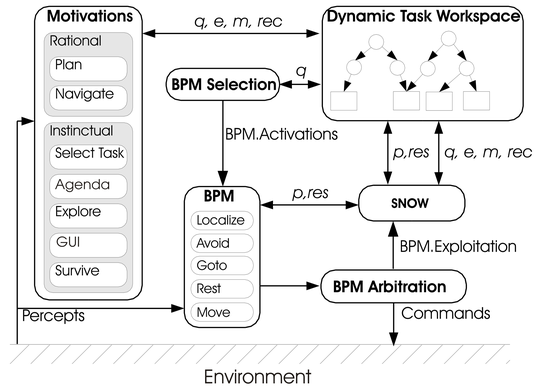
Reactive Planning in a Motivated Behavioral Architecture

Éric Beaudry, Yannick Brosseau, Carle Côté, Clément Raïevsky, Dominic Létourneau, Froduald Kabanza, François Michaud
Université de Sherbrooke, Québec, Canada
http://www.gel.usherb.ca/laborius/ - 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 predetermined 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.

Motivated Behavioral Architecture (MBA)



Motivations

Motivational modules (MM) are high level independent modules that influence the way that the robot is behaving. MM can add, modify, query and recommend tasks in DTW.

Dynamic Task Workspace (DTW)

DTW is a central component that organizes tasks in a hierarchy using a tree-like structure, from high-level/abstract tasks to primitive/BPM-related tasks.

System Know How (SNOW)

The SNOW is a rule-based template that maps low level tasks with BPM.

Behavior-Producing Modules (BPM)

BPMs define how particular percepts and conditions influences the control of the robot's actuators. An arbitration mechanism (priority-based in this implementation, but other methods like fuzzy logic could be used) filters the commands generated by BPMs before applying them to the actuators.

Reactive Planning Process

Inside the Plan motivation. we introduce a reactive planning process. The Planner is decomposed in two sub-processes that can run concurrently. The first one is the ExecMonitor sub-process (for execution and monitoring) that communicates with the Dynamic Task Workspace to extract the robot's current mission. It also adds and recommends lower level tasks to achieve high-level tasks in DTW. The Planning sub-process invokes a planner for planning the mission. The reactive planning process is generic and is not limited to one particular planner.

MBAPlanner_ExecMonitor( ) :

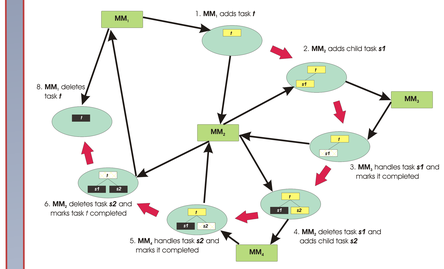
```
while(true) :
  e = waitForEventOrTimeout();
  switch case(e) :
    newTask(t) :
      if(isPlannableTask(t)) :
        mission+=t;
        needreplan = true;
    cancelledTask(t) :
      if(mission.contains(t)) :
        mission -= t;
        needreplan = true;
        if(currenttask.isDecendendOf(t)) :
          currenttask = <notask>;
    completedTask(t) :
      if(t == currenttask) :
        if(currenttask.isLastDecendendOf(currenttask.parent)) :
          workspace.SetCompleted(currentTask.parent);
        currenttask = plan.nextTask();
      if(mission.contains(t)) :
        mission -= t;
        needreplan |= plan.containsDecendantOf(t);
    failedTask(t) :
      if(t == currenttask) :
        currenttask = <notask>
        needreplan = true;
    sensedData(d) :
      currentstate.update(d);

  needreplan |= plan.validate(currentstate)==FAILURE;
  if(needreplan):
    resetPlanningProcess();
  workspace.setRecommendation(currenttask);
```

MBAPlanner_Planning( ) :

```
while(true) :
  // Detect and accept the possibility of an opportunity
  if(plan.progressFasterThanPlanned() &&
    !plan.nextTask().isReady(now + planningAvgDuration))
      needreplan = true;

  // Plan as needed
  if(needreplan) :
    needreplan = false;

    // First : removing unachievable tasks
    for each task t in mission :
      if(planner.fastplan(t) == FAILURE) :
        mission -= t;
        workspace.failedTask(t);

    // Second : try to detect mutex tasks by sufficient condition
    for each task pair {t1, t2} :
      tempmission = {t1, t2}
      if(planner.fastplan(tempmission) == FAILURE):
        mission -= lowestprioritytask(t1, t2);

    while(newplan == FAILURE) :
      newplan = planner.plan(currenttask, mission);
      if(newplan == FAILURE) :
        mission -= lowestprioritytask(mission);

    plan = newplan;
    currenttask = plan.nextTask();
    monitorexec.sendTimeoutEvent();
```

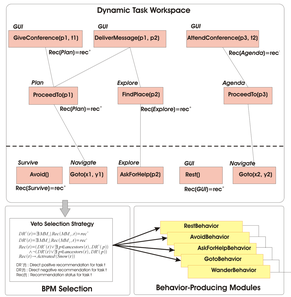
Tasks Mangagement and Selection in MBA

Task Life Cycle



Each MM is responsible for handling a subset of tasks in DTW and each task can be handled by one or more MM. In the case that more than one MM can handle a task, they compete for determining how to achieve the task.

Task Selection for BPM Activations



Experimentation Setup : AAAI Challenge 2005

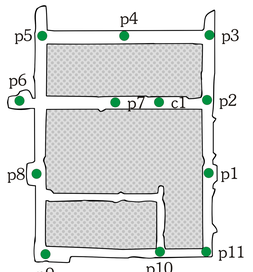
Testing Scenarios

To experiment and validate the MBA architecture, we created a set of scenarios inspired from the AAAI Challenge. In these simplified scenarios, the robot has to perform tasks specified by people in a simulated conference site.

- **GiveConference** : make a presentation at a predetermined location and at a specific time.
- **AttendConference** : listen to a presentation at a specific location during a fixed period of time.
- **AttendPoster** : look at a poster, specified by a location and a duration, during the poster session time window.
- **DeliverMessage** : receive a message to a person at an initial location and bring the message to another person at the destination location.
- **Guard** : guard a location during a fixed time period.
- **Explore** : explore the environment by doing wandering and looking for interesting things (e.g.: tracking symbols).
- **OpenInteraction** : interact with other attendees. Help people or receive help as needed.

Hypothesis

- The metric map of the environment is already known (no mapping to do).
- Locations are partially known : the robot has to found the location position on the map by asking help or searching for landmark.
- Tasks can be given using a graphical interface on the robot and/or directly from a mission text file.



Office map experimentation

Hardware

We use U2S/Spartacus, a UdeS wheeled robot platform equipped with a laser ranger finder for navigation. This robot is controlled by a laptop computer .

Software

Player : low-level interface for communicating with the hardware.
MARIE : software integration environment.
FlowDesigner / RobotFlow : execution control platform for BPM implementation.
ConfPlan : a HTN-based planner with anytime and metrics capabilities.
CARMEN : localization and path planning for position tracking and navigation.

Planning Domain

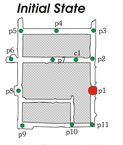
- Navigation table with distances between each location pairs. Dynamically computed as new locations are found.
- Average speed of the robot.
- Operators with preconditions and effects modeling the robot's actions.
- HTN task templates as search controls.
- Optimization criteria :

$$minimize(\alpha * traveldist + \beta * duration + \gamma * \sum_{m}^{messages} waittime(m))$$

Exemples, Tests and Results

Complementarity of Motivational Modules

1 — Initial State



time=0, RobotAt(p1)

2 — Mission

-**Guard** ***p6*** at time=0:15 for a duration of 0:05.
-**DeliverMessage** from ***c1*** to ***p7***.

** Location **c1** is unknown at start.*

3 — Planner Plan

Since **c1** position is unknown, the planner cannot make a good estimation on how to reach it. So, to be sure to guarantee the accomplishment of the **Guard** task, it assumes the worst case, that is **c1** is very far. The consequence is that going to ***p6*** is the first action.

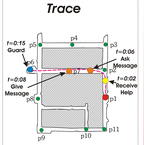
4 — Instinctive Explore

While the Plan MM is executing its plan by recommending a **ProceedTo**(***p6***) task, the Explore MM is recommending finding location **c1**. This forces the GUI to ask for help. A person clicks on the robot's touch screen to show **c1** position on the displayed map.

5 — Planner Replan

When the Plan MM is notified about the position of **c1** (between ***p2*** and ***p7***), the planner is reinvoked to find a new plan. Because **AskMessage** and **GiveMessage** is feasible before **Guard**, the planner chooses this order.

Trace



Mission Optimization and Failure Detection from the Planner

One objective with the MBA architecture is not to have one central module on which all decisions depend on. At the same time, the MBA wants to take advantage of a deliberative module (from AI planning) to improve overall performance of the architecture.

To show that MBA is not centralized on a planner, test cases demonstrate that the system can continue to operate and achieve totally or partially submitted missions.

We generate random tests that executed with and without the planner.

Random Tests

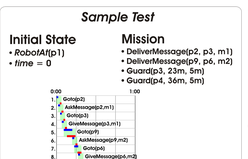
Initial State
- *RobotAt*(Random_Place)
- *time* = 0

Mission
- 1 or 2 **DeliverMessage** with random origin and destination places
- 1 or 2 **Guard** at random places and times
- 1 or 2 **AttendConference** at random places and times

Sample Test

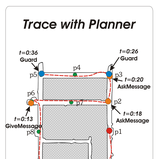
Initial State
- ***RobotAt*(p1)**
- ***time* = 0**

Mission
- **DeliverMessage(p2, p3, m1)**
- **DeliverMessage(p9, p6, m2)**
- **Guard(p3, 23m, 5m)**
- **Guard(p4, 36m, 5m)**

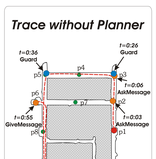


Optimal plan for this test

Trace with Planner



Trace without Planner



Conclusion

The intelligence of a system depends on its sensing, acting and processing capabilities, not taken individually but as a whole. The work presented here 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.

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* (NSERC), the Canada Research Chair program and the Canadian Foundation for Innovation.

Many students on the project are supported by NSERC and by the *Fonds québécois de la recherche sur la nature et les technologies* (FQRNT) programs.