
What About Hybrid Planning

Hui Li
Brian Williams

MIT Computer Science and Artificial Intelligence Laboratory, 32 Vassar Street, 32-275, Cambridge MA, 02139 USA

ROOTLESS@CSAIL.MIT.EDU
WILLIAMS@MIT.EDU

1. Introduction

In the past decade the planning community has moved its research interest out of the blocks world, and focused on more realistic planning problems, which involve time and many other types of constraints. One important category is planning for mobile robots, such as planetary rover exploration and unmanned aerial vehicle team coordination. It would be ideal if the human operator only needs to give a high level goal, for example, "let the three rovers collect all the science samples in the vicinity with as little fuel use as possible", and the planner outputs a set of optimal trajectories for the robot team so that the high level goal is accomplished. We call such a problem Hybrid Planning (HP).

Previous research has divided the problem into two parts: first, planning without considering the environment and vehicle dynamics; and second, designing trajectories based on the plan. In realistic problems, however, this two-fold approach often does not lead to optimal solutions and sometimes is not feasible to execute. To the author's knowledge, up till now there has not been any planner capable of doing HP. This is because incorporating trajectory design into planning makes the problem intractable. The paper addresses the HP problem, and proposes that with a smart encoding and efficient inference methods the problem can become manageable.

2. Motivative Example

As mentioned before, high level planning is usually separate from trajectory planning for its manageable computation requirement. However, without considering the environment and dynamics the higher level plan can be sub-optimal or even infeasible to execute. Consider the following toy example.

One of the 2 robots, Bender or Gender Bender, has to go to rescue Leela and then come back to its home. If their energy runs out, they have to be towed to a brewer to be *energized*. The 2 robots have the same traveling speed but Bender consumes energy much faster than Gender Bender. Leela has to be saved within 60 minutes, otherwise she will

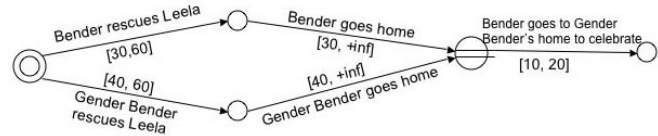


Figure 1. The temporal planning network with choices

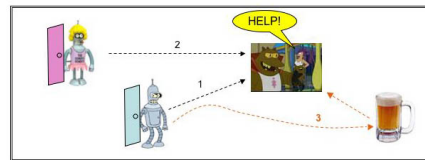


Figure 2. Illustration of the environment. If the problem is divided into two parts, the planner chooses route 1 and ends up route 3 and fails the mission, while the optimal solution to the problem as a whole is route 2.

die. The rescue takes Bender at least 30 minutes and Gender Bender at least 40, because Bender lives closer. The objective is to minimize the total mission completion time.

Fig. 1 shows the high level goal, which in this case is a temporal planning network (TPN) with choices. The HP problem is to find the optimal set of trajectories for the robots so that the mission is finished as early as possible. In Fig. 2 we can see that if we divide the problem into two, as done by previous work, the solution to the first part is Bender does the rescue and then during trajectory design infeasibility of the plan is found. Sadly, Leela dies.

3. Problem Statement

The paper addresses a simple version of the HP problem, the input to which is a high level goal in the form of a TPN with choices¹, as well as the environment, robot dynamics and a cost function. The output is an optimal trajectory for a robot or a set of optimal trajectories for the robot team to reach the goal with the least cost.

¹It will be extended to generative planning in the future.

4. Related Work

The planning community has moved increasingly towards application of planners to problems that involve time and other resources. PDDL3.0 (Gerevini & Long, 2006) is the latest version of the widely used planning language PDDL, which includes numeric expressions, durative actions, objective functions and preferences. LPGP (Long & Fox, 2002) and TGP (Smith & Weld, 1999) are both examples of temporal planners that have exploited the Graphplan foundation. LPSAT (Wolfman & Weld, 1999) and RIPP (Koehler, 1998) handle planning under resource constraints. TPN with choices like in the example has also been studied and solved as conditional CSP (Effinger, 2006). However, none of the work considers robot dynamics and plan trajectories.

On the other hand, research has been done in trajectory design based on low level plans (Léauté & Williams, 2005). Task assignment on top of trajectory design for UAVs has been also intensely studied (Bellingham et al., 2001)(Alighanbari & How, 2005). However, none of the work can take as input high level goals, such as the TPN in the example.

5. Proposed Approach

This section will first introduce a formulation for the HP problem that can benefit from existing advanced techniques in areas such as CSP, and then discuss efficient solution methods.

5.1 Formulation

We formulate the HP problem as a mixed discrete continuous optimization problem, which involves finite and infinite domain variables, along with logical and linear (or non-linear) constraints. For simplicity we focus on linear constraints, but the formulation is not limited to linearity. The problem is formally defined in Definition 1, which can be considered a conditional optimal CSP combined with linear programming (LP).

Definition 1 - The HP problem is formulated as a 8-tuple $\langle I, V, C_I, X, C_X, I_A, C_A, f \rangle$, where I is the set of all finite domain variables that may appear in the problem; V is the set of the finite domains corresponding to I ; C_I is the set of constraints associated with I ; X is the set of the infinite domain variables; C_X is the set of constraints associated with X ; $I_A \subseteq I$ is the set of non-empty initially active variables; C_A is the set of activity constraints describing when variables become active; and f is the objective function.

Consider the example in Fig. 3, $I = \{A, B\}$, $V = \{\{a1, a2, a3\}, \{b1, b2\}\}$ and $I_A = \{A\}$. C_A includes

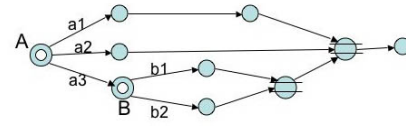


Figure 3. The TPN with multiple choice nodes

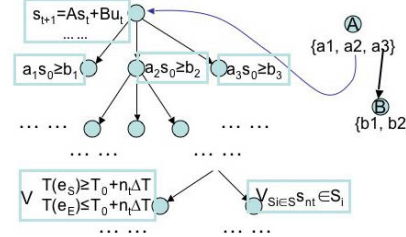


Figure 4. At the top right corner is the dependency graph. An example of the non-clausal LP tree at a leaf node $\{A=a2\}$ is shown on the left. The root of the tree contains all the non-disjunctive constraints, such as state equations. Each time a node splits into children, one disjunction is added into the constraint set inherited from the parent node.

$A = a3 \Rightarrow active\{B\}$ and constraints stating that assignments to the discrete variables activate linear constraints. C_X can contain state equations, $s_{t+1} = As_t + Bu_t, \forall t = 0, \dots, N_t$, obstacle avoidance inequalities, $\bigwedge_{t=0, \dots, N_t} \bigvee_i a_i^T s_t \ge b_i$, timing constraints, $\{T(e_S) \le T_0 + n_t \Delta T \wedge T(e_E) \ge T_0 + n_t \Delta T\} \Rightarrow \{\bigvee_{S_i \in S} s_{nt} \in S_i\}$, etc..

5.2 Solution Methods

The advantages of the formulation are that firstly it preserves the structure of CSP and therefore CSP techniques like tree decomposition (Kask et al., 2005) can apply, and secondly the continuous constraints can be dealt with relatively independently so that the conflict learning strategy in (Li & Williams, 2005) can be used to prune search space. For the simple version of HP, where a TPN with choice nodes is the high level goal, the constraint network formed by its discrete variables and constraints is not complicated enough to apply tree decomposition. Hence it will be discussed in the future for the generative HP case, and in this paper we will focus on conditional optimal CSP and conflict learning.

We propose to solve the HP as follows. First, as in (Effinger, 2006) from the discrete variables and the activity constraint set C_A , create a dependency graph, shown in top right corner of Fig. 4 for the example in Fig. 3. Second, search over the dependency graph and for each assignment check temporal consistency and symbolic constraints in C_I . Third, at each consistent leaf node of the graph a set of linear constraints are activated. Finally, as shown on

the left of Fig. 4 the non-clausal LP at each leaf node of the dependency graph is solved through a search tree, where conflict learning adapted from (Li & Williams, 2005) is applied. More specifically, infeasible and sub-optimal nodes are detected along the search tree and learned as conflicts to reduce search space. It has shown to be powerful in speeding up search in the CNF setting. The differences from it lie in that first we now have a more general form than before, non-clausal instead of CNF, and second since the continuous tree structure at each leaf node of the dependency graph is similar to one another, we can solve them efficiently by learning conflicts incrementally.

References

- Alighanbari, M., & How, J. (2005). Cooperative task assignment of unmanned aerial vehicles in adversarial environments. *Proceedings of the IEEE American Control Conference*.
- Bellingham, J., Tillerson, M., Richards, A., & How, J. (2001). Multi-task allocation and trajectory design for cooperating uavs. *Proceedings of Coordination, Control and Optimization*.
- Effinger, R. (2006). Optimal temporal planning at reactive time scales via dynamic backtracking branch and bound. *MIT S.M. Thesis*.
- Gerevini, A., & Long, D. (2006). Plan constraints and preferences in PDDL3. *The Language of the Fifth International Planning Competition*.
- Kask, K., Dechter, R., Larrosa, J., & Dechter, A. (2005). Unifying cluster-tree decompositions for reasoning in graphical models. *Artificial Intelligence Journal*.
- Koehler, J. (1998). Planning under resource constraints. *Proceedings of ECAI*.
- Léauté, T., & Williams, B. (2005). Coordinating agile systems through the model-based execution of temporal plans. *Proceedings of AAI*.
- Li, H., & Williams, B. (2005). Generalized conflict learning for hybrid discrete/linear optimization. *Proceedings of CP*.
- Long, D., & Fox, M. (2002). Fast temporal planning in a graphplan framework. *Proceedings of ICAPS*.
- Smith, D., & Weld, D. (1999). Temporal planning with mutual exclusion reasoning. *Proceedings of IJCAI*.
- Wolfman, S., & Weld, D. (1999). The Ipsat engine and its application to resource planning. *Proceedings of IJCAI*.