Interleaving Planning, Scheduling, and Execution in Mobile Robots

Lenka Mudrova and Nick Hawes
School of Computer Science, University of Birmingham, United Kingdom

Abstract

This paper gives one possible answer to the question of "whether we should use planning or scheduling to control a robot in order to execute a task specified by a user?". To be able to answer this question, we specify a new research problem, which actually interleaves both planning and scheduling techniques. In addition, the problem also contains three new research questions. The first question relates to estimating of an initial state of a task, second to recognising similar actions across tasks and third to minimising negative effects of uncertainty. Finally, we present a homogeneous representation of the problem, which can be expressed as a conditional directed labelled graph, and we propose a hierarchy structure to solve the problem.

1 Introduction

Assume that a mobile robot operates in a hospital building. The robot receives the set Ω containing a variety of tasks to perform:

- ω_1: "*Check emergency exits today.*" This task has a relatively large time window when it should be performed.
- ω_2: "*Find a free wheelchair before 14:00.*" A wheelchair can be located anywhere in the building.
- ω_3: "*Map the second floor of the building today.*" This task represents a robot's curiosity which often has a lower priority than user's tasks.
- ω_4: "*Update the database after performing* ω_3."

The activities for ω_2 and ω_3 can have pre-defined actions.

If we assume that the robot can understand and execute given tasks, then such autonomous robot needs to solve the following problems:

1. **What** actions must it perform?
2. **When** should it execute those actions?

These questions are more general and they appear in a wide array of research areas. Thus, considerable research has been undertaken to address each of them.

Classical planning techniques as summarised in (Nau, Ghallab, and Traverso 2004) can be used to answer question (1). Moreover, they were extended to *temporal planning* and to *resource planning* to also address question (2). In contrast, *scheduling* as summarised in (Pinedo 2012) is used to answer only question (2). In conclusion, planning and scheduling techniques overlap in their approaches to solve aforementioned questions.

In real world scenarios as our model example, there is no clear-cut boundary when to use them. This unclear boundary is also address in the existing research, for example (Smith, Frank, and Jónsson 2000), (Barták 1999). Therefore, this paper firstly points out some advantages and disadvantages of the temporal planning and scheduling from the point of view of an autonomous mobile robot. Secondly, it extends the problem by introducing three novel research questions. Finally, the paper introduces a new *homogeneous* approach that incorporates the advantages of planning and scheduling.

2 Problem definition

First, we clarify the terms used in this paper. Then, we point out why neither standalone scheduling, nor standalone planning are sufficient techniques for solving our problem. In addition, we formulate new open questions related to the problem, when a solution to them has the potential to increase the overall performance of a robot.

2.1 Terms definition

The terms *task, activity* and *action* are used differently in literature.

Task We refer to a single task by ω with numeric subscript i, for example ω_1. Its properties are then referred by the same subscript.In this paper, a *task* ω_i has the following properties:

- time properties:
  - *release date* r_i (the earliest time instant when a task can start),
  - *deadline* d_i,
- weight function $w_i(t)$, which penalises the completion time of a task and it is used in a scheduling optimisation criteria,
- an *activity* π_i to perform.

Activity Each activity π_i has:

- a description, for example "*check the fire extinguisher*",

- a goal state s_g, for example s_g = *objectKnownState(fireExtinguisher, loc5)*
- a sequence of actions $(a_{i_1}, a_{i_2}, \ldots, a_{i_m})$.

Action Actions $(a_{i_1}, \ldots, a_{i_m}) \subset A$ are indivisible and a robot is able to perform them. The set of all actions, which robot can perform, is referred by A and is defined by a specific robot's domain $\mathcal{D}$. Actions can be used in any combination to create a new activity for a new task.

2.2 Scheduling approach

Scheduling finds execution time instants for tasks in order to guarantee that a chosen optimisation criterion is maximised (or minimised). For example, the criterion using total completion time should be minimal in order to achieve all task as soon as possible. Tasks are then executed in those time instants. However, ω_2 from model example cannot be executed, as the robot does not know *what* actions to perform. Therefore, planning is needed.

In addition, ω_2 and ω_3 can have significantly different durations which are not known beforehand as they depend on a robot's environment. For example, when assuming that the robot has no ability to open office doors, the robot checks state of each door during execution and can only map offices, it can visit. Therefore, the duration of ω_3 depends on how many offices have open doors. This issue can be solved by *conditional scheduling* (also referred as contingent) (Drummond, Bresina, and Swanson 1994).

2.3 Planning approach

Planning answers *what* actions should be performed to achieve a goal state from an initial state. This is necessary to perform ω_2. Furthermore, temporal planning techniques can answer the question of *when* these actions should be executed. Thus, they will solve our model example.

Temporal planning works as follows. First, the goal states of the given tasks are combined by logic conjunction in a global goal state. Then, a planner answers both question (1, 2) at the same time by producing a sequence of time dependent actions. The advantage is that the tasks are performed in an interleaved manner - the robot can perform some action for ω_2, followed by ω_3 and then ω_2 again. This interleaving of different tasks increases the robot's efficiency. However, the conjunction of the goal states leads to larger search spaces for sets containing more tasks. For tens of tasks, this approach is slow. As we assume tens of tasks in our domain, this is not sufficient approach to us. Therefore, we will use planning only for a single tasks. Scheduling will be used for answering question (2) in global case.

ω_2 and ω_3 require techniques of *conditional planning* for the same issue as explained previously in conditional scheduling. In conclusion, we need to use conditional temporal planning for a single task in order to know *what* actions should be executed and *how much time* they will take.

2.4 Going beyond the state of the art

Lets assume that scheduling and planning techniques are both used in some structure which is able to answer *what* to and *when* to execute a set of actions. However, there are more unsolved questions behind our model example.

Question A What is the initial state of a task and when is it valid?

The first question refers to the fact that planning needs as an input an initial state. But, what is the initial state for a task ω_i? Is it the state of a robot's world at release date? Is it the state when we expect that a task will have been executed? If so, when will the task have been executed? What is the initial state for a task with a large time window for execution as ω_1 has? Is the initial state the same state in the morning and in the afternoon? What about ω_4, which depends on the completion time of ω_3? All these questions are summarised in question A.

Question B How can a robot realise which actions of tasks are similar and perform them together if time constraints are not violated?

The second question relates to the fact, that planning is called for a single task in order to keep a search state space small. As a result, a robot performs tasks in a serialised way. The next task can only be executed, once the current task has been completed. However, we – people – very often fulfil tasks in an interwoven way which can increase our efficiency. For example, we take a letter to the post office while heading to a restaurant for lunch (assuming that the post office is more or less on the way).

Question C When shall a robot make decisions in order to minimise negative effects of partial observability?

The third questions refers to partial observability in the system which occur in many places. For example, how long does an action take, what is the state of state variables, what to do if a task cannot be fulfilled as the deadline for it has already passed. Partial observability might invalidate an earlier decision thus causing rescheduling and replanning, which negatively affects the robot's performance.

2.5 Summary of the problem

We will refer to the combined scheduling and planning problem by the Greek letter Ξ. To summarise, it contains:

- conditional temporal planning to answer the question of *what* actions should be executed and *how much time* will they consume,
- conditional scheduling to answer the question of *when* actions should be executed,
- execution with monitoring system in order to choose between conditional branches and to change taken decision if an action takes much time then it was expected.
- new approaches to answer Questions A-C.

Input:

- set $\Omega = \{\omega_1, \omega_2, \ldots, \omega_n\}$ of tasks to be performed,
- temporal constraints $\mathcal{C}$ on the tasks which are:

  - unary - $\mathcal{C}\omega_i$ relates to the fact that task ω_i needs to start after release date and end before deadline,
  - binary - $\omega_i \mathcal{C} \omega_j$ refers to the fact that the relation of two tasks can be constrained, as in our model example $\omega_3 < \omega_4$,
- for each task ω_i a planning space $\Sigma_i = (S_i, A_i, E_i, \gamma_i)$
- domain $\mathcal{D}$ describing a robot's world.

.

Output: Conditional time-dependent sequence of actions, for example $(\{a^t_{i_1}\}, \{a^t_{i_2}\}, \{a^t_{i_3}, a^t_{i_4}\}, \{a^t_{i_5}, \emptyset\})$. In this example, the two following sequences are possible:

- either $(a^t_{i_1}, a^t_{i_2}, a^t_{i_3}, a^t_{i_5})$,
- or $(a^t_{i_1}, a^t_{i_2}, a^t_{i_4})$.

3 Proposed representation

The defined problem can be expressed as a tuple

$$\Xi = (\Omega, \mathcal{C}, \Sigma, \mathcal{D}).$$

This tuple can be represented as *conditional directed labelled graph*

$$G_\Xi = (N, E, L),$$

which is inspired by representation of scheduling as a Simple Temporal Network (STN) (Dechter, Meiri, and Pearl 1991). The nodes $N = \{T, \mathcal{P}, S\}$ can be one of three different types:

- T is a set of time nodes. Each node represents an exact time instant. We will use the symbol $\square$ in a visualisation of a graph.
- $\mathcal{P} = (\Sigma, s_0, s_g)$ is a planning node, which refers that a plan will take place in the graph, but the final plan is still unknown. We will visualise it by $\triangle$.
- As the plan is known, the planning node is replaced by the plan represented by state nodes S and action edges. A state node contains description of a state based on domain D. We will refer to it by $\bigcirc$.

According to these nodes, conditional, directed edges $E = \{\tau, \delta, \alpha\}$ are also three different types:

- Time edges τ can only link two different time nodes. The time difference Δt between them can be computed.
- Dependency edges δ link two nodes n_1, n_2 in a sense that node n_1 needs to happen before n_2. Thus, dependency edges set up an order on nodes, but we do not know how much time is between nodes. The dependency edges can link:
  - a time node with a plan node, or opposite order,
  - a time node with a state node, or opposite order,
  - two plan nodes.
- Action edges α transform one state node to another one.

The different symbols for all edges used for a visualisation of a graph can be seen in Fig. 1.

The edges are labelled by $L = \{(\Delta t, w(t)), (\emptyset), (a, p)\}$:

- Time edges are labelled by the time difference Δt and the weight function $w(t)$. The weight function for edges connecting release and due dates is set by the particular task. For other time edges, the weight function is constant to one.
- Dependency edges are not labelled.
- Action edges are labelled by the name of a performed action a from the set A and also by its estimated processing time p.

In summary, the graph is

$$G_\Xi = (\{T, \Sigma, S\}, \{\tau, \delta, \alpha\}, \{(\Delta t, w), (\emptyset), (a, p)\}).$$

4 Proposed solution - a hierarchy

The described conditional directed labelled graph G_Ξ can be used to solve given problem Ξ. We propose a hierarchy structure to construct such a graph and use it to solve the problem. First, a simplified graph G_{in} is created from time unary constraints $\mathcal{C}$ of tasks in a pre-stage. These constraints are represented by time nodes and time edges, see Fig. 1 for our model example. The binary constraint is represented as a dependency edge. We also know that a plan node will take a place between the release and due dates, but we do not when, thus we use the dependency edges. Then, following hierarchy containing seven stages extends this graph. The influence of each stage to the graph G_Ξ is shown in Fig. 1[1].

Stage 1: Prediction

The input to this top layer is a simplified graph G_{in}. The prediction layer answers question (A). It predicts several best time intervals, which are limited by time nodes t_s^- and t_s^+, and corresponding initial states node $s'_{0i}, s''_{0i}, \ldots$ which are valid within the intervals. For example, ω_1 has two possible initial states, see Fig. 1. For intervals, we will extend ideas presented in (Laborie 2003).

Stage 2: Conditional planning

For each task and each prediction, conditional plans are produced as graphs containing states and action edges, see ω_4 in Fig. 1. The final state of a plan is a task's goal. We assume to use existing temporal planner in this stage.

Stage 3: Least commitment scheduling

Some tasks might have large time windows. Thus, one prediction can be for the near future, another one for the far future. It is more likely that the first prediction would be more certain comparing to another one. This stage answers question (C) by using techniques of least commitment scheduling (Berry 1993) to set time horizon. For tasks which start within the interval from now to this horizon, below stages will provide more certain decision. However, the below stages will be applied later for those tasks which start behind time horizon, which is the case of ω_4 in Fig. 1.

[1]Please, print this paper in colour to see the figure correctly

Stage 4: Time dependent plan mixing

This stage answers question (B) by combining two branches in conjunction relation to a single branch, as for ω_1 and ω_2 in Fig. 1. This is done by putting states of original plans in different order and by adding interconnecting states. We will built on plan repair (Krogt and Weerdt 2005) and refinement planning techniques (Kambhampati, Knoblock, and Yang 1995) despite the fact that it was proved that modifying an existing plan is no more efficient than a complete replanning in general case (Nebel and Koehler 1995). Plan repair and plan refinement might be still efficient our case as we combine it with a scheduler.

Stage 5: Time dependent plan merging

Plan merging analyses two situations. First, it finds which actions are same. Thus, one of them can be eliminated, see *Stage5b* label in Fig. 1. Second, it analyses which actions can be combined to a new action with shorter processing time, see label *Stage5a*. The important extension of the plan merging relates to the fact that two merged tasks still need to satisfy their time constraints. We will extend the work (Tsamardinos, Pollack, and Horty 2000).

Stage 6: Conditional scheduling

The previous stages produce more conditional branches. The processing time of branches might strongly differ. Thus, conditional scheduling answers what shall be executed next (in a new time horizon) depending on which of the branches will be performed. In our model example, only possibility is to execute ω_3 afterwards, as ω_4 follows it.

Stage 7: Execution and monitoring

This stage is the lowest level which controls that a robot executes an action from the conditional schedule. It also monitors not only actions processing times, but also world state variables and choose between branches based on observations.

5 Discussion

This paper presents the novel problem Ξ connecting planning and scheduling. Moreover, it goes beyond the state of the art by adding three new research questions. We introduce homogeneous representation as a conditional directed labelled graph G_Ξ. We are aware that there are related problems such as resources that we did not covered in our problem. A resource with limited capacity can be a property of a single robot or it can relates also to multiple robots operating in the world. This is one of the possible extensions of our proposal. We believe, it can be solved by constructing separate graphs G_Ξ for each resource and finding such a solution, which is valid in all of graphs.

In conclusion, we aim for a compact representation that allows a robot to quickly and reliably decide *what* and *when* to execute in order to perform a given task.

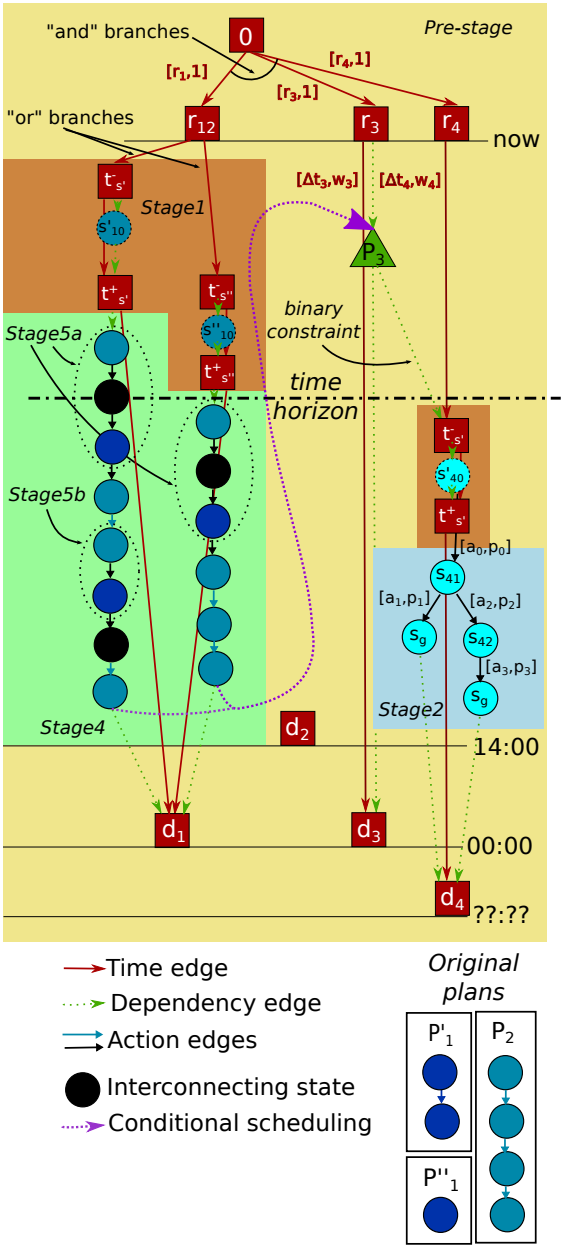

Figure 1: Due to lack of space in this paper, all stages are shown in one graph G_Ξ. However, stages are normally applied in serialised way, thus they influence all tasks and we will need more graphs to visualise it.

6 Acknowledgements

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement No 600623, STRANDS.

References

[Barták 1999] Barták, R. 1999. On the boundary of planning and scheduling: A study. Technical report, Faculty of Mathematics and Physics, Charles University, Czech Republic.

[Berry 1993] Berry, P. M. 1993. Uncertainty in scheduling: Probability, problem reduction, abstractions and the user. In *IEE Colloquium on Advanced Software Technologies for Scheduling, Digest No*.

[Dechter, Meiri, and Pearl 1991] Dechter, R.; Meiri, I.; and Pearl, J. 1991. Temporal constraint networks. *Artificial Intelligence* 49(1-3):61–95.

[Drummond, Bresina, and Swanson 1994] Drummond, M.; Bresina, J.; and Swanson, K. 1994. Just-in-case scheduling. In *In Proceedings of the Twelfth National Conference on Artificial Intelligence*, 1098–1104.

[Kambhampati, Knoblock, and Yang 1995] Kambhampati, S.; Knoblock, C. A.; and Yang, Q. 1995. Planning as refinement search: A unified framework for evaluating design tradeoffs in partial-order planning. *Artificial Intelligence* 76:167–238.

[Krogt and Weerdt 2005] Krogt, R. V. D., and Weerdt, M. D. 2005. Plan repair as an extension of planning. In *In Proceedings of the 15th International Conference on Automated Planning and Scheduling (ICAPS-05*, 161–170.

[Laborie 2003] Laborie, P. 2003. Algorithms for propagating resource constraints in ai planning and scheduling: Existing approaches and new results. *Artificial Intelligence* 143(2):151–188.

[Nau, Ghallab, and Traverso 2004] Nau, D.; Ghallab, M.; and Traverso, P. 2004. *Automated Planning: Theory & Practice*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

[Nebel and Koehler 1995] Nebel, B., and Koehler, J. 1995. Plan reuse versus plan generation: A theoretical and empirical analysis. *Artificial Intelligence* 76(1-2):427–454.

[Pinedo 2012] Pinedo, M. L. 2012. *Scheduling: Theory, Algorithms, and Systems*. Springer Science+Business Media, 4th edition.

[Smith, Frank, and Jónsson 2000] Smith, D. E.; Frank, J.; and Jónsson, A. K. 2000. Bridging the gap between planning and scheduling. *Knowledge Engineering Review* 15.

[Tsamardinos, Pollack, and Horty 2000] Tsamardinos, I.; Pollack, M. E.; and Horty, J. F. 2000. Merging plans with quantitative temporal constraints, temporally extended actions, and conditional branches. 264–272. AAAI.