z \in B | B_z \text{ no longer needs to run \& required resource } R_j\}$$

With these two components, the change in the behavior sets for a resource R_j, at time t is modeled as follows:

$$B_{R_j}(t) = B_{R_j}(t-1) + P_{R_j}(t) - Z_{R_j}(t) \quad \text{where } j = 1 \ldots N \tag{2}$$

Each behavior has a set of metrics that apply to it. Some can be spatial, such the distance to the objective, some can be temporal, such as the time remaining before the behavior must complete, and finally some may be state based such as the content of the robotic gripper. We denote the set of all metrics as:

$$M = \{m_1, m_2 \ldots m_Q\} \text{ and } M_{B_i} = \{m_{Bi} \in M | m_{B_i} \text{ is a metric for } B_i\}$$

The values of the individual metrics can change over time, which is one reason conventional scheduling does not work for this problem. This is due to the fact that an inactive behavior may have the value of its metrics changed if another behavior runs first, as the other behavior may have altered the state of the robot in the environment (e.g., moving toward a goal in one direction would increase the distance to another goal, which is in the opposite direction). We consider $Value(m_{Bi}, t, m_q)$ the value of the q-th metric for behavior B_i at time t and $Delta(m_{Bi}, t, m_q)$ the change in the value of the q-th metric for behavior B_i at time t. Thus:

$$Value(M_{B_i}, t, m_q) = Value(M_{B_i}, t-1, m_q) + Delta(M_{B_i}, t, m_q) \text{ for } q \text{ is } 1 \ldots Q$$

At this point the problem is choosing, for each resource R_j, at time t, the behavior BS_{Rj} for which the aggregate of the metric values are minimized. In other words, the architecture chooses to run the behavior with the most critical measurements from the environment (Equation 3).

$$BS_{R_j}(t) = argmin_{B_{r_j}(t)} \left(\frac{\left(\sum_{q=1}^{Q} Value(M_{B_i}, t, m_q)\right)}{Q} \right) \tag{3}$$

The set of behaviors chosen to run for all resources at time t is:

$$BS(t) = U_{j=0}^{N} BS_{Rj}(t) \tag{4}$$

3.4 Calculating the Activation Level

The value that behaviors use to bid for control of the actuators is the activation level. This value represents the importance of the specific behavior to run at a particular time. In order to derive its activation level each active behavior will use a certain set of metrics. These

metrics could be spatial distance, temporal constraints, or state variables. Once the behavior has calculated the values for its metrics, it will bid for control of the actuator by sending these values to the specific resource mutex. The mutex then determines the maximum value for each metric type from the information received from all the behaviors bidding for it. ABBRA is a single stage auction where the mutex takes the metric values given to it by the competing behaviors and calculates an activation level for each. The behavior with the highest activation level is awarded control of the actuator for that round. The activation level of a particular behavior (B) is the sum of all metric contributions divided by the total number of metrics used by the particular behavior (See Equation 5).

$$AL_B = \frac{\sum_{i=1}^{N} CM_i}{N} \tag{5}$$

where AL_B is the activation level for behavior B and CM_i is the contribution to the activation level for metric M_i.

Each one of the contributions CM_i represents the influence of a specific measurement (or metric) from the environment or the imposed time constraints. There are two methods of calculating an activation level contribution: one for non-temporal and one for temporal constraints. Some examples of metrics are: distance to goal (both Euclidean and actual distance), time remaining before the goal must be complete, starting time for the current behavior, change in the direction of the robot, blob size, and direction the camera is facing. The contribution for non-temporal measurements uses a fitness function that takes the ratio of the specific behaviors metric value to the maximum value of that metric type. If smaller values are more desirable, such as in spatial distance, then the whole ratio is subtracted from 1 (Equation 6).

The contributions for metric Bi ($CTAL(m_{Bi})$) is calculated as follows:

$$\mathrm{CTAL}(m_{Bi}) = 1 - \frac{|m_{Bi}|}{Max|m_{Bi}|} \tag{6}$$

where $|m_{Bi}|$ is the value of metrics for behavior Bi and is calculated by the behavior or taken from the environment. $Max|m_{Bi}|$ is the maximum over all the values for metric m_{Bi} submitted by the behaviors to the mutex and is calculated by the mutex itself. To compute the contribution to a behavior's activation level for temporal constraints, the time constraint provided by the user is used. Based on mapping and localization data ABBRA can estimate how long it should take the robot to reach a goal. The temporal fitness function is calculated as follows (Equation 7):

$$CTAL(m_{b_t}) = e^{\frac{-1*(TC_B - TSys - TLeft_B)}{LatestTC - EarliestST}} \quad (7)$$

where TC_B is the time constraint for current behavior (deadline by which the behavior should end), *TSys* is the current time of the system clock, $TLeft_B$ is the estimated time it will take behavior B to reach the goal based on true distance, *LatestTC* is the longest existing time constraint and *EarliestST* is the earliest starting time for any unfinished behavior that has a time constraint.

The reason two methods were necessary is that the effect of time on the activation level is not linear. Experimentally it was observed that the desired influence of time on the activation level should be minimal until roughly 20% of the remaining time is left, after which the time constraint should begin influencing the activation level at a much more substantial rate. For example, if the robot has two behaviors, where one takes five minutes to complete and the other takes ten minutes, and both must be finished in an hour, the time constraint should have almost no influence as to which goal wins the auction. However, if the behavior that is finished in 5 minutes has a time constraint of ten minutes, then it should be substantially more important than the behavior with an hour time constraint. Because of this, a non-linear exponential decay curve was used to model the influence the time constraint had on the activation level.

4 Experimental Results

This section presents the experimental validation for the three stages of development of ABBRA. The first stage tested the architecture's basic capabilities: (1) allowing multiple behaviors to function in unison, (2) resolving conflicting behaviors with conflicting goals and a (3) basic mechanism to allow a time constraint to influence the outcome. The testing on this stage was performed in the Player/Stage simulation environment [47]. The second stage incorporated the addition of new goals during runtime and allowed every goal to have temporal influence. The testing for this stage was also performed in Player/Stage. However, these tests were composed of multiple goals, whose objective's locations were known and unknown, with varying time constraints making for more complex scenarios than in the previous phase. The final stage of testing incorporated mapping functionality into ABBRA. This allowed the behaviors to know the true distance to any known location, which also allowed for the estimation of the task's completion time. The third stage of testing used a real time clock instead of counting the number of auction cycles as a means to enforce time constraints. The third phase was

integrated with the Robotic Operating System (ROS) and the tests were performed on the Segway Robotic Mobile Platform (RMP).

4.1 Validation of Phase 1

The first phase of validation used the Stage robotic simulator to simulate a robot with three actuators: a pivoting camera, a 2D gripper and the robot's differential drive. Although in tests, the gripper was never used, both the pivot camera and the robot were used in unison. The robot is equipped with a behavior set that allows it to reach locations indicated by known (x,y) position information and different colored markers (blue, red) and to center or align the robot on a colored marker (green). The robot also has a safe avoid that kicks in when an obstacle is detected, steering the robot around the obstacle. At this point in time mapping was not used. The center camera behavior is represented as an abstract behavior as described in Section 3.2 with two sub-behaviors: one that centers the robot on a green object, and one that centers the camera.

In phase one of the architecture, three experimental evaluations were performed to demonstrate the feasibility of using the auction mechanism for action selection. We also ran preliminary tests to see if goal oscillation, degradation or emergent behavior would occur. Because of the imperfect control of the robot, it would always successfully converge onto a goal.

Test 1.1: in this experiment the robot was requested to visit a red and blue goal with known locations. There were no other constraints or goals so the robot visited the blue goal first because it was closer (Figure 2). Since the robot had no knowledge about the environment, it had to wander around the obstacle and use a best guess exploration to reach the goal. After reaching the blue goal, the robot then moves towards the red goal.

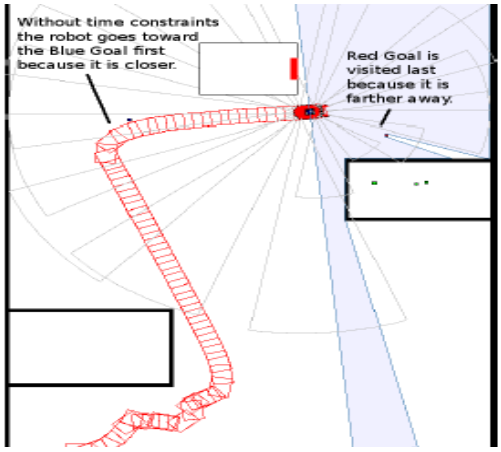

Figure 1: Robot trajectory during ***Test 1.1***

Figure 3, details the graph of the winning behaviors versus time. Although a small oscillation occurred between the goals at time 3, the robot does converge onto finishing the blue goal first, and then the red goal.

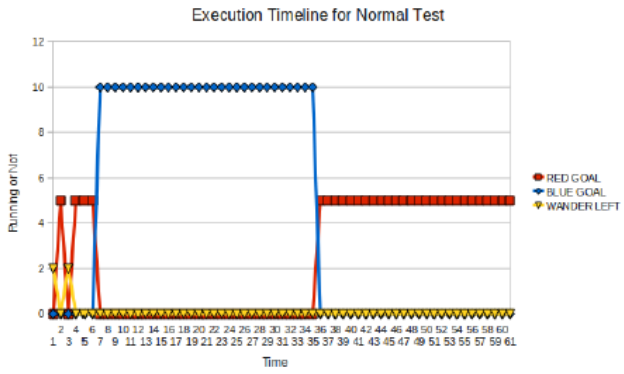

Figure 2: The winners of the auction over time for *Test 1.1*

Test 1.2: This experiment demonstrates the ability of the auction system to handle requests with time constraints. Similar to the last test the robot had to visit two goals, red and blue, with the difference that this time the red goal has a critical time constraint (i.e. it needs to be completed by a certain deadline). Given that the robot does not have a map of the environment in this phase, it cannot estimate the time it would take it to reach the red goal. At this phase of the research, the robot based the time constraint on the Euclidean distance, this changed in the third phase when mapping was introduced. During the experiment the robot heads directly to the red goal, avoiding the wall on the right, visits the red goal and then moves toward the blue goal (Figure 4).

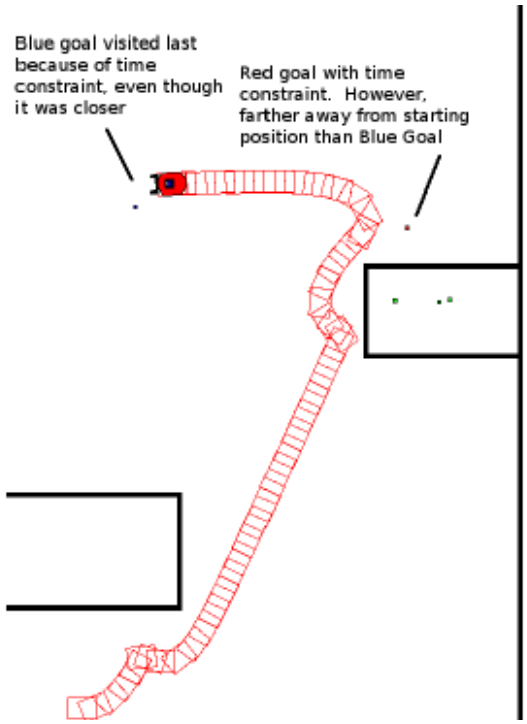

Figure 3: Robot trajectory for ***Test 1.2*** (time constraint on the red goal).

Figure 5 shows the plot of winning behaviors over time. The observed oscillation comes from too much emphasis being placed on spatial metrics because redundant forms of the same spatial metric were used by each behavior. For example, the Euclidean distance was used, as well as the difference in the X and Y axes. This was modified in the second phase of development to correct the problem so the auction mechanism would only consider the distance to goal as a single metric.

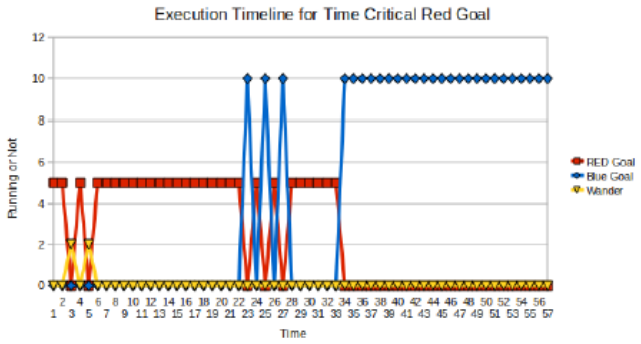

Figure 4: The winners of the auction over time for ***Test 1.2***.

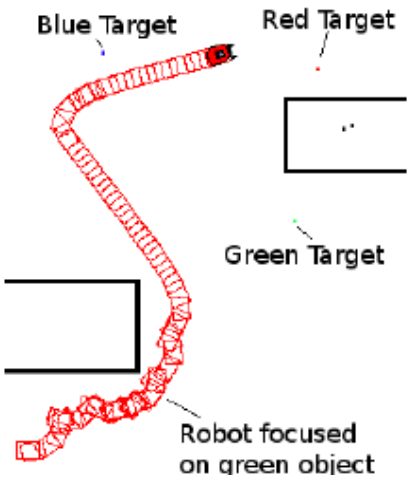

Figure 5: Robot trajectory for ***Test 1.3***.

Test 1.3: This experiment demonstrates the robot's ability to act opportunistically if situations arise in the environment (such as detecting a target whose location was unknown). In this test the robot is required to visit the blue and red targets (which had known coordinates) and then to center itself on a green target, whose location was unknown. While going toward the known location targets, after the robot turns around the corner of the first wall it detects the green object. This is an opportunity for the robot to fulfill its center task while on the way to the other targets: the auction mechanism takes advantage of this dynamically allowing the robot to finish this task before continuing to the blue and red goals (Figure 6). Since the time constraint component was not used in this test therefore, the robot visits the blue goal first (because it is closer) and then the red goal.

The oscillation observed in Figure 7 is due to the wander behavior, which activated whenever the robot came within a certain distance of a wall. At the point where the green object was detected the robot stopped what it was doing and finished that behavior. The rest of the oscillation was due to the over-emphasis placed on the spatial metrics. However, the robot still behaved as expected by finishing the green goal first, then blue, and finally red.

Figure 6: The winners of the auction over time for ***Test 1.3***.

4.2 Validation of Phase 2

The second phase of ABBRA's development focused on adding the following main capabilities: (1) a more robust time constraint system and (2) the ability to handle the addition of new goals dynamically at runtime. Six tests were performed with this phase of research. These test scenarios included the following tasks: visiting of 5 goal targets with known (x,y) locations (Figure 8) and centering on the green target, which will be referred to as green goal. The green goal's location was unknown by the robot.

Test 2.1: This is a basic control test where the robot had to execute all the tasks mentioned above (visit the 5 goal targets and center on the green target). In this test no time constraints were used nor was any goal dynamically added during runtime. As shown in Figure 8, the robot visits the goal with the shortest distance first until it discovers the green goal in which it will dynamically reprioritizes. Then it finishes the remaining goals. The goals were met in the following order: 4, 2, green, 1, 3 and 5.

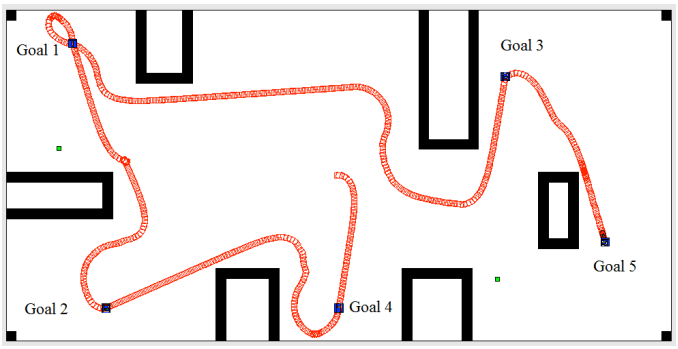

Figure 7: Path taken by control test for ***Test 2.1***.

Figure 9 details the winning behaviors over time. The robot simply follows the shortest goal first until it detects the green object and then proceeds with the rest of them.

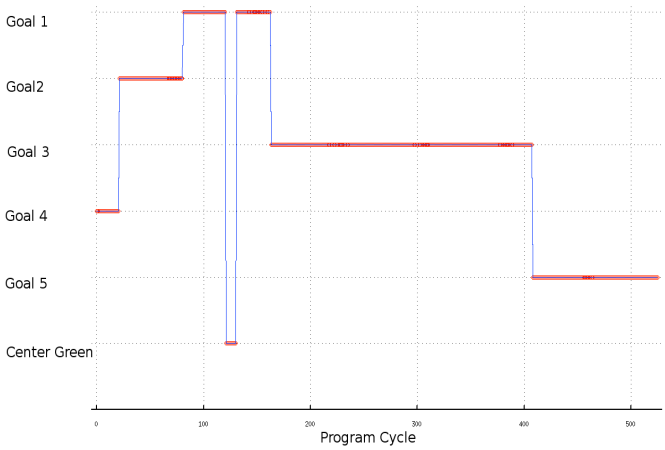

Figure 8: The winners of the auction over time for ***Test 2.1***.

Test 2.2: In the second test two goals were requested with critical time constraints. Both goals 1 and 5 have a critical time requirement remaining before they had to start. The robot would immediately choose the closer of the two, goal 5, and finish it. It would then move directly

to goal 1 and finish it. After it finished goal 1 it then finishes the green goal next to it, therefore, proving that the time constraint outweighed the opportunity presented by the discovery of the green target, which was very close to the robot. The robot thus passes by the green target while moving toward goal 5, meaning that the green goal was not as important as a time critical goal (Figure 10). The order in which the goals were completed is as follows: 5, 1, green, 2, 4, 3.

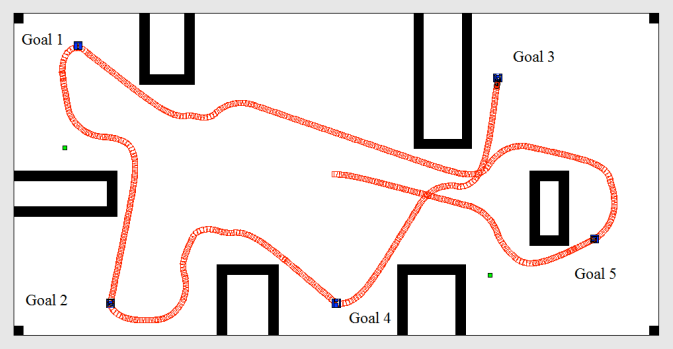

Figure 9: Robot trajectory for ***Test 2.2*** (goals 1 and 5 have a critical time constraint).

Figure 11 shows the winning behaviors over time, indicating that goals 1 and 5 finished first. Notice that the oscillation exhibited from the previous stage is no longer present because in this phase only one form of the distance to goal is used (Euclidean distance)

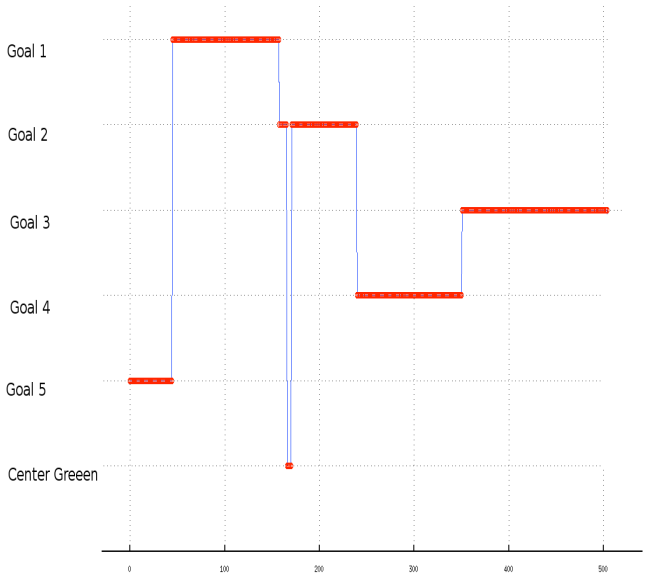

Figure 10: The winners of the auction over time for ***Test 2.2*** (goals 1 and 5 have critical time constraints.)

Test 2.3: This experiment tested the performance of the system in situations when a critical time constraint is given to tasks whose objectives' location in the environment was unknown (center on green target). In this case, instead of going to the goal that had the shortest distance, the robot began wandering around trying to locate the green object (Figure 12). The robot passes by goal 3, however this was purely a coincidence because the robot was wandering looking for a green object. After the robot locates the green object, it resumes going to the goal whose objective is the shortest distance. The goals were finished in the following order: 3 (due to wandering), green, 5, 4, 2, 1.

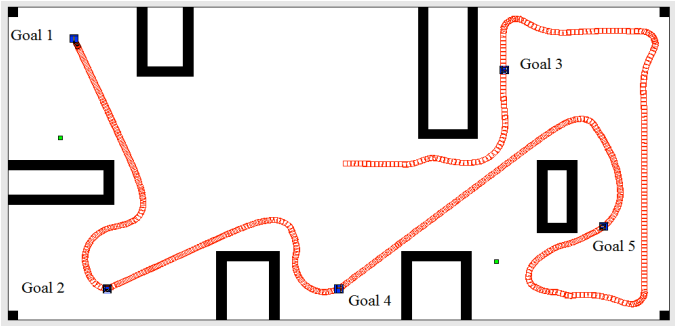

Figure 11: Robot trajectory for ***Test 2.3*** (goal with unknown location has a critical time constraint).

After plotting the winning behaviors versus time it is possible to see that center on green object remained the only active goal to be completed (Figure 13). Goal 3 was completed un-intentionally because the wandering behavior took the robot in the proximity of the goal (the corresponding behavior never won an auction cycle.)

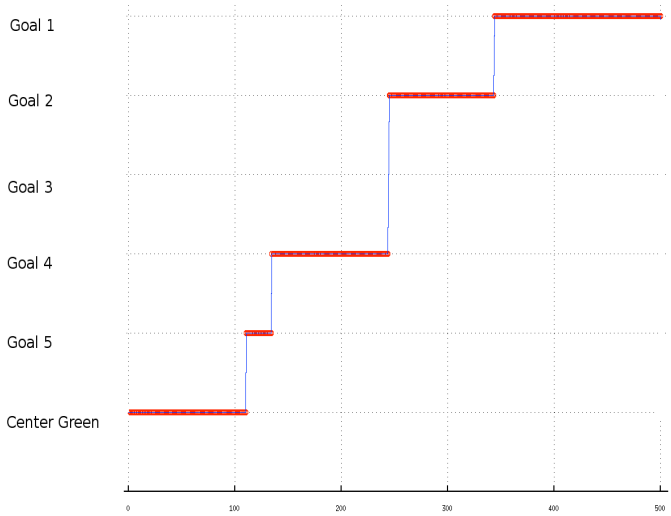

Figure 12: The winners of the auction over time for ***Test 2.3*** (goal with an unknown location has a critical time constraint.)

Test 2.4: This test validates the ability to handle task requests given at run time. In particular, in this scenario the green goal is requested, with a time constraint, right after goal 2 is accomplished. Thus, the new task takes precedence over other goals when it is created. The path of the robot looks very similar to the control run (Figure 14). However, the green goal was not introduced until after goal two finished (Figure 15). The goals were finished in the following order: 4, 2 green, 1, 3, 5.

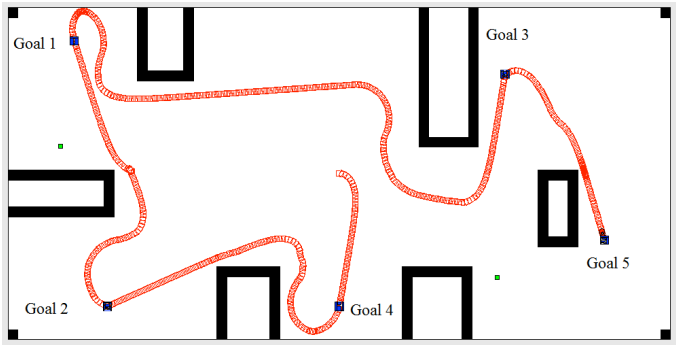

Figure 13: Robot trajectory for ***Test 2.4*** (goal with unknown location is added at runtime with a critical time constraint.)

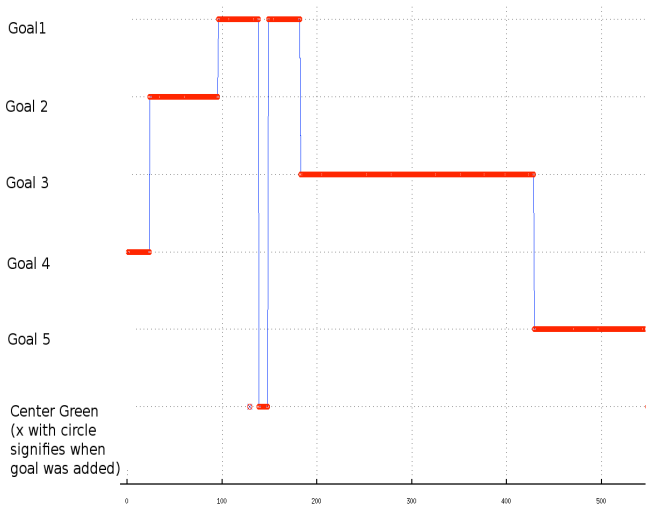

Figure 14: Winning behaviors over time for ***Test 2.4*** (goal with unknown location is added at runtime with a critical time constraint.)

Test 2.5: This experiment is similar to test 2.4, except that a goal with a known location for its objective (goal 5) is dynamically added with a critical time constraint after goal 2 finishes. Once this goal is created, it will take control and move toward goal 5 because of the time constraint. This test demonstrates that it is possible for ABBRA to add a goal whose objective has a known location dynamically during run time (Figure 16). The goals were finished in the following order: 4, 2, 5, 3, 1, green.

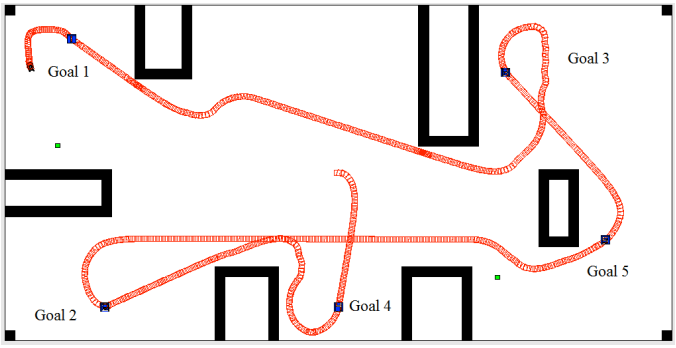

Figure 15: Robot trajectory for ***Test 2.5*** (goal with a known location is added during runtime with a critical time constraint.)

Figure 17 plots the winning behaviors versus time and demonstrates that behavior 5 immediately takes control from behavior 1 when added. This behavior was automatically introduced into the system after goal 2 has been met.

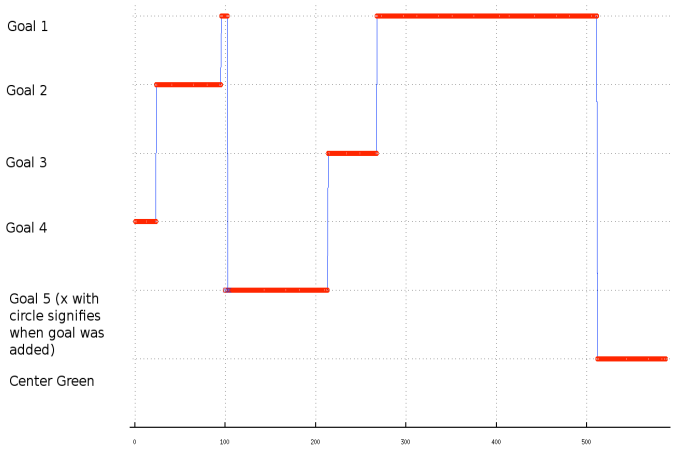

Figure 16: Winning behaviors over time for ***Test 2.5*** (goal with a known location is added during runtime with a critical time constraint.)

Test 2.6: This experiment was similar to test 2.5 (dynamically adding goal 5 after goal 2 was reached), however this time no time constraint was given to goal 5. This in resulted in a run that looked exactly like the control test, where the robot would simply visit the shortest goal first because the architecture had no reason to react when goal 5 was added (Figure 17). The goals were finished in the following order: 4, 2, green, 1, 3, 5.

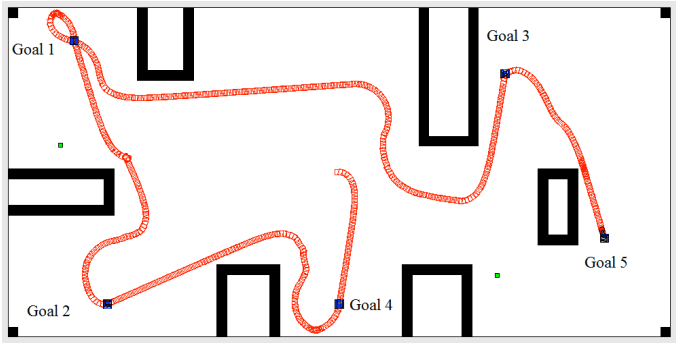

Figure 17: Robot trajectory for ***Test 2.6*** (goal with a known location is added during runtime and has no time constraint.)

Likewise graphing the winning behaviors against time shows the same results even though goal five is added after goal 2 is reached (Figure 19).

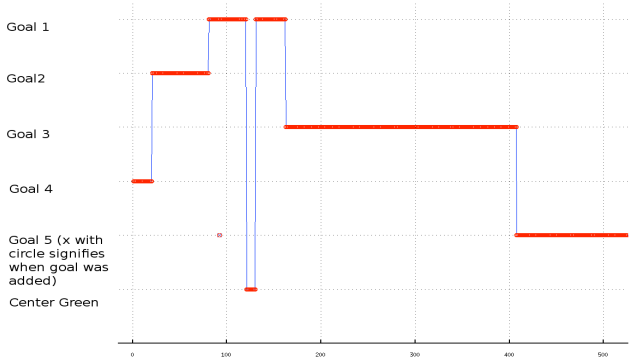

Figure 18: Winning behaviors over time for ***Test 2.6*** (goal with a known location is added during runtime and has no time constraint.)

4.3 Validation of Phase 3

The third phase of development added the following capabilities: 1) modified the timer system to use a real-time clock (this allowed the user to enter time constraints in terms of hours, minutes, seconds) and 2) introduced mapping and localization capabilities (allowing the behaviors to predict whether they could meet certain time constraints.) For this phase ABBRA was validated on a Segway Robotic Mobile Platform (RMP), using Willow Garage's robotic operating system (ROS)[48]. These tests were performed in the halls of the Computer Science and Engineering building at the University of Nevada, Reno.

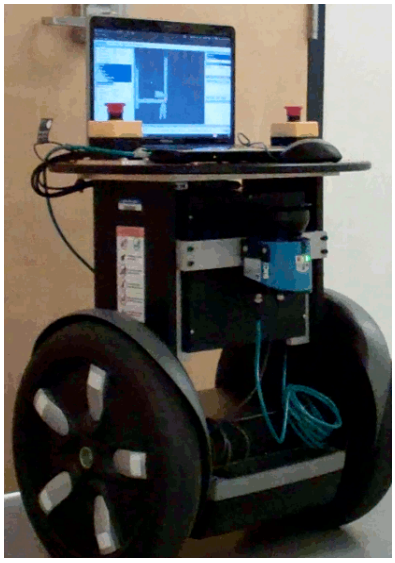

Figure 19: Segway Robotic Mobile Platform (RMP)

The five different testing scenarios for this phase used three goals with known locations, represented by yellow, orange, and red targets. Goal 4, or green goal, was used once again to demonstrate the center on green behavior.

Test 3.1: The first test performed on the third phase was a basic control test, in which the robot had to visit three goals whose objectives had known locations and also find the green object. With no time constraints, the robot would simply visit the goal with the shortest distance first and only interrupting the active goal to take advantage of discovering the green goal (Figure 21). Likewise, the graph showing the winning behaviors over time demonstrate a clear convergence onto the desired goals (Figure 22). The goals were finished in the following order: 1, green, 2, 3.

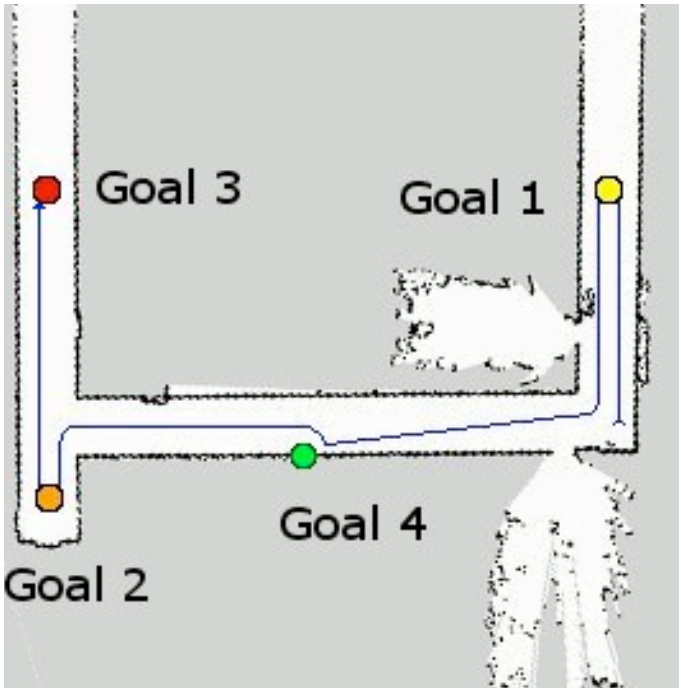

Figure 20: Physical robot trajectory for Test 3.1.

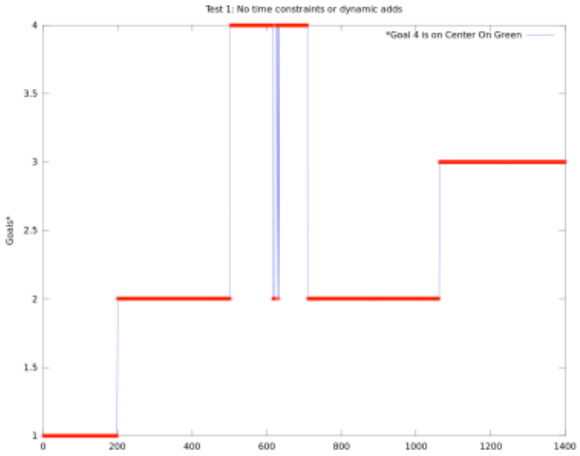

Figure 21: Winning behaviors over time for ***Test 3.1***.

Test 3.2: This experiment tested a scenario in which goal 2, which has a time constraint, could not be accomplished in time. The architecture estimates the amount of time necessary to reach the goal based on the distance from the map and determined that it could not meet the temporal constraint. In this scenario the robot simply drops the goal and considers it missed. This is to demonstrate a different approach as compared to the previous phases of the architecture, where if a goal missed its time constraint it would get higher priority. Both approaches are thus feasible to be used, the choice depending only on the preferences of designer for a particular domain. In this experiment the robot visits the closest goals first, taking time to finish the green goal but without going to goal 2 (Figure 23). Figure 24 shows an oscillation when the robot is going toward the green goal, which is due to noise on the color blob tracker, but the architecture gracefully handles the noise in the sensory data, accomplishing the requested tasks. The goals were finished in the following order: 1, green, 3.

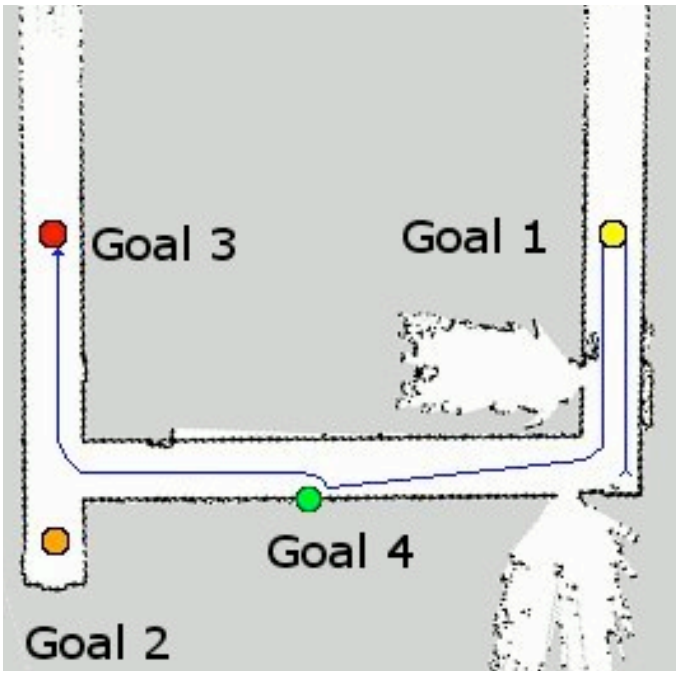

Figure 22: Robot trajectory for ***Test 3.2*** (goal 2 cannot meet its time constraint.)

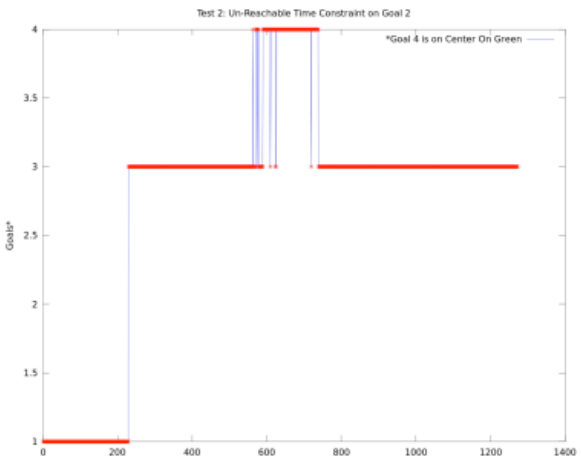

Figure 23: Winning behaviors over time for ***Test 3.2*** (goal 2 cannot meet its time constraint.)

Test 3.3: This experiment demonstrates that the architecture can handle scenarios where attainable time constraints are given; the goal with the time constraint is completed before any other goal with lesser or no time constraint. In this scenario goal 2 is given a time constraint that can be achieved therefore the robot goes to that goal first. After this goal is finished, it will simply find the closest goal to finish next (Figure 25). Note that the green goal is completed after goal 2 is reached (Figure 26). The goals were finished in the following order: 2, 3, green, 1.

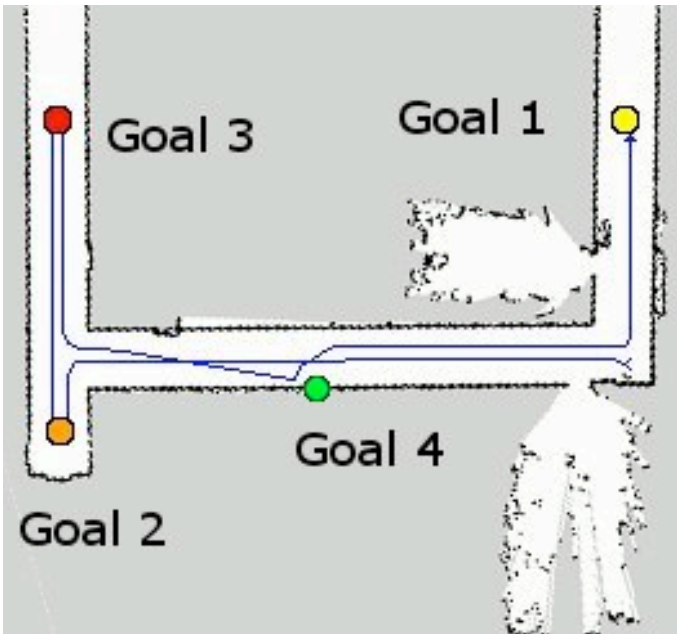

Figure 24: Robot trajectory for ***Test 3.3*** (feasible time constraints.)

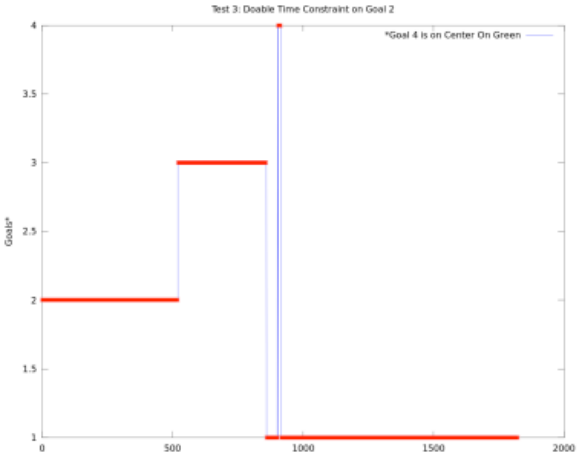

Figure 25: The winning behaviors over time for ***Test 3.3*** (feasible time constraints.)

Test 3.4: This experiments demonstrates that the architecture can handle the addition of new task requests dynamically at runtime. Goal 1 is added after four minutes of the program running. For this test, goal 1 had no time constraints therefore

the robot would continue with the closer goals and deal with goal 1 last (Figure 27). As shown in Figure 28, although the blob tracker caused oscillations, the auction mechanism handled the noise efficiently and enabled the robot to finish its tasks. The goals were finished in the following order: green, 2, 3, 1.

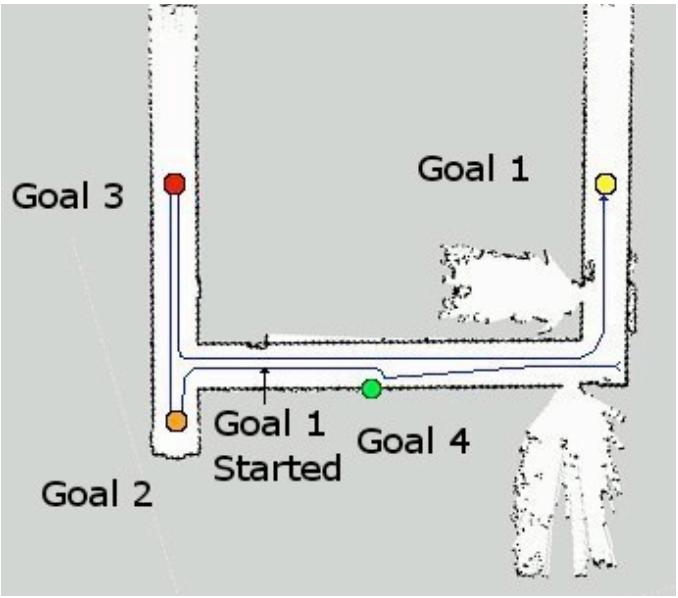

Figure 26: Robot trajectory for ***Test 3.4*** (new task requested at runtime without time constraint.)

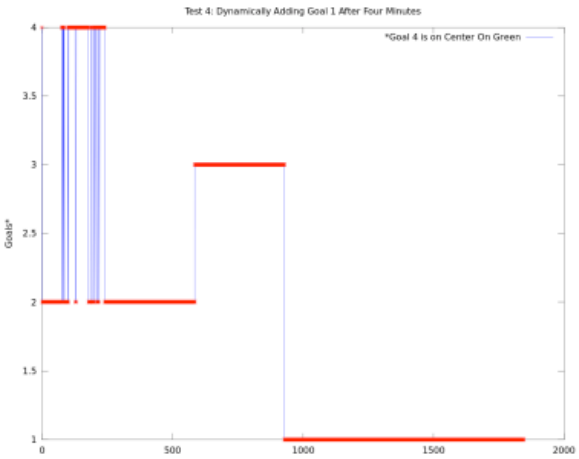

Figure 27: The winning behaviors over time for ***Test 3.4*** (new task requested at runtime without time constraint.)

Test 3.5: This experiment is similar to Test 3.4, with the difference that the newly added task had a time constraint. Goal 1 is dynamically added at the four-minute marker with a critical time constraint. The new task is requested shortly after the robot has finished the green goal. Due to the critical time constraint the robot switches to working on the new goal (Figure 29). The graph of winning behaviors shows this as goal 1 interrupts goal 2 when it is added (Figure 30). The goals were finished in the following order: green, 1, 2, 3.

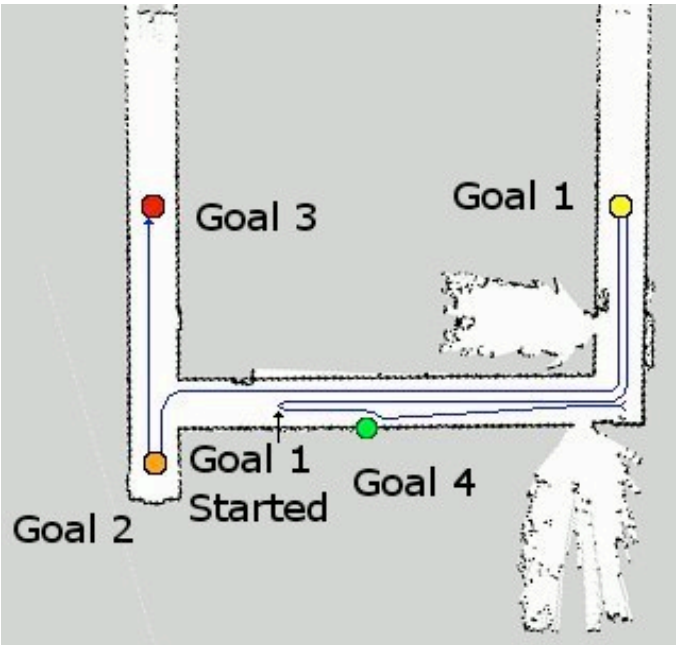

Figure 28: Robot trajectory for ***Test 3.5*** (new task requested at runtime with time constraint.)

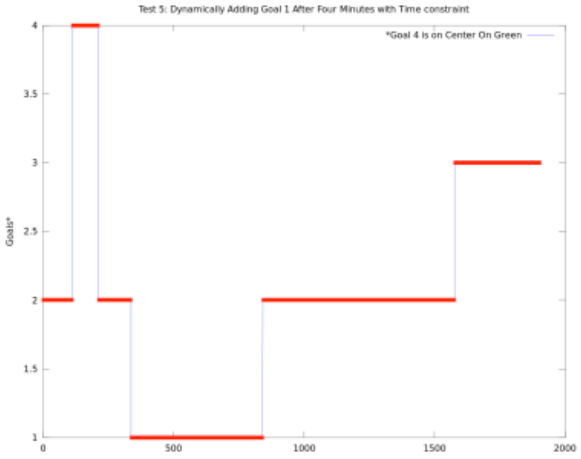

Figure 29: Winning behaviors over time for ***Test 3.5*** (new task requested at runtime with time constraint.)

5 Discussion and Future Work

ABBRA has demonstrated that it can handle many dynamic constraints in the real world environment. However, the architecture has a few drawbacks. For example, if a user would like that the robot determines what it should do on its own (in the absence of specific user requests), a different module would have to be implemented to handle this situation. Another, potential disadvantage is that in its current implementation ABBRA does not support learning. The robot can only perform tasks that it already knows, but it should be possible to incorporate learning approaches that allow the robot to acquire new skills. Furthermore, ABBRA is not a cognitive architecture – while a robot can make decisions on what tasks to perform next in order to be efficient using ABBRA, additional extensions to the architecture would be needed in order to handle symbolic representations and reasoning.

The next phase of research consists of integrating capabilities for Human Robotic Interaction (HRI) into ABBRA, thus adding another degree of complexity to the action selection mechanism. This research will consist of two phases. The first phase will develop capabilities that allow a human user to make task requests to the robot in an efficient and easy to

use way. Once challenge that needs to be address in this context is handling the situations in which the robot may not have time to interact with the human due to existing deadlines. Another requirement of this phase is to create an interface that is modular and can be ported to different hardware. Currently, mobile devices are becoming more wide-spread along with laptops and Tablet PCs, which makes these devices a more ideal HRI interface. This phase of the tests will focus on human interaction with the robot directly, interacting through a PC and interacting through a mobile device.

The second phase of this research is expanding the HRI to multiple robotics. This area research will focus on the ease of controlling multiple robots with the mechanisms as developed in the previous phase. Once the interface is developed the last requirement would be creating a collaborative control system that allows humans and robots to work together [49-51].

6 Conclusion

This paper presents a control architecture (ABBRA) that allows a robot to handle multiple types of constraints and determines what order the robot should pursue its assigned tasks. This is vital for service robot so they can function in dynamic environments and provide useful services to human users. This paper experimentally shows that the auction mechanism provide robust action selection and allow the robot to consider multiple constraints of different types. This paper also demonstrates that the auction method resolves tasks that have conflicting goals. ABBRA adheres to time constraints and determines whether it is possible to achieve a task in time. This architecture is also capable of handling tasks added dynamically during runtime without having to stop and re-plan or degrade the quality of the robot's performance. This paper details the work done on ABBRA by presenting the results from the three different stages of testing, which include results from both simulation and in real dynamic environments. This paper presents a formal description of the scheduling problem solved by ABBRA and gives a detailed description of the solution. The results support the use of auctions as a form of arbitration in real-world environments, yielding efficient and robust results that can handle many dynamic conditions that are challenging in real-world domains.

7 Reference

[1] Arkin, and R. C., *Behavior-Based Robotics*, 1 ed., p.^pp. 1-491, Massachusetts: Massachusetts Institute of Technology, 1998.

[2] J. K. Rosenblatt, "DAMN: A distributed architecture for mobile navigation," *Journal of Experimental & Theoretical Artificial Intelligence,* vol. 9, no. 2-3, pp. 339-360, 1997.

[3] J. Riekki, and J. Roning, "Reactive task execution by combining action maps." pp. 224-230 vol. 1.

[4] J. Hoff, and G. Bekey, "An architecture for behaviour coordination learning." pp. 2375-2380.

[5] A. Saffiotti, "The uses of fuzzy logic in autonomous robot navigation," *Soft Computing-A Fusion of Foundations, Methodologies and Applications,* vol. 1, no. 4, pp. 180-197, 1997.

[6] A. Saffiotti, K. Konolige, and E. H. Ruspini, "A multivalued logic approach to integrating planning and control," *Artificial intelligence,* vol. 76, no. 1-2, pp. 481-526, 1995.

[7] J. Yen, and N. Pfluger, "A fuzzy logic based extension to Payton and Rosenblatt's command fusion method for mobile robot navigation," *Systems, Man and Cybernetics, IEEE Transactions on,* vol. 25, no. 6, pp. 971-978, 1995.

[8] R. C. Arkin, "Motor schema-based mobile robot navigation." pp. 264-271.

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

[10] 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.

[11] 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.

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

[13] 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.

[14] P. Haazebroek, S. van Dantzig, and B. Hommel, "A computational model of perception and action for cognitive robotics," *Cognitive processing,* vol. 12, no. 4, pp. 355-365, 2011.

[15] G. H. Lim, and I. H. Suh, "Improvisational goal-oriented action recommendation under Incomplete Knowledge Base." pp. 896-903.

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

[17] 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,* vol. 1860, pp. 95-106, 2000.

[18] 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.

[19] 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.

[20] 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.

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

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] W. Sheng *et al.*, "Distributed multi-robot coordination in area exploration," *Robotics and Autonomous Systems,* vol. 54, no. 12, pp. 945-955, 2006.

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

[29] M. K. Sahota, "Action selection for robots in dynamic environments through inter-behaviour bidding," *From animals to animats,* vol. 3, pp. 138-142, 1994.

[30] A. Almeida, and L. Figueiredo, "A product oriented approach to Dynamic Scheduling." pp. 523-528.

[31] F. T. S. Chan, T. Wong, and L. Chan, "Lot splitting under different job shop conditions." pp. 4722-4728.

[32] J. B. Wang, and M. Z. Wang, "Worst-case behavior of simple sequencing rules in flow shop scheduling with general position-dependent learning effects," *Annals of Operations Research*, pp. 1-15, 2011.

[33] M. Younas *et al.*, "Priority scheduling service for E-commerce web servers," *Information Systems and E-Business Management,* vol. 6, no. 1, pp. 69-82, 2008.

[34] H. Liu, A. Abraham, and Z. Wang, "A multi-swarm approach to multi-objective flexible job-shop scheduling problems," *Fundamenta Informaticae,* vol. 95, no. 4, pp. 465-489, 2009.

[35] D. Lei, "Solving fuzzy job shop scheduling problems using random key genetic algorithm," *The International Journal of Advanced Manufacturing Technology,* vol. 49, no. 1, pp. 253-262, 2010.

[36] J.-Q. Li, Q.-K. Pan, and K.-Z. Gao, "Pareto-based discrete artificial bee colony algorithm for multi-objective flexible job shop scheduling problems," *Int. J Adv Manuf Technol,* vol. 55 pp. 10, 2011.

[37] X. Miao, P. B. Luh, and D. L. Kleinman, "Dynamic job scheduling with strict deadline." pp. 116-121 vol. 1.

[38] C. Anandaraman, "An improved sheep flock heredity algorithm for job shop scheduling and flow shop scheduling problems," *International Journal of Industrial Engineering,* vol. 2, pp. 749–764, 2011.

[39] Y. Hu, M. Yin, and X. Li, "A novel objective function for job-shop scheduling problem with fuzzy processing time and fuzzy due date using differential evolution algorithm," *The International Journal of Advanced Manufacturing Technology,* vol. 56, pp. 1-14, 2011.

[40] A. Kouider, and B. Bouzouia, “Multi-agent job shop scheduling system based on co-operative approach of idle time minimisation,” *International Journal of Production Research,* vol. 50, no. 2, pp. 409-424, 2011.

[41] J. P. Watson *et al.*, "Toward a descriptive model of local search cost in job-shop scheduling."

[42] A. Yahyaoui, and F. Fnaiech, "Recent trends in intelligent job shop scheduling." pp. 191-195.

[43] 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.

[44] B. A. Towle Jr, and M. Nicolescu, “Applying dynamic conditions to an auction behavior-based robotic architecture,” *Int'l Conf. Artificial Intelligence (ICAI'11),* vol. Volume 1, no. July 18-21, 2011, pp. 6, 2011.

[45] B. Towle, and M. Nicolescu, “Real-world implementation of an Auction Behavior-Based Robotic Architecture (ABBRA),” in IEEE International Conference on Technologies for Practical Robot Applications (TePRA), Woburn, MA, 2012, pp. 79-85.

[46] M. N. Nicolescu, and M. J. Mataric, "A hierarchical architecture for behavior-based robots." pp. 227-233.

[47] P. Stage. "Player," 15 Oct 2012, 2012; http://playerstage.sourceforge.net/.

[48] W. Garage. "ROS | Willow Garage," 7 Mar 2012, 2012; http://www.willowgarage.com/pages/software/ros-platform.

[49] T. Fong *et al.*, “A personal user interface for collaborative human-robot exploration,” in 6th International Symposium on Artificial Intelligence, Robotics, and Automation in Space (iSAIRAS), Montreal, Canada, 2001, pp. 23.

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

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