Hierarchical Hybrid Planning in a Mobile Service Robot

Sebastian Stock[1,2(✉)], Masoumeh Mansouri[3], Federico Pecora[3], and Joachim Hertzberg[1,2]

[1] DFKI-RIC Osnabrück Branch, 49069 Osnabrück, Germany
sebastian.stock@dfki.de
[2] Osnabrück University, 49069 Osnabrück, Germany
[3] Örebro University, 70182 Örebro, Sweden

Abstract. Planning with diverse knowledge, i.e., hybrid planning, is essential for robotic applications. However, powerful heuristics are needed to reason efficiently in the resulting large search spaces. HTN planning provides a means to reduce the search space; furthermore, meta-CSP search has shown promise in hybrid domains, both wrt. search and online plan adaptation. In this paper we combine the two approaches by implementing HTN-style task decomposition as a meta-constraint in a meta-CSP search, resulting in an HTN planner able to handle very rich domain knowledge. The planner produces partial-order plans and if several goal tasks are given, subtasks can be shared, leading to shorter plans. We demonstrate the straightforward integration of different kinds of knowledge for causal, temporal and resource knowledge as well as knowledge provided by an external path planner. The resulting online planner, CHIMP, is integrated in a plan-based robot control system and is demonstrated to physically guide a PR2 robot.

Keywords: Robot planning · Hierarchical task networks · Cognitive robotics

1 Introduction

Robot task planning has to be hybrid, i.e., span over temporal, spatial, and resource reasoning, in addition to task sequence and condition achievement. Consider a waiter robot as used as the demo example throughout the paper (Fig. 1). To serve sugar and a hot coffee, it must reason about the consequences of each action's duration (reasoning about time), explore alternative ways of bringing coffee and sugar considering available resources (reasoning about method decomposition and resources), and obtain a kinematically feasible path (reasoning about space). A solution to each subproblem has to take into account the solutions of the others. In fact, any feasible plan fulfilling the high-level requirements

This work was supported by the EC Seventh Framework Program theme FP7-ICT-2011-7, grant agreement no. 287752.

S. Hölldobler et al. (Eds.): KI 2015, LNAI 9324, pp. 309–315, 2015.
DOI: 10.1007/978-3-319-24489-1_28

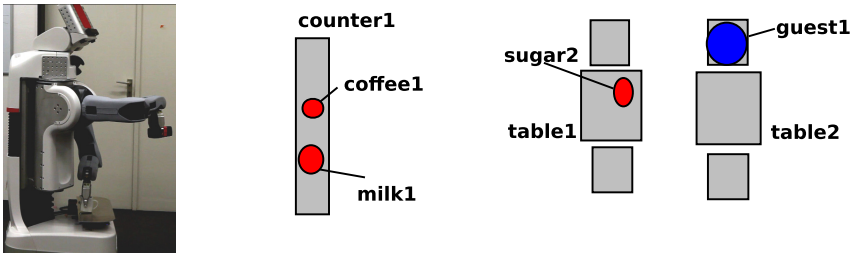

Fig. 1. Demo scenario. Photo: PR2 carrying milk pot and coffee jug. Sketch: Part of the fake restaurant layout and initial situation. Another counter2 is located far away.

(e.g., serve a "hot" coffee) is a solution in the cross-product of the individual search spaces. Note that many of these search spaces correspond to domain-dependent knowledge that varies among different robot platforms. Therefore, as proposed by [8], the different sub-spaces should be integrated in a general way to avoid designing a new integrated algorithm for each new application.

In response to this challenge, we use a meta-CSP approach [8] for achieving hybrid problem solvers. It is based on principles of constraint reasoning and the notion of abstraction through meta-constraints, which represent high-level requirements. However, as casting hybrid robot planning as a meta-CSP means searching for a plan in the cross-product of the different representation sub-spaces (time, space, resources, etc.), a powerful means for guiding the search is needed to keep this feasible. In this paper, we propose to use HTN planning [4] as this means. It focuses dramatically the search through the cross-product search space by its decomposition rules. We show how to integrate HTN planning in a straightforward way in the disguise of a meta-constraint in the meta-CSP; this is done in our new planner CHIMP ("Conflict-driven Hierarchical Meta-CSP Planner"). The resulting plans are hybrid owing to the meta-CSP representation, and they are hierarchical owing to the HTN decomposition structure, which keeps being visible in the final plan representation.

2 Related Work

Integrating HTN planning with other forms of reasoning has been attempted in several ways. ANML [10] is an expressive planning language supporting temporal relations, resource usage, and HTN methods. The first planner to integrate most of ANML, including HTN task decomposition, is the recently introduced planner FAPE [3]. It interleaves deliberative acting, plan repair and re-planning.

Attaching a theory procedurally to a set of symbols (e.g., a task or state) is common practice in combining task and motion planning, and is essential for online planning. Predicates act as interfaces between the discrete space of high-level specification and the continuous space of robot configuration. Many application systems use HTN planning as a means of high-level domain modeling for task planning. [9] interleave geometric planning and HTN-based task planning.

Predicates are shared between the two planners, allowing them to backtrack to a certain level in the joint search space. [5] integrate task and motion planning with a hierarchical planner. It uses the hierarchy not only as a heuristic, but commits to choices at an abstract level and starts to execute parts of the hierarchy without creating a full plan. This assumes that actions are reversible, which is not required by our approach, and it does not include temporal and resource reasoning, and it creates only total-order plans. [6] verifies kinematic feasibility of choices made by HTN planning through geometric backtracking.

While the mentioned approaches aim at integrating specific forms of knowledge, we argue that the problem should be solved in a more general way. Therefore, we extend the work by [8], which casts the problem of reasoning about action into a meta-constraint. Although capable of combining task planning with other forms of reasoning, it does not leverage sophisticated planning heuristics, nor does it provide hierarchical decomposition capabilities in its domain specification language, an issue we address explicitly in this paper.

3 Approach

A meta-CSP [8] is a high-level CSP representing a hybrid problem in different levels of abstraction. Meta-constraints impose high-level requirements on a common constraint network that is called a *ground-CSP*. Parts of this constraint network that do not adhere to these requirements are called *meta-variables*. Meta-variables represent flaws, the resolution of which are *meta-values*, i.e., different ways of resolving a flaw. Meta-constraints and their meta-variables define a *meta-CSP*, i.e., a constraint network at a higher abstraction level.

Therefore, our planner CHIMP uses a constraint-based representation for its state and tasks. This allows CHIMP to impose requirements by adding constraints or variables. Its variables in the ground-CSP are *fluents* that consist of a predicate symbol, a set of symbolic variables, a flexible temporal interval, within which the predicate evaluates to `true`, and a function that indicates the fluent's use of reusable resources. We distinguish state fluents and task fluents. They can be bound by three types of constraints: temporal, binding and causal constraints. We use convex relations in Allen's Interval Algebra (IA) [1,7] as temporal constraints. Binding constraints restrict the domain of symbolic variables of fluents. They are used to ground methods and operators. Causal constraints represent the causal relations of the resulting plan. For details we refer to [11].

To do HTN planning in the meta-CSP approach, we represent standard HTN task decomposition as a meta-constraint. Its meta-variables are the set of unplanned task fluents with no unplanned predecessors. These conflicts get resolved either by applying an operator or method to the unplanned task, or by unifying the unplanned task to a previously planned task fluent. In both cases additional constraints or variables are added to the ground-CSP. As usual in the meta-CSP approach, propagation in the underlying constraint networks is applied. If this leads to an inconsistency in one of those constraint networks, we backtrack.

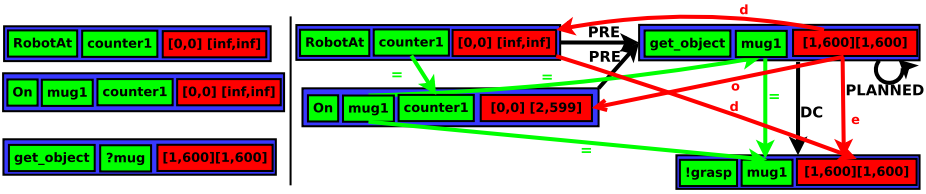

Fig. 2. Left: Constraint network of the initial situation. Predicates and variables of a fluent are green, time intervals are red. Right: Result of applying a method. Causal constraints are black, binding constraints are green and temporal constraints are red.

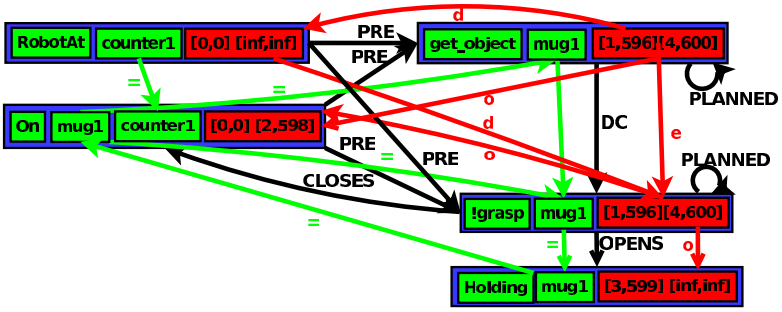

Fig. 3. Result of applying a method and an operator.

Fig. 2, left, shows an example. The constraint network consists of two fluents representing the initial situation (the robot is at counter 1 and mug 1 is on counter 1) and a fluent for the task of getting some mug. Applying a method to get_object results in the right constraint network of Fig. 2. The method connects the RobotAt and On fluents as preconditions to the task fluent. This adds binding and temporal constraints, too, by which the symbolic variables of the get_object fluent are ground. Furthermore, the new task fluent !grasp is created and connected to get_object with a decomposition relation (dc) and temporal and binding constraints. As the robot is already at counter 1, the task get_object is decomposed to the single subtask !grasp. If the robot were at a different position, another method that involves driving would be used.

Next, we can apply an operator to the fluent !grasp (cf. Fig. 3). Analogously to the previous method, it adds constraints for the preconditions. The fluent On is a negative effect of grasping. Therefore, the causal relation closes is added. As a positive effect the robot is now Holding mug 1. The finish times of both effects are updated according to the temporal relations they have with the operator.

Fig. 4 gives an example for unifying a task to a task that was already planned. There are two goal tasks for getting mugs: one for mug 1 and another for mug 2. Initially, the conflict is the second task for driving, which is unplanned. This can be resolved by adding a unification constraint to the other driving tasks that has already been planned. This way the second drive need not be decomposed.

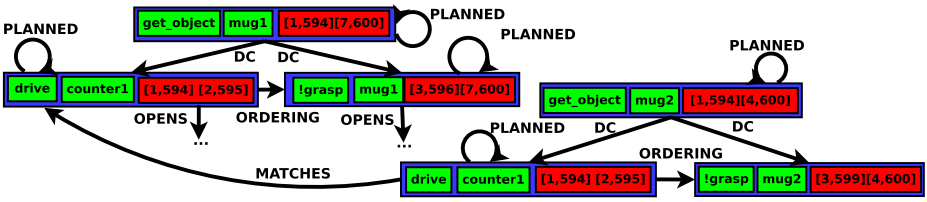

Fig. 4. Constraint network after unifying the tasks for driving. The Binding and temporal constraints and preconditions and effects are omitted for simplicity.

Whereas the HTN approach as such provides a very powerful means to reduce the search space, choosing appropriate value-ordering heuristics is important. The value-ordering heuristic used in our scenario favors, first, unification of tasks; second, a low number of subtasks; and third, binding preconditions to fluents with a late starting time. Details of this are out of the scope of this paper.

An advantage of the meta-CSP approach is that we can enforce other kinds of high-level requirements by adding further meta-constraints. In our demonstration scenario it is important for the robot to estimate the time it takes to drive from one area to another to make sure that tasks that have a deadline will not take too long. This depends on the restaurant layout itself but also objects like unforeseen chairs that block the robot's path. An appropriate source for this kind of knowledge is the robot's path planner. Therefore, a meta-constraint is created whose meta-variables are tasks of type !move_base for which no duration was assigned. Such a conflict is resolved by setting its expected duration based on a the distance between the start and goal poses calculated by the robot's path planner by way of procedural attachment.

Resource reasoning becomes crucial when generating partial-order plans. To ensure resource feasibility a meta-constraint is added like in [8], who use a precedence constraint posting method as proposed by [2].

4 Performance Example

We demonstrate our approach in the restaurant scenario shown in Fig. 1. Our robot gets the goal task of serving a coffee to guest 1, The latest finish time is set to 600 sec. to make sure that the coffee is served hot. Serving a coffee implies that a sugar pot and a milk jug have to be on the table, too. We model this requirement as subtasks in the method for serving coffee.

The initial situation has two sugar pots, one on counter 2 and one on table 1. The only milk jug is on counter 1. Standard HTN planning could create a plan that involves driving to counter 2 in the kitchen to get the sugar. This is causally feasible, but the guest would get served cold coffee. As CHIMP is aware of time, it notices that this plan takes too long, i.e., is temporally infeasible. Therefore, it tries alternative HTN methods that lead to using the other sugar from table 2.

An estimate of the expected duration of driving is added to the constraint network as a meta-value posted by the meta-constraint encapsulating the path

planner (see previous section). This duration accounts for the actual restaurant layout and provokes the search to choose method decompositions that do not conflict with the temporal requirement. This is but one example of other kinds of external knowledge that the planner can use, thanks to the meta-CSP approach.

A domain representation consisting of 11 operators and 28 methods was used. After sending the goal to the robot, it generated a plan containing 39 operators, of which Listing 4.1 shows the first 8 with their predicates and flexible temporal intervals in milliseconds. The time points indicate that the plan is partially ordered, e.g., !move_torso and !tuck_arms have the same earliest start time, and both arms may be moved to the side in parallel. By using resource constraints we made sure that it can only manipulate one object at a time, as the robot has to look at the object. Therefore, it picks up coffee 1 after picking up milk 1. In the remainder of the plan the robot completes the goal task, using sugar 2 from table 1. Note that the domain contained no methods for moving multiple objects. The planner used its partial-order planning capability and the task unification to already planned tasks for interleaving the tasks of moving the coffee and the milk. As a result of its limited holding capacity it was planned to bring the sugar after the first two objects.

The complete plan-based execution cycle was run on a physical PR2 robot. CHIMP's planning time was 20.5 seconds. For plan execution, a time-line based dispatching approach was employed. The constraint network is permanently updated as the time proceeds and actions are finished. An action is dispatched as soon as its earliest start time is less than the robot's time. For details about the integration on the real robot and an evaluation of the runtime for various demo problems we refer to [11], which also describes how plans for additional goal tasks can be merged online into an existing plan that is being executed.

```
!move_base(preMAreaECnt1)          [12, 464970], [30012, 494970]
!move_torso(TorsoUpPosture)    [30013, 498972], [34013, 502972]
!tuck_arms(UnTucked UnTucked) [30013, 494971], [34013, 498971]
!move_arm_to_side(lArm1)       [34014, 498972], [38014, 502972]
!move_arm_to_side(rArm1)       [34014, 498972], [38014, 502972]
!move_straight(mAreaEastCnt1) [38015, 502973], [42015, 506973]
!pick_up_object(milk1 rArm1)  [42016, 506974], [46016, 510974]
!pick_up_object(coffee1 lArm1)[46016, 510974], [50016, 514974]
```

Listing 4.1. First 8 operators of the plan for serving a hot coffee with milk and sugar.

5 Conclusion and Outlook

We have presented the hierarchical hybrid planner CHIMP that combines the advantages of HTN planning and meta-CSP reasoning; and have integrated it on a physical PR2 robot. With HTN task decomposition as a meta-Constraint it employs a powerful tool to restrict the huge hybrid search space – a part that was lacking in the meta-CSP approach. It produces partial-order plans with actions that can be executed in parallel. This and the sharing of subtasks may lead to shorter plans without the need of modeling further HTN methods.

More kinds of knowledge may be introduced by attaching further meta-Constraints. We demonstrated this by example of an external path planner; we will extend it with spatial knowledge as done in [8]. In other future work we will investigate means for repairing parts of the plan and we will explore value ordering heuristics for meta-values in more detail.

References

1. Allen, J.: Towards a general theory of action and time. Artif. Intell. **23**(2), 123–154 (1984)
2. Cesta, A., Oddi, A., Smith, S.F.: A constraint-based method for project scheduling with time windows. Journal of Heuristics **8**(1), 109–136 (2002)
3. Dvorak, F., Bit-Monnot, A., Ingrand, F., Ghallab, M.: A flexible ANML actor and planner in robotics. In: Proc. of Planning and Robotics Workshop at ICAPS (2014)
4. Erol, K., Hendler, J., Nau, D.: HTN Planning: complexity and expressivity. In: Proc. AAAI, pp. 1123–1128 (1994)
5. Kaelbling, L.P., Lozano-Pérez, T.: Hierarchical planning in the now. In: IEEE Conference on Robotics and Automation (ICRA) (2011)
6. Lagriffoul, F., Dimitrov, D., Saffiotti, A., Karlsson, L.: Constraint propagation on interval bounds for dealing with geometric backtracking. In: Proc. of IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems (2012)
7. Ligozat, G.: A new proof of tractability for ORD-Horn relations. In: AAAI Workshop on Spatial and Temporal Reasoning (1996)
8. Mansouri, M., Pecora, F.: More knowledge on the table: Planning with space, time and resources for robots. In: IEEE International Conference on Robotics and Automation (ICRA) (2014)
9. de Silva, L., Pandey, A.K., Alami, R.: An interface for interleaved symbolic-geometric planning and backtracking. In: IROS. IEEE (2013)
10. Smith, D.E., Cushing, W.: The ANML language. In: Proc. of the Scheduling and Planning Applications Workshop at ICAPS (2008)
11. Stock, S., Mansouri, M., Pecora, F., Hertzberg, J.: Online task merging with a hierarchical hybrid task planner for mobile service robots. In: Proc. IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems (IROS) (2015) (in press)