Continuous online task planning and plan acting in a deterministic robot environment

Ahmed A Al-Moadhen¹, Ali R Khayeat², and Maysam J AlMayli³
¹College of Engineering, University of Kerbala, Karbala, Iraq
²College of Sciences, University of Kerbala, Karbala, Iraq
³University of Kerbala, Karbala, Iraq

Abstract. In a deterministic environment, continuity in planning robot tasks and acting upon the generated plans is essential. This aspect does not commit the planner to generate all the required actions in advance, however; instead, the robot planner can generate the necessary abstract actions, then make the actor responsible for refining these actions into commands. Such commands are executed directly by the robot platform. In this paper, the planning and acting framework, known as RosPlanAct, is modified by including a temporal model that deals with the time interval for each action in the plan. Thus, a chronicle is associated with the abstract actions during the planning phase to act as a base for refining these actions into more detailed activities. The extended framework also has the ability to postpone the execution of any action that requires more details and to activate the monitoring function to compare what was observed with predicted results. This framework was tested in simulation and in real environments. The simulation environment was ROS, with a smart robot car used as a real testing environment. The results showed high performance in accomplishing the specified tasks with an increasing rate of success in executing the actions in the determined period of time and a low rate of failed execution. Throughout this paper, the actor was supported by being given the ability to avoid failure by online interactions between planning and acting sections in order to trigger the executing platform to perform the given task in a deterministic environment. The robot can thus refine the actions in the plan, re-planning as required in order to handle events arising.

1. Introduction

Achieving robot tasks correctly requires the unification of planning and acting subsystems. In order to maintain robot objectives, the planning system must be consulted as to which action is best to accomplish the objectives at a given moment, while the acting system is also consulted (after planning) as to the manner in which to perform each action in a plan. It is thus important for the acting system to have the ability to decide what refined actions to pursue, given the current environmental characteristics, in order to achieve the stated
goal(s). Increasing interaction between planning and acting enables the robot to be more responsive and autonomous.

In reality, environments offer a diversity of open conditions; autonomous robots should thus be able to deal with several different types of tasks. For these reasons, a robotic system must deliberate as part of its main functions. According to [1], the main five deliberation functions of any robotic system are planning, acting, monitoring, observing, and learning (goal reasoning). Planning is driven by acting [2], in that the former deals with generating a sequence of actions, which called a plan, that is predicated on accomplishing the given task [3]. The latter works upon this plan to find ways to command the execution platform to start doing the task. The monitoring function recognises variances between predicted and observed states, identifying both possible causes and recoveries. The observing function offers the ability to diagnose the state of the robot’s activity and the related objects and environment.

This paper is concerned mainly with the planning and acting deliberation functions and how these interleaf in order to achieve robot tasks. The planner is conscious of the fact that the execution of the plan will begin only once planning finishes because the plan is generated, while the actor is aware that the acting process begins once the plan is generated and disposed of for execution. The actor functions are detailed in [4]; these consist of plan refinement, reaction to events, instantiation, time management, nondeterminism, and plan repair or re-planning.

However, in cases where external timing constraints are imposed, such as emergencies and deadlines, the system becomes more complex. To tackle this problem, this paper proposes a continuous online task planning and plan acting framework. This framework extends the main framework explained in [5] and builds upon it. Online planning and acting solves the problem of meeting deadlines and emergencies and takes into account all of the necessary time for planning, disposing of action, and acting; thus, planning and acting can be blended as described in [6].

The rest of this paper is organised as follows: Section 2 describes the system architecture and main components, and then section 3 specifies how the system is modelled and represented and the new algorithms adopted to deal with framework activities. Section 4 is dedicated to testing the system and collecting the necessary results for validation. Finally, conclusions and suggestions for the future work are presented in section 5.
2. The System Architecture

Figure 1 represents the proposed framework architecture for continuous online planning and acting. It is important to specify that this architecture is used only in deterministic environments. This framework includes an acting component (subsystem), which represents the core of the system and is responsible for

- Querying the planning section about the necessary sequence of actions, that is, the plan that achieves the given goal.
- Refining the actions into low-level actions used to command the robot.
- Dealing with the external events that might impede the robot.
- Re-planning, when necessary if the cost of failure is more than the cost of updating or repairing the plans.

![Diagram of System Architecture](image)

**Figure 1.** Overview of Planning and Acting System with a continuous Planning and Acting Loop

The other part of the system is the planning subsystem, which consists of a classical planner that can handle temporal events. This planner is applicable to online planning for robotics, where a robot must generate a plan to be executed but the rest of the world does not stop while the robot is planning and acting according to its plan.
2.1 Related Architectures

The field of planning has seen many contributions in terms of search and prediction, but the field of acting lacks such contributions. In this section, a number of architectures that combine planning and acting are thus reviewed and compared with the proposed architecture in this paper.

The work in [7] and [8] described the ROSPLAN framework, which is an architecture for inserting task planning into Robot Operating Systems (ROS) [9]. This is focused on embedding the planning system into ROS, but these works do not deal with the other important part of any robotic system, the acting part.

The planning and acting model seen in [10] and [11] specified a model for temporal planning combining hierarchical and generative procedures. This model was constrained particularly by the requirement to provide the system with the ability to deal with temporal events and resolve flaws in the plans in order to achieve the objective; however, it did not express clearly how the high-level action translated into low-level commands to the execution platform. The proposed model, in contrast, provides the actor with the ability to refine high-level actions into low-level commands by exploiting various skills.

In recent work in [12], the authors presented a planning framework known as PLATINUM that has the ability to deal with temporal uncertainty in events both at the planning and plan execution level, with no acting stage. Thus, PLATINUM can detect errors at the time of execution as it has no acting stage.

The ActorSim toolkit in [13] was used to study autonomous systems based on goal reasoning. Goal reasoning has strong ties to planning, acting, and robotics; however, it is an area of research that is currently understudied and which requires more investigation. Currently, no available language exists publicly that offers all of the necessary definitions and universal implementations required.

This review of existing systems suggests that it is necessary to develop a system that clearly defines its acting part as the core of the system and coordinates the robot activities to address any failure in plan execution due to unexpected (exogenous) events. The work in this paper thus seeks to fill this gap in previous systems in the field.
3. System Modelling and Representation

The proposed system is an extension of the work in [5]; however, this paper focuses on the concept of continuous online planning and acting.

3.1 Planning

The planning part of the framework is modelled as described in [4):

1. In classical planning, the domain is specified using tuple-4 variables \( \Sigma = (S, A, \gamma, \text{cost}) \), where the cost is optional, \( S \) is a finite set of states that the system may be in, \( A \) is a finite set of actions that may be used by the actor to change the state and environment, and \( \gamma : S \times A \rightarrow S \) is prediction function (or state-transition function) such that

   - partial function \( \gamma(s,a) \) is not defined unless \( a \) is applicable in \( s \)
   - \( \text{Dom}(a) = \{ s \in S \mid \gamma(s,a) \text{ is defined} \} = \{ s \in S \mid a \text{ is applicable} \} \)
   - \( \text{Range}(a) = \{ \gamma(s,a) \mid s \in \text{Domain}(a) \} \)
   - \( \text{cost}: S \times A \rightarrow \mathbb{R}^+ \) -or- \( \text{cost}: A \rightarrow \mathbb{R}^+ \)
   - where \( \text{cost}(a) \equiv 1 \), this can be omitted from the domain definition.

2. Classical planning problem: \( P = (\Sigma, s_0, S_g) \)

   - \( \Sigma \): planning domain, \( s_0 \): initial state, \( S_g \): set of goal states.

3. Solution for \( P \): a plan (sequence of actions) that will produce a state in \( S_g \)

   - \( \pi = \langle a_1, a_2, \ldots, a_n \rangle \)

4. Action: (head, preconditions, effects, cost)

   - head: name and parameter list. Get actions by instantiating the parameters.
   - preconditions: Computationally test to predict whether an action is applicable in a state \( s \)
   - effects: Procedures that modify the current state

5. \( \pi \) is applicable in \( s_0 \) if the actions can be applied in the order given, i.e., there are states \( s_1, s_2, \ldots, s_n \) such that

   - \( \gamma(s_0, a_1) = s_1 \), \( \gamma(s_1, a_2) = s_2 \), \ldots, \( \gamma(s_{n-1}, a_n) = s_n \)

   If so, define \( \gamma(s_0, \pi) = s_n \)

   - \( s_0 \in S \) is the initial state
   - \( g \) is a set of ground literals called the goal

6. \( S_g = \{ \text{all states in } S \text{ that satisfy } g \} \)
\[ s \in S \mid s \cup R \text{ contains every positive literal in } g, \text{ and none of the negative literals in } g \]

7. If \( \gamma(s_0, \pi) \in S_g \), then \( \pi \) is a solution for \( P \).

### 3.2 Acting

Throughout the acting process, the actor refines and monitors its actions, reacts to events, and extends, updates, and repairs its plan based on its perception of the relevant parts of the environment. All these functions are completed by issuing commands to the specified system components, parts of the RosPlanAct framework [5]. The main activities in the actor section are thus:

1. **Command**: a primitive function that orders the execution platform to perform a given action. For example, the action \( \text{take}(r,o,l) \) can be used as a primitive function and sent as a command to order the robot \( (r) \) to pick up an object \( (o) \) at a location \( (l) \).

2. **Task**: any activity the actor seeks to perform; for each task, there is a set of refinement methods.

3. **Refinement Methods**: operational models that inform the robot how to perform the given task where they have no ability to predict what will result. The general form of the refinement method is as follows [6]:

   ```
   \text{method-name}(\text{arg}_1, \ldots, \text{arg}_k)
   \text{task: } \text{task-identifier}
   \text{pre: } \text{test}
   \text{body: a program}
   ```

### 3.3 Planning and Acting Algorithms

This section examines the two relevant types of algorithms. The first deals with planning activities and how actor commands are handled. The second deals with acting activities and how the actor handles acting functions. In Figure 2, an algorithm for task planning and disposing of the actions is shown, and Figure 3 depicts an algorithm for acting upon the plan.
To perform, for example, an open-door action in different contexts, the robot must depend on a set of different skills to accomplish the given task. These skills are defined by methods and specified in order to refine an action into commands.

### 3.4 Time Management

Online planning and acting manages the time required for generating a plan and executing its actions. Each action is thus attached to various time points: $t_s$, the time when the action started, $t_e$, the time when the action ended, and $t_n$, the current time.
These time points can be used in both planning and acting. In planning, \( t_s, t_e, \) and \( t_n \) represent the start, end, and current planning time, respectively. The time elapsed executing an action or planning the task can thus be calculated as \( t_e = t_n - t_s \).

No action can start before planning starts, as the action refinement process does not begin until the planner has generated a plan. So, for each action \((a)\) in the plan \((\pi)\), a constraint exists that the time at which acting will start \((t_{as})\) must be less than the time set for the disposal of action \((t_s)\), i.e., \( t_{as} \leq t_s \).

4. **Experiment and case study**

In this section, the proposed framework, including its model and algorithms, is tested and verified. The planning benchmark \([14]\) is used to test the performance of the proposed system. Planning and acting are done under the ROS environment, with planning and acting processes represented as ROS nodes that communicate and exchange messages across ROS topics as shown in **Figure 4**. The planning domain and the problem to solve are as explained in \([5]\). These are related to moving a car down four types of paths: Rectangle, Triangle, Multiline path, and Polygon paths. The details of the proposed platform (car) are also explained in \([5]\).

![Figure 4. ROS nodes and topics for the RosPlanAct [5]](image-url)
4.1 System Nodes, Topics, and Messages

The main nodes in the system are the planning node and the acting node. Table 1 shows the main messages used to exchange information between the system nodes through ROS topics.

| Message Name     | Message Type | Topic Name     | Specification                                      |
|------------------|--------------|----------------|---------------------------------------------------|
| Plan             | Action       | dispatchPlan   | Used to post the plan to the actor                |
| ActionDispatch   | Action       | dispatchAction | Used to post the action of the plan action by action |
| FeedBack         | Action       | feedbackAction | Used to get feedback about the state of the action |

4.2 System Performance

The proposed framework uses two types of planner. These planners are classical; one of them, LPG-td [15], does not deal with the time, while the second one, popf [16], does deal with time. The system specifications are as follows: processor speed of 2.7 GHz, core i7, and a memory size of 4 GB.

Table 2 shows the system performance in term of response time, which represents the time between starting planning and the actor dispatching a command to the platform. The response time also depends on the problem size, i.e., the number of objects in the problem space. System performance using the popf planner is better than the performance when using the LPG-td planner as, in the case of popf, each action is allocated a required time to implement while as the other planner does not deal with time, the system is required to attach the time. Time is attached based on experience and can also be obtained from the skills manager. Figure 5 represents the response time for both LPG-td and popf planners for different problem sizes. It is clear that response time for the popf planner is better than for the LPG-td planner.

| Problem size | LPG-td Response time (seconds) | popf Response time (seconds) |
|--------------|-------------------------------|-----------------------------|
| 10           | 0.22                          | 0.13                        |
| 30           | 1.3                           | 1.07                        |
| 60           | 2.1                           | 1.8                         |
| 100          | 2.9                           | 2.4                         |
In a real environment, the experiment was repeated 100 times for a problem size of 50 objects for both planners; the results are summarised in Table 3. It is clear that the system using the popf planner has more success states than the LPG-td planner, due to the ability of popf planner to manage temporal events.

**Figure 5: The response time of LPG-td and popf planners**

| Planner type | Success states | Failed states | Unknown states |
|--------------|----------------|---------------|----------------|
| LPG-td       | 60             | 35            | 5              |
| Popf         | 80             | 17            | 3              |

5. **Conclusions and Future Work**

Empirical evaluation revealed that the proposed framework has the ability to handle continual online planning and acting problems. The system can integrate planning and acting independent of planner type. However, it is preferable to use a planner that deals with temporal issues to obtain better performance. While this framework can use a classical planner that does not deal with time, this requires the system to specify the amount of time to be attached to each action in the plan by exploiting experience and the skill manager.
Integrated planning and acting can help unify robot objectives and can specify what actions must be taken to achieve the task and how to perform these actions. This leads to the concept of goal reasoning.

In the future, however, it will be necessary to support the proposed framework with the capability to deal with uncertainty and probabilistic issues in both deterministic and non-deterministic robot environments.

References

[1] Ingrand F and Ghallab M 2017 Deliberation for autonomous robots: A survey Artif. Intell. 247 10–44

[2] Ghallab M, Nau D and Traverso P 2014 The actor’s view of automated planning and acting: A position paper Artif. Intell. 208 1–17

[3] Ghallab M, Nau D and Traverso P 2004 Automated Planning: Theory and Practice (Morgan Kaufmann)

[4] Ghallab M, Nau D and Traverso P 2016 Automated Planning and Acting (Cambridge University Press)

[5] Al-Moadhen A A, Abdulhussein A M and Kamil H G 2018 Planning and acting framework under robot operating system IOP Conf. Ser. Mater. Sci. Eng. 433

[6] Nau D, Ghallab M and Traverso P 2015 Blended Planning and Acting: Preliminary Approach, Research Challenges Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence pp 4047–51

[7] Cashmore M, Fox M, Long D, Magazzeni D, Ridder B, Carrera A, Palomeras N, Hurtos N and Carreras M 2015 ROSPlan: Planning in the Robot Operating System Proc. Int. Conf. AI Plan. Sched. 333–41

[8] Sanelli V, Cashmore M, Magazzeni D and Iocchi L 2017 Short-Term Human-Robot Interaction through Conditional Planning and Execution Proc. Int. Conf. Autom. Plan. Sched. 540–8

[9] Quigley M, Gerkey B, Conley K, Faust J, Foote T, Leibs J, Berger E, Wheeler R and Ng A 2009 ROS: an open-source Robot Operating System ICRA Workshop on Open Source Software pp 1–9

[10] Bit-Monnot A 2016 Temporal and hierarchical models for planning and acting in robotics
[11] Dvorak F, Bartak R, Bit-Monnot A, Ingrand F and Ghallab M 2014 Planning and Acting with Temporal and Hierarchical Decomposition Models Proc. - Int. Conf. Tools with Artif. Intell. ICTAI 2014-Decem 115–21

[12] Umbrico A, Cesta A, Cialdea Mayer M and Orlandini A 2017 PLATINUm: A new framework for planning and acting AI*AI pp 498–512

[13] Roberts M, Hiatt L M, Coman A, Choi D, Johnson B and Aha D W 2016 ACTORSIM, A Toolkit for Studying Cross-disciplinary Challenges in Autonomy AAAI Fall Symposium pp 202–8

[14] Pommerening F, Torralba Á and Balyo T 2018 IPC 2018 Exhibitors 1–2

[15] Gerevini A, Saetti A, Serina I and Toninelli P 2004 LPG-TD: a fully automated planner for PDDL2.2 domains 14th International Conference on Automated Planning and Scheduling (ICAPS-04) International Planning Competition abstracts (Whistler, Canada)

[16] Coles A, Coles A, Fox M and Long D 2010 Forward-Chaining Partial-Order Planning Twentieth International Conference on Automated Planning and Scheduling (ICAPS 2010)