Reactive Planning in a Motivated Behavioral Architecture

Éric Beaudry, Yannick Brosseau, Carle Côté, Clément Raïevsky,
Dominic Létourneau, Froduald Kabanza, François Michaud
Université de Sherbrooke
Sherbrooke (Québec) CANADA J1K 2R1
{laborius-challenge}@listes.USherbrooke.ca

Abstract

To operate in natural environmental settings, autonomous mobile robots need more than just the ability to navigate in the world, react to perceived situations or follow pre-determined strategies: they must be able to plan and to adapt those plans according to the robot's capabilities and the situations encountered. Navigation, simultaneous localization and mapping, perception, motivations, planning, etc., are capabilities that contribute to the decision-making processes of an autonomous robot. How can they be integrated while preserving their underlying principles, and not make the planner or other capabilities a central element on which everything else relies on? In this paper, we address this question with an architectural methodology that uses a planner along with other independent motivational sources to influence the selection of behavior-producing modules. Influences of the planner over other motivational sources are demonstrated in the context of the AAAI Challenge.

Introduction

The AAAI Mobile Robot Challenge (or simply the AAAI Challenge) is a rich setup for working toward human-like intelligence of autonomous mobile robots operating in real life settings. Introduced in 1999, it consists of having a robot enter the conference site, find the registration desk, register, perform volunteer duties and give a presentation (Maxwell *et al.* 2004). These specifications imply that robots must be able of autonomous navigation in the world, planning ahead a strategy for getting through the different steps of the AAAI Challenge (such as registering, going to the presentation room, and making a presentation) and adapt to the occurrence of unanticipated situations (for instance when interacting with people). Obstacle avoidance, navigation, localization, mapping, planning, modeling, recognition, searching, tracking, interaction, cooperation, decision-making, just to name a few, are good examples of recently developed capabilities for making a mobile robot act as an intelligent and useful agent in the real world. Such capabilities go from purely reactive approaches to deliberative reasoning over knowledge abstractions. Assuming that both types of approaches are required in designing autonomous machines for real life settings, a lot still remain to be understood about the nature of the boundary between deliberation and reaction (Arkin 1998). Acquiring a better understanding of how to integrate decision-making capabilities is a necessary step in designing mobile robots that can manifest high-level intelligence in real life settings.

Research on hybrid deliberative/reactive architectures for mobile robots is oriented toward solving this question, with approaches based on hierarchical integration of planning and reaction, planning to guide reaction, or coupling planning and reacting (Arkin 1998). In most of them, planning always remain a central element in the decision-making processes. The decisions about which features are to be handled using an automated planning system, and which ones are to be managed by other decision-making modules, are ultimately a matter of design choices that depend on the planner's capabilities. We may choose to ignore uncertainty during the planning phase by generating deterministic plans that are sequences of actions, and monitoring failures to re-invoke the planner whenever necessary. This is the strategy used with Xavier (Haigh & Veloso 1998). It is also conceivable not to involve any automated planning at all and still manage to make the robot accomplish complex tasks, as did many teams participating in previous AAAI Challenges (Michaud *et al.* 2001; Maxwell *et al.* 2004). Therefore, other mechanisms may be useful to complement the planner's capabilities in the robot's decision-making processes.

To do so, the architectural methodology we are developing is based on the notion of motivated selection of behavior-producing modules. The general idea is to have different motivational sources such as perceptual influences, pre-determined scenarios, navigation algorithms, a planner and alike, share knowledge about how to activate and use behavior-producing modules (which can be typical behavior-based reactive controllers with or without internal states, goal-oriented behaviors or other types of behavior).

Using different motivational sources allows efficient use of various influences regarding the tasks the robot must accomplish, while not having to rely on only one of them for the robot to work.

The paper is organized as follows. After presenting related work, we present our hybrid architectural methodology and its underlying reactive planner. This is followed with a presentation and analysis of trials done in simulation and with a mobile robot, using the AAAI Challenge as the experimental setup, and with perspectives for future work.

Related Work

Most architectures that integrate task planning and robot execution interleave these two processes. In particular, ROGUE is an architecture integrating a task planner (PRODIGY) with a Xavier's path planner and navigation capabilities (Haigh & Veloso 1998). PRODIGY takes as input an initial state and a goal state condition, and produces a plan that is a sequence of actions that if executed, and if things happen as planned for, the robot reaches a new state satisfying the goal condition. As planning is interleaved with execution, PRODIGY can update its current planning state and orient its backtracking to respond to changes in the environment, which increases the likelihood of overall plan success. However, since the robot always pick the next action to execute as the next action in the plan generated by PRODIGY, whether complete or still under construction, it is clear that ROGUE inherits all the limitations of PRODIGY. Many other approaches follow this model of interleaving planning and execution, but differ in the underlying robot platforms, in additional strategies for plan monitoring and recovery, and in their capabilities such as handling metric time constraints (e.g., see (Chien *et al.* 2000; Lemai & Ingrand 2004; Ambros-Ingerson & Steel 1998)).

Many other robot architectures provide an Executive interface between high-level deliberative processes (including planning) and lower-level ones. For instance, Task Description Language (TDL) (Simmons & Apfelbaum 1998) is a language for describing robot behaviors as task trees, such as inner nodes correspond to abstract tasks or goals and leaf nodes correspond to commands directly executable by the robot. A planner synthesize such task-tree behaviors, thereby somehow determining the action executed by the robot (i.e., commands at the leaf nodes of the task tree). Fundamentally this is as in the other examples above (e.g., Xavier and PRODIGY), the difference with TDL being to provide a rich language for specifying tasks.

In our case, we chose to follow the 'planning to guide reaction' (PGR) model, in which the planner influences the configuration of behavior-producing modules rather than being responsible for completely determining the actions of the robot. For TDL to reassemble the PGR approach we are interested in, the command on the leaf of a task tree that would be computed by a planner, would have to compete with commands from other behaviors potentially aimed at the same goal. In other words, we do not want the planner to synthesize behavior but to only be a source of influence.

Propice-Plan is one illustration of PGR based on the Procedural Reasoning System (PRS) (Ingrand & Despouys 2001). Robot behaviors are implemented by rule-based templates, called operators, attached with robot executable procedures. Precondition and effect specifications are also attached to these templates, so that when the robot is executing the procedure, Propice-plan simulates its control engine ahead of real-time using, not the executable procedures, but the more abstract action and precondition specifications. Traces of this simulation are then exploited by execution engine to anticipate failures. Our hybrid architectural methodology extends the PGR model by adding influences other than the ones from a planner (like pre-determined strategies drive by perceptual conditions, temporally-influenced activation variables acting as motives, etc.) to guide the selection and configuration of behavior-producing modules. Compared to Propice-Plan, our architecture is more process-oriented (a feature inherited from EMIB (Michaud 2002)), whereas Propice-Plan is more production-rule oriented (a feature inherited from PRS). Therefore, the planner is no longer a central component of the architecture. Limiting the scope of the planner's influence makes it possible to use more suitable mechanisms to handle unpredictable events, reuse planning algorithms developed in previous work (just like the use of PRODIGY in ROGUE (Haigh & Veloso 1997)), and facilitates robot reconfiguration to new capabilities and applications.

Motivated Behavioral Architectural (MBA) Methodology

Figure 1 represents the Motivated Behavioral Architecture (MBA). Behavior-producing modules (*BPM*, or behaviors) constitute the basic MBA's components that control the robot's actuators. The actual use of a behavior is determined based on its operation conditions and the arbitration scheme. Arbitration might be priority-based, fusion (e.g., motor-schemas), action selection or defuzzification, depending on the implementation. Operation conditions come from percepts, internal states and activations values given by the *Selection* module. These activation values (*BPM.Activation* reflect the robot's intentions derived from interactions between motivational sources.

In order to keep motivational sources as generic and independent from each other as possible, their interactions occur through a task-oriented data structure shared in the *Dynamic Task Workspace* (DTW). Tasks are organized in a hierarchy using a tree-like structure, from high-level/abstract tasks (e.g., deliver message), to primitive/BPM-related tasks (e.g., avoid obstacles). Motivational sources can add or modify tasks in the DTW (through requests *m*), request information about them (*q*) or subscribe to events (*e*) regarding their status. They also issue behavior recommendations (*rec*) concerning tasks: these recommendations can either be positive, negative or neutral, regarding the desirability of allowing the robot to accomplish specific tasks. The *Selection* module determines which tasks to accomplish by selecting those that are more desirable than undesirable. The mapping between tasks and BPM as for the communication of parameters (*p*) and behavior results (*res*) are managed by the *System Know How* (SNOW) module. *BPM.Exploitation*

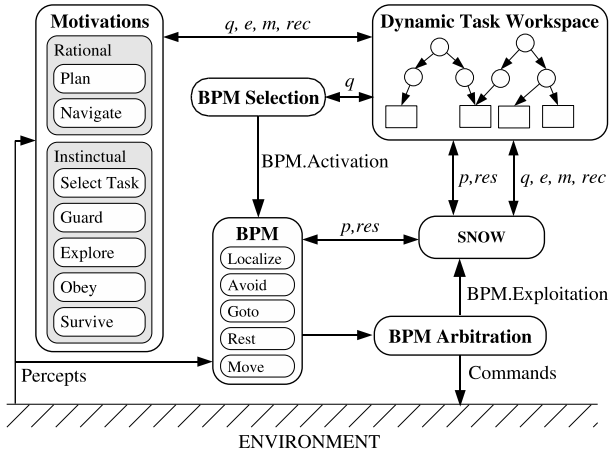

Figure 1: MBA architectural methodology.

represents the effective use of BPM, an important information about the robot's interactions within the environment. A BPM that is activated may or may not be used to control the robot, depending on the sensory conditions it monitors and the arbitration mechanism used to coordinate the robot's behaviors. So, an activated behavior is exploited only when it provides commands that actually control the robot.

Motivational sources are the drives that prompts the robot to act in a certain way. They are influenced by percepts, results and states of pursued goals, and from monitoring the robot's tasks. *Motivations* in this work are categorized as either instinctual or rational. Instinctual motivations provide basic operation of the robot using simple strategies driven by perceptual and internal states influences. Rational motivations involve abstract reasoning over knowledge acquire or innate about the world. We consider rational motivations here to have a greater priority in case of conflicts with instinctual ones.

All MBA's modules run concurrently to derive goals and expressing behavior recommendations. For the work reported in this paper, the architecture uses six BPM and seven motivations. The BPM are the following: Moves, generating a constant forward velocity; Rest, stopping the robot; Goto, allowing the robot to move to a specific location; Avoid, making the robot move safely in the environment; Localize, determining the robot position on a given map according to laser and odometry data. This last BPM is a module that does not give direct commands to the robot's actuators. It provides information to other BPM or motivational modules through the SNOW and DTW, only when a motivational source consider this capability appropriate (e.g., when sufficient processing power is available). Subsumption is used for BPM arbitration. For instinctual motivations, Survive urges the robot to maintain its physical integrity by recommending the obstacle avoidance task. Obey is a process allowing to directly take user's requests for tasks. Explore motivates the robot to discover its environment. Rest makes the robot stay in position and guard an area. Select Task selects one high-level task when none has yet been prioritized: for instance, for tasks that require to go to a specific location, this motivation selects the task for which the robot is physically closest to. For rational motivations, two modules are used to provide motivations that are more related to cognitive processes. Navigate determines a path to go to a specific location, as required for tasks in the DTW. Plan is where a planner can influence the decisions of the robot. In MBA, the role of the planner is to provide the robot with the capability of determining which primitive tasks, and which sequenced order of them, are needed to accomplish high-level tasks under temporal constraints and limited capabilities (as defined by the set of BPM). The following section provides more detailed explanations regarding the planning capabilities in MBA.

Reactive Planning in MBA

Planning problems in relation to the AAAI Challenge involve metric time constraints (e.g., the robot has to be on time for its presentation), resource constraints (e.g., complex robot processes such as navigation, map registration and planning consume processing power), safety goals generated reactively at unpredicted times (e.g., charging the battery whenever it becomes low), preferences among goals (e.g., it is more critical to charge the battery than doing anything else, or presenting on time has priority over helping other attendees) and uncertainty in plan execution (e.g., the time it takes to navigate from one point to another depends on the accuracy of the robot's navigation capability).

To handle this kind of planning problem, we first look for existing planners that support domain definition with metrics and temporal actions. We focused on planners that support PDDL 2.2 with timed initial literal. Since we actually use a planner like a black box, our MBA architecture supports several planners. These planners are SAPA (Do & Kambhampati 2003), LPG (Gerevini, Saetti, & Serina 2003) and TLPlan (Bacchus & Kabanza 2000). We can also use a custom planner we developed, named ConfPlan (Beaudry, Kabanza, & Michaud 2005), based on an HTN (Hierarchical Task Network) approach. For now, planning reactivity is limited to launch the planner when a new task occur, when a plan fails, or when the operation conditions change. Plan failure is detected when a recommended primitive task fails or when a primitive task progresses slower than anticipated, causing a violation of temporal constraints or task achievability.

Reactive Planning Loop

The procedure for a comprehensive description of our reactive planner is described in Algorithm 1. The planner maintains a current plan (P), a current set of high-level tasks (or goals G), the current action (primitive task) of the plan (a), the current environment features relevant to plan (S), and an auxiliary variable e for incoming events from other MBA modules. For instance, when a new goal arrive (line 6) or is cancelled (line 8), the goals list is updated and the planner compute a new plan from this. From lines 17 to 22, replanning is done when the currently sensed information invalidates the plan. The function ISPLANINVALID checks

the validity of the current plan by projecting its action from the current sensed state. Similarly, at ActionTimedout (line 23), when an action takes too much time (i.e., navigation is slower due to the presence of an obstacle that forces to take a longer path), a new plan is needed. If the planner fails to generate a new one, it means that one or more goals are now unreachable (i.e., presentation is not yet possible due to time constraints). In this case, the current high-level task G is cancelled.

Algorithm 1 Reactive Planning Procedure Sketch

```
1.  Sketch Algorithm MBAPLANNER()
2.    Variables Plan P, GoalList G,
.       State S, Action a, Event e
3.    while (true)
4.      e= WaitNextEvent()
5.      Case (e)
6.        NewGoal(g) :
7.          P = CallPlanner(S, G = G + g);
8.        CancelledGoal(g) :
9.          P = CallPlanner(S, G = G - g);
10.       ActionCompleted(a) :
11.         P = P - a; send P.nextAction();
12.       ActionFailed(a) :
13.         P = CallPlanner(S, G);
14.         if (plan=FAILURE)
15.         ga = GoalAssociatedTo(a);
16.         P = CallPlanner(S, G = G - ga);
17.       SensedData(d) :
18.         newS = S.update(d);
19.         if (newS ≠ S)
20.           S = NewS;
21.           if (isPlanInvalid())
22.             P = CallPlanner(S, G);
23.       ActionTimedOut(a):
24.         P = CallPlanner(S, G);
25.         if (P=FAILURE)
26.           g = GoalAssociatedTo(a)
27.             P = CallPlanner(S, G = G - g);
```

Experimental Setup and Results

We experimented the above architecture in simulation using Player/Stage (Vaughan, Gerkey, & Howard 2003) (on a Pentium IV 3.2 GHz computer) and on a wheeled robot platform (also using Player for the low-level interface to the laser range finder). The robotic platform is equipped with a laser range finder and one laptop computer (Pentium M 2.0 GHz). High-level programming is done using RobotFlow and MARIE (Cote *et al.* 2004). RobotFlow/FlowDesigner is a modular data-flow programming environment that facilitates visualization and understanding of what is really happening in the robot's control loops, sensors, actuators. MARIE is a middleware allowing multiple applications, operating on one or multiple machines/OS, to work together in an implementation of mobile robotic nature. This environment proposes a software architecture that avoids making a choice on particular programming tools, and makes it possible to share code and applications. For instance, through MARIE, CARMEN's path planner and localizer (Thrun, Fox, & Burgard 1998) are used in the Navigate motivation and the Localize BPM, respectively. Those programming environments makes it possible to reuse all the software implementation for each setup, except for odometry data and motor commands that needed to be treated differently.

To experiment and validate the MBA architecture, we created a set of scenarios inspired from the AAAI Challenge. These simplified scenarios used nine tasks in the DTW, tasks that are constrained (i.e., they are defined by locations or time parameters) or unconstrained:

- Avoid (A). Unconstrained task submitted by Survive, making the robot navigate safely in its environment by avoiding obstacles.
- Deliver-Message (D). Constrained task submitted by Obey. It allows the robot to deliver a message from one location (parameter #1) to another location (parameter #2).
- Explore (E). Unconstrained task submitted by the Explore motivational source, making the robot move to different locations in the environment.
- Goto (G2). Constrained task generated by Navigate and allowing the robot to move to a specific point (parameter #1) on its map.
- Guard (G). Constrained task making the robot guard an area (parameter #1), at a specific time (parameter #2) for a given period (parameter #3). This task is submitted by Obey.
- Localize (L). Unconstrained task submitted by Explore, allowing the robot to localize itself in its environment.
- ProceedTo (P). Constrained task to move the robot at a specified destination (parameter #1). This task can be submitted by either Plan or Select Task.
- Rest (R). Unconstrained task making the robot remain at the same location. This task can be submitted by Plan and Guard.
- Wander (W). Unconstrained task submitted by Explore, making the robot navigate randomly in its environment.

To illustrate the intended role of the planner in MBA's decision-making process, trials with and without the planner have been executed. When the planner is activated, we expect tasks to be organized and scheduled in an optimal or sub-optimal way, otherwise the robot still will accomplish tasks using the Select Task motivation but without considering time constraints.

To work, our planner uses its own world and state representation. Its world representation is characterized by a list of waypoints, a table of distances for the shortest paths between all pairs of waypoints, and the average velocity of the robot. Based on this model, the planner can derive the travel time between locations. Its state representation is defined by the current time and the robot's nearest (or current) waypoint. Operators in our domain are Goto, Guard, Get-Message and Give-Message. For each operator, a list of preconditions and effects are defined. The optimization criteria are minimizing the distance travel and the time required to accomplish the task. Note that other metrics could have been

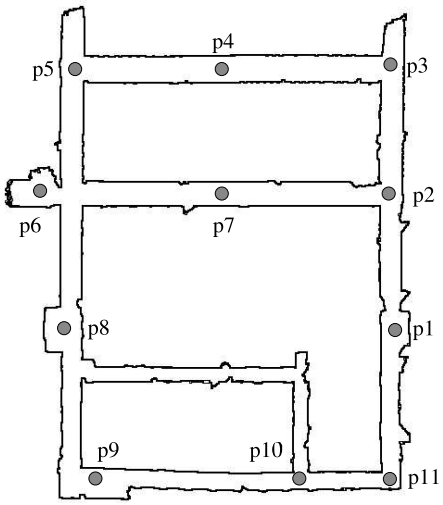

Figure 2: Office floor map for experimentation.

Table 1: Trace of trial #1.

Time	Primitive Tasks (Motivational Source/parameters)
00:00	A(S), E(E), L(E), W(E)
03:20	A(S), E(E), L(E), W(E), G(O/p6), P(ST/p6)
04:12	A(S), E(E), L(E), G(O/p6), D(O/p3,p8), P(ST/p3)
06:20	A(S), E(E), L(E), G(O/p6), D(O/p3,p8), P(ST/p6)
08:29	A(S), E(E), L(E), G(O/p6), D(O/p3,p8), P(ST/p8)
08:44	A(S), E(E), L(E), G(O/p6), D(O/p3,p8), P(ST/p6)
12:42	A(S), E(E), L(E), G(O/p6), D(O/p3,p8)
13:33	A(S), E(E), L(E), D(O/p3,p8), P(ST/p8)
16:31	A(S), E(E), L(E), W(E), D(O/p3,p8)
17:10	A(S), E(E), L(E), W(E)

Table 2: Trace of trial #2.

Time	Primitive Tasks (Motivational Source/parameters)
00:00	A(S), E(E), L(E), W(E)
00:26	A(S), E(E), L(E), W(E), G(O/p6)
00:27	A(S), E(E), L(E), W(E), G(O/p6), P(ST/p1)
00:27	A(S), E(E), L(E), G(O/p6), P(ST/p1)
01:39	A(S), E(E), L(E), G(O/p6), P(ST/p1), D(O/p3,p8)
01:40	A(S), E(E), L(E), G(O/p6), P(ST/p3), D(O/p3,p8)
03:34	A(S), E(E), L(E), G(O/p6), P(ST/p3), D(O/p3,p8)
03:37	A(S), E(E), L(E), G(O/p6), P(ST/p3), D(O/p3,p8)
03:38	A(S), E(E), L(E), G(O/p6), P(P/p6), D(O/p3,p8)
03:43	A(S), E(E), L(E), G(O/p6), P(P/p6), D(O/p3,p8)
05:39	A(S), E(E), L(E), G(O/p6), P(ST/p6), D(O/p3,p8)
07:31	A(S), E(E), L(E), G(O/p6), D(O/p3,p8)
09:35	A(S), E(E), L(E), G(O/p6), D(O/p3,p8), R(P)
09:51	A(S), E(E), L(E), W(E), G(O/p6), D(O/p3,p8), R(P)
16:04	A(S), E(E), L(E), W(E), P(P/p8), D(O/p3,p8)
16:10	A(S), E(E), L(E), P(P/p8), D(O/p3,p8)
17:48	A(S), E(E), L(E), W(E)

Table 3: Trace of trial #3.

Time	Primitive Tasks (Motivational Source/parameters)
00:00	A(Sur), E(E), L(E), W(E)
08:02	A(Sur), E(E), L(E), W(E), G(O/p1), P(ST/p1)
08:08	A(Sur), E(E), L(E), G(O/p1), P(ST/p1)
09:20	A(Sur), E(E), L(E), G(O/p1)
10:54	A(Sur), E(E), L(E), G(O/p1), D(O/p2, p7)
14:54	A(Sur), E(E), L(E), D(O/p2, p7)
15:35	A(Sur), E(E), L(E), D(O/p2, p7), P(ST/p2)
18:32	A(Sur), E(E), L(E), D(O/p2, p7)
18:38	A(Sur), E(E), L(E), D(O/p2, p7), P(ST/p7)
21:12	A(Sur), E(E), L(E), W(E)
21:35	A(Sur), E(E), L(E), W(E), G(O/p6), P(P/p6)
21:42	A(Sur), E(E), L(E), G(O/p6), P(P/p6)
23:04	A(Sur), E(E), L(E), G(O/p6)
24:47	A(Sur), E(E), L(E), W(E), G(O/p6), R(P/p6)

used, such as minimizing energetic consumption: we chose not to consider this metric since it will be handle eventually by an instinctual motivational source.

The first set of trials consists of placing the robot at location $p1$ and requesting a Guard task (using the Obey motivational source at specific times) at location $p6$ under a fixed length of time (200 sec). When this task occurs, the planner creates a plan to accomplish this task, and the robot moves to $p6$. During that time, a Deliver-Message task from $p3$ to $p8$ is requested. A new plan is therefore needed to include this new task, and the robot's next action depends on the time constraints on the Guard task.

Trial #1 (Table 1) illustrates a case with the planner inhibited, intentionally or because it cannot provide a plan (e.g., it may require too much processing time). Initially, the robot wanders until it receives the Guard task at time 3:20. The Select Task module suggests a ProceedTo $p6$ task. At time 4:12, the Deliver-Message task is added. Since the robot is closer to $p3$ than to $p6$, Select Task suggests a ProceedTo $p3$ task. At $p3$ it gets the message and proceed to $p6$, the nearest location. This simple task selection mechanism can cause some oscillations in the pursued tasks, as shown at 8:29 and 8:44: for a short period of time, the robot comes closer to $p8$ than $p6$. Eventually, the robot arrives at $p6$, fulfill the Guard task, and then proceeds to $p8$ since it is the only task left. The robot delivers the message at time 17:10. Task scheduling is done opportunistically, without considering time constraints.

With the planner activated, trial #2 (Table 2) illustrates a case when the robot has sufficient time to get the message at $p3$, but cannot deliver to $p8$ before it is time to execute its Guard task at $p6$. In the first part of the trial, Select Task is faster than Plan to initiate a ProceedTo task to $p1$, and Plan provides the same solution shortly afterwards. At time 20:32, the robot gets a message at location $p3$, and Plan determines that there is not enough time left to deliver the message before Guard must begin. Therefore, it adds a ProceedTo task to $p6$ location. From time 26:29 to time 32:58, the robot remains at $p6$ to fulfill its Guard task. Once Guard done, at time 32:58 Plan adds a ProceedTo task to $p8$ location.

Trial #3 (Table 3) is done with the robotic platform. At the beginning, the planner is inhibited and the robot does a Guard task at location $p1$. A Deliver-Message task from $p1$ to $p2$ and one from $p2$ to $p7$ are added next. The robot ends

its Guard task and, being already at $p1$, gets the first message and goes to $p2$. At $p2$, it delivers the first message and asks for the second. The robot then goes to $p7$ and delivers the second message. At that time, the planner is activated and a Guard task at $p6$ is requested. The robot proceeds to $p6$ and rest. The main objective of this is to show that MBA is able to make coherent decisions in a real dynamic environment under unpredictable situations (e.g., people in the robot's way and differences with the environment model such as opened doors and obstacles), and that the MBA architecture can adapt its decision-making process to on-line addition of the planner during a trial.

Conclusion

The intelligence of a system depends on its sensing, acting and processing capabilities, not taken individually but as a whole. The MBA architectural methodology offers one solution by integrating different motivational sources such as a planner to influence the decision-making process of an autonomous mobile robot. Just using a planner to select behavioral modes would require frequent generation of plans to handle dynamic changes in real life settings. Not using a planner makes it difficult to anticipate and reorganize behavioral strategies. Our objective is to try to find the right balance between the two, using the planner as a motivational source allowing the robot to act rationally by selecting and sequencing primitive tasks.

Results presented in this paper are preliminary as we are still preparing for the AAAI Challenge. Extensive tests are underway to validate the influences of the planner and the other motivational sources on MBA's decision-making process. In future work, we want to refine our planner to generate plans more suitable for plan repairing and for more complex scenarios involving a higher number of tasks, such as localizing a waiting line (to register to the conference) or searching for a docking station. Currently these are handled by non-planned modules; planned tasks would improve time management for the overall robot mission and for accepting more volunteer duties. Handling more tasks makes it harder to replan from scratch every time a new goal is added or canceled. We also want to add more motivational sources and increase the robot's capabilities, validating the MBA architectural methodology in social operational settings with humans and other mobile robots.

Acknowledgments

F. Michaud holds the Canada Research Chair (CRC) in Mobile Robotics and Autonomous Intelligent Systems. Support for this work is provided by the Natural Sciences and Engineering Research Council of Canada, the Canada Research Chair program and the Canadian Foundation for Innovation.

References

Ambros-Ingerson, J., and Steel, S. 1998. Integrating planning, execution and monitoring. In *Proc. National Conf. on Artificial Intelligence*.

Arkin, R. C. 1998. *Behavior-Based Robotics*. The MIT Press.

Bacchus, F., and Kabanza, F. 2000. Using temporal logic to express search control knowledge for planning. *Artificial Intelligence* 116(1-2).

Beaudry, E.; Kabanza, F.; and Michaud, F. 2005. Planning for a mobile robot to attend a conference. In *Proc. Canadian AI Conf.*

Chien, S.; Knight, R.; Stechert, A.; Sherwood, R.; and Rabideau, G. 2000. Using iterative repair to improve the responsiveness of planning and scheduling. In *Proc. Int. Conf. on Artificial Intelligence Planning and Scheduling*.

Cote, C.; Letourneau, D.; Michaud, F.; Valin, J.-M.; Brosseau, Y.; Raievsky, C.; Lemay, M.; and Tran, V. 2004. Code reusability tools for programming mobile robots. In *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*.

Do, M., and Kambhampati, S. 2003. Sapa: A scalable multi-objective metric temporal planner. *J. Artificial Intelligence Research* 20:155–194.

Gerevini, A.; Saetti, A.; and Serina, I. 2003. Planning through stochastic local search and temporal action graphs in LPG. *J. Artificial Intelligence Research* 20:239–290.

Haigh, K., and Veloso, M. 1997. Interleaving planning and robot execution for asynchronous user requests. *Autonomous Robots*.

Haigh, K., and Veloso, M. 1998. Planning, execution and learning in a robotic agent. In *Proc. Int. Conf. on Artificial Intelligence Planning Systems*, 120–127.

Ingrand, F., and Despouys, O. 2001. Extending procedural reasoning system toward robot actions planning. In *Proc. IEEE Int. Conf. on Robotics and Automation*.

Lemai, S., and Ingrand, F. 2004. Interleaving temporal planning and execution in robotics domains. In *Proc. National Conf. on Artificial Intelligence*.

Maxwell, B.; Smart, W.; Jacoff, A.; Casper, J.; Weiss, B.; Scholtz, J.; Yanco, H.; Micire, M.; Stroupe, A.; Stormont, D.; and Lauwers, T. 2004. 2003 AAAI robot competition and exhibition. *AI Magazine* 25(2):68–80.

Michaud, F.; Audet, J.; Letourneau, D.; Lussier, L.; Theberge-Turmel, C.; and Caron, S. 2001. Experiences with an autonomous robot attending the AAAI conference. *IEEE Intelligent Systems* 16(5):23–29.

Michaud, F. 2002. EMIB – Computational architecture based on emotion and motivation for intentional selection and configuration of behaviour-producing modules. *Cognitive Science Quaterly* 3-4:340–361.

Simmons, R., and Apfelbaum, D. 1998. A task description language for robot control. In *Proc. IEEE/RSJ Int. Conf. Intelligent Robotics and Systems*.

Thrun, S.; Fox, D.; and Burgard, W. 1998. A probabilistic approach to concurrent mapping and localization for mobile robots. *Machine Learning* 31:29–53.

Vaughan, R. T.; Gerkey, B. P.; and Howard, A. 2003. On device abstractions for portable, reusable robot code. In *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 2421–2427.