The LMTD Planner
On the Discovery and Utility of Precedence Constraints in Temporal Planning

Yanmei Hu[1], Dunbo Cai[2] and Minghao Yin[3]

Northeast Normal University, Changchun, China[1, 3]

huym260@gmail.com[1] , ymh@nenu.edu.cn[3]

Wuhan Institute of Technology, Wuhan, China[2]

dunbocai@gmail.com[2]

Abstract

LMTD is a satisfied planning system based on heuristic search. Based on the work of Eyerich et al (2009), it extends the precedence constraints contexts heuristic (h^{pcc}) (Cai et al. 2009) to a temporal and numeric setting. Its core feature is the rules, which are derived from landmarks and used to account precedence constraints among comparison variables and logical variables. It builds on the TFD Planning System, using multi-valued state variables and any time search. The planner will continue to search for plans of better quality until the limited search time is due or the remaining search space is empty.

Introduction

LMTD is a temporal planning system based on heuristic state space search. It builds on the TFD (Eyerich et al, 2009), inheriting the general structure of TFD. PDDL tasks with binary state variables are translated to *SAS+* formulism with multi-valued state variables, and comparison variables are introduced for considering numeric resources. A search architecture is designed to search solution in the search space. Every search state is time stamped, due to temporal planning considering temporal dependencies. A heuristic named *temporal precedence constrains contexts* (h^{tpcc}) is derived from landmarks to guide the search, in the spirit of temporal h^{cea} notated as h^{tcea} (Eyerich et al, 2009) and h^{pcc} (Cai et al. 2009). h^{tpcc} considers the precedence constraints over both comparison variables and logical variables, while h^{tcea} considers dependencies among fluents and h^{pcc} only suit for fluents.

Structure of Planner

LMTD consists of three separate programs:

1. the translator (written in Python). This part is used to translate temporal PDDL tasks to temporal *SAS+* tasks. Here we directly use the translator in TFD.
2. the knowledge compilation module (written in C++). In this part, a number of data structures, including data transformation graphs (DTG) and causal graph (CG), are built from the temporal *SAS+* task representation generated by the translator. These data structures play a central role in the generation of landmarks and search. The more details about knowledge compilation are referred to (Helmert, 2004). To handle numeric resources, comparison variables are introduced. In these data structures, comparison variables are also considered. The more details about comparison variables are referred to (Eyerich et al, 2009).
3. the search engine (also written in C++). Using the data structures generated by the knowledge compilation module, the search engine attempts to find a plan using heuristic search with some enhancements, such as the use of *preferred operators* and *deferred heuristic evaluation* (just like that in Fast Downward). LMTD applies a greedy best-first search as the search algorithm and use the *temporal precedence constraints context heuristic* (h^{tpcc}) to guide the search (in the next section, we will introduce the heuristic in detail). Once a plan solution is found, it searches progressively for better solutions until the search is terminated.

To solve a planning task, the three programs are called in sequence; they communicate via text files.

Temporal Precedence Constraints Context Heuristic

Like in TFD, for using the heuristic to guide search, Instant actions are extracted to approximate the durative actions. And for the numeric variables, they do not directly occur in conditions or in the goals but only influence them via comparison variables, so it is sufficient to consider these comparison variables and logical variables (details are referred to Eyerich et al (2009)). In h^{tcea} , notice that other conditions is evaluated in the context state satisfying the pivot condition. That's mean for an atom, we first achieve the pivot condition of the chosen rule and get a context state, and then the remaining conditions are satisfied in the gotten context state. However, in many cases, it is not

reasonable to evaluate conditions in this order. Here we follow the example in Eyerich et. al (2009). Assuming that there are three locations l_0, l_1 and l_2, robot r_1 is at l_1 with a water tank which has capacities of $c = 150$ units and is initially empty $w = 0$, additionally, the robot can only refill its tank at l_0 and there is a path between every two locations. Now it is required to water flower f_1 at location l_1 and flower f_2 at location l_2 with each has current water levels $h_i = 0$ and needs to be watered until levels $n_i = 10$ is reached ($i = 1, 2$). For water some flower at l_j, the rule could have the form "$f_i = unwatered, at(l_i), w - (n_i - h_i) \geq 0 \rightarrow f_i = watered$" ($n_i - h_i = 10$, so the water tank must be fill in at least 10 units water). Hence the estimated cost of $f_1 = watered$ based on h^{tcea} is $d_{01} + 10$ (d_{ij} is the distance between d_i and d_j). While it is more reasonable if we first achieve the condition $w - (n_1 - h_1) \geq 0$, and then achieve $at(l_1)$ in the context state satisfying $w - (n_1 - h_1) \geq 0$. We can obtain a more precise estimated cost $10 + 2 * d_{01}$. Therefore, in this section, we define the h^{tpcc} heuristic function, extending h^{tcea} to capture the order information discussed above. Here we first give out an important notation and introduce the equation of h^{tpcc}, then we give the highlight to how to obtain the precedence constraints which indicating the orders over conditions.

Context Functions

Here we borrow Cai et al.'s notation, each instant action is transformed to a set of rules, and rules are considered to satisfy some condition. Context function is a partial function $ctx : R \times P \mapsto P$, where $ctx(r, q) = p$ indicates that the context of condition $q \in Z_r$ should be the state that results from achieving $p \in Z_r$. Given a rule r, assume that we have constructed the context function for conditions Z_r (how to construct it? We will leave this for later), then a directed and acyclic graph corresponding to the context functions can be build, which explicitly show the precedence constraints between conditions. Each node indicates one condition in Z_r, and there is an edge from p to q if $ctx(r, q) = p$. Moreover, each node has at most one parent to ensure that every condition in Z_r just has one context. So we can get that the corresponding graph is a forest,
and $Z_r^L := \{y \mid y \in Z_r, not \text{ ex. } y' \in Z_r \text{ s.t } ctx(r, y') = y\}$ denotes the leaves, $Z_r^R := \{y \mid y \in Z_r, ctx(r, y) \text{ is undef.}\}$ denotes the roots.

The definition of h^{tppc}

By introducing the precedence constraints over conditions Z_r, we derive h^{tpcc} from h^{tcea}, and

$$h^{tpcc}(x \mid s) = \begin{cases} 0 & \text{if } x \in s \\ \min_{r: Z_r \rightarrow x} (c(s') + \sum_{y \in Z_r} h^{tpcc}(y \mid s^{tc}(y, r, s))) & \text{if } x \notin s, \\ & \text{var}(x) \notin V_c \\ \min_{\substack{r: Z_r \rightarrow v \circ v' \in \\ prom(x,s)}} (c(s') + \sum_{y \in Z_r} h^{tpcc}(y \mid s^{tc}(y, r, s))) & \text{if } x \notin s, \\ & \text{var}(x) \in V_c \end{cases} \quad (1)$$

This equation has the similar structure with h^{tcea} and naturally be extended. In our heuristic function we don't just limit the context state to the pivot condition, but also consider the context state corresponds to the other conditions. Each condition y is evaluated in the resulted state after achieving its direct parent condition in the landmark graph. The resulted state is projected by the function $s^{tc}(s, r, y)$. This function is a map from three sets: states, rules and conditions to the sets of states. In this function, r is applied to achieve x from the corresponding context state s, y is specified to achieve in the new state obtained by the function itself. Similarly, we have the following formal result:

$$s^{tc}(s, r, y) = \begin{cases} s & \text{if } ctx(r, y) \text{ is undef} \\ s[s^{tc}(y', r, s)][r'] & \text{if } ctx(r, y) = y' \end{cases} \quad (2)$$

In the first case in equation (2), that's if $ctx(r, y)$ is undefined, then the context state of y is the same as the original state s applying r. While in the second case, that's if $ctx(r, y) = y'$, then the context state of y is the state that results from achieving y', in corresponding context state indicated by $s^{tc}(s, r, y')$, captured by iterating from its respective root $y_0 \in Z_r^R$. s^{tc} also consider the changes by the rule r' applied to achieve y'. Moreover, what is the state after achieving x? Here we need to consider two things. One is that for achieve x, we should achieve the conditions of the chosen rule ("best rule") r^m, according to equation (1), in the order obtained in advance and indicated by ctx. Recall that the graph indicating the precedence constraints for conditions of each rule is a forest, so the result must be the state after achieving some leaf $y_0 \in Z_r^L$ (the leaf is the last one to achieve) in its respective context state, denoted as $s[s^{tc}(y_0, r, s)]$. However, there may be several leaves in the forest, which one we should pick up? Since they have no order constraints to each other, we can arbitrarily pick one up and treat it as s' in equation (1). The other is that incorporating the changes made by r^m itself. This can be solved by capturing its "side effects" as in h^{cea} (Helmert and Geffner, 2008). Therefore, the final state corresponding to x is $s[s'][r^m]$.

We embed the h^{tpcc} into the best-first-search algorithm, and use it to guide the search process. For every time stamped state met in search, we using the h^{tpcc} to evaluate the cost to the goals, and choose the state having smallest heuristic cost as the expanded state and to generate a new search state till the goals are achieved.

Obtaining Temporal Precedence Constraints

Recently, landmarks attract a large number of researchers, and it is well known that the current heuristic estimators based on them works well in the planning, eg the inadmissible heuristic h^{LM} in the LAMA planner (Richter, Helmert and Westphal, 2008), the admissible heuristics h^{LA}, h^{L} and h^{LM-cut} (Helmert and Domshlak, 2009). For a given propositional planning task, Porteous, Sebastia and

Hoffmann (2001) define that landmarks are propositions that must be true at some point in every solution plan. Also, they propose an algorithm called LM^{RPG} to extract landmarks and their orderings from the relaxed planning graph of a planning task. Richter, Helmert and Westphal (2008) firstly derive landmarks to *SAS+* tasks and propose a landmark-based heuristic h^{LM} mentioned above. Herein, we derive landmarks to temporal and numeric planning task, and apply them to generate the precedence constraints over rule conditions.

Definition 1 landmark

Let $\prod = < V, s_0, s_*, A, O >$ be a temporal planning task.

A logical landmark is an atom associate logical variable that must be true at some point in every plan of $\prod$.

A comparison landmark is a atom associating comparison variable that decided by the numeric expression which must hold at some point in every plan of $\prod$.

The logical landmark is as same to that in LAMA. While for explaining the comparison landmark, we return back to the example mentioned above. For achieve the goal that flower f_1 is watered until its level $n_1 = 10$, it's easy to know that the numeric expression $w - (n_1 - h_1) \geq 0$ as the condition of the action water f_1 must hold at some point in every plan. Patrick et. al introduce auxiliary variables to represent the numeric condition, see Fig. 1. aux_3 is a comparison variable, the figure visualize the comparison axiom of aux_3 and also show the dependency graph of aux_3 and numeric variables aux_2, aux_1. $aux_3 = aux_2 \geq 0$ express the condition from the angle of comparison variable and $aux_2 \geq 0$ must be true at some point in every plan. Hence $aux_3 = true$ is a comparison landmark.

The precondition $w - (n_1 - h_1) \geq 0$ must holds.

$aux_3 = aux_2 \geq 0$

$aux_2 = w - aux_1$

$aux_1 = n_1 - h_1$

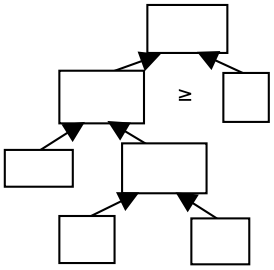

Figure 1: visualization of numeric and comparison axioms.

How do we apply the landmarks to generate the precedence constraints for temporal planning task? Firstly, we build a landmark graph in which nodes indicate landmarks and arcs indicate orderings between landmarks (to guarantee landmark graph acyclic, possible cycles are broken). Secondly, according to the landmark graph adding some rules, we extract the temporal precedence constraints and explicate them as context functions.

To build the landmark graph, we borrow the methods from Hoffmann et .al (2003) and Richer et .al (2008).Following is the briefly explain, and the details refer to the original paper: 1. Set the goals as the initial landmarks, backtrack the relaxed graph to get other landmarks and orderings (If A is a landmark in step i of relaxed graph, and the operators in step i-1 that achieve A has a common precondition B, then B is a candidate landmark which is ordered before A). In addition, one-step lookahead and domain transition graph are applied to find further landmarks. Disjunctive landmarks are considered too 2. A sufficient criterion is applied to eliminate non-landmarks: each fact A is tested by removing all achievers of A from the original task, and then check whether the task is still relaxed solvable (since Hoffmann et al. (2003) show that deciding if a fact is a landmark and deciding orderings between facts are PSPACE-complete, the detection of landmarks and orderings is approximate, so there is no guarantee that the generated landmarks are sound). 3. Further orderings is introduced. For two landmarks A and B, if achieving A must delete B, or if achieving A can decrease the cost of achieving B, then we add the order $A \rightarrow B$.

To obtain precedence constraints, we consider each rule r appearing in the heuristic cost computation and extract ctx for Z_r by the following several rules:

Rule 1 $p, q \in Z_r$ and p, q are landmarks, if p is directly orders before q in landmark graph, then $ctx(q, r) = p$.

Rule 2 $p, q \in Z_r$, p is a comparison fact and q is a logical fact. If $\exists r \in prom(p, s) \wedge \exists x \in Z_r \wedge \text{var}(x) = \text{var}(q)$, then $p <_d q$ (p directly orders before q).

According to Rule 1, we traverse the topology order for Z_r from Landmark Graph, if $p \in Z_r$ is directly order before $q \in Z_r$, then $ctx(q, r) = p$. But if there are more than one facts (in Z_r) directly order before q, we arbitrarily choose one. However, how to embed the comparison facts into the topology order? Rule 2 is applied to solve this question for obtaining more information about temporal precedence constraints. To explain this, we consider the example of watering f_1. $r: f_1\_watered = 0, \text{var}\_p = l_1, aux_3 = 1 \rightarrow f_1\_watered = 1$ is chosen to be the "best rule" (transformed from the propositional form $r: f_1 = unwatered, at(l_1), w - (n_1 - h_1) \geq 0 \rightarrow water\ f_1$, the detail of transforming is referred to Helmert (2004)) and the current state is $s = \{\text{var}\_p = l, aux_3 = 0, f_1\_watered = 0, w = 0, h_1 = 0\}$. We can know that $fill\ tank_{l_0}$ (fill the water tank of robot at location l_0) is an action that can change the value of aux_3 since it can increase w, so $r': \text{var}\_p = l \rightarrow w = 150 \in prom(aux_3 = 1, s)$. Supposing we set $\text{var}\_p = l_1 < aux_3 = 1$, when achieve $aux_3 = 1$, $\text{var}\_p = l_1$ is deleted due to $\text{var}\_p = l$'s addition, and according to Koehler and Hoffmann's (2000) goal orderings (if achieve A should delete B, then A < B), $aux_3 = 1 < \text{var}\_p = l_1$ holds, contradictory to our suppose. Hence, we should set $aux_3 = 1 <_d \text{var}\_p = l_1$.

As we known that when we detect the rules to obtain the "best rule" in equation (4), we will meet some facts that are not in landmark graph. That's mean not all the conditions of actions are landmarks. How do we capture the orderings between those facts? We draw on the previous works on goal orderings (Koehler, Hoffmann, 2000). For logical

facts we just directly follow the methods in Koehler and Hoffmann, and leave comparison facts to Rule 2 for simple.

Implementation and Discussion

We implement h^{tpcc} on top of the code of TFD (Eyerich et al, 2009) and LAMA (Richter et al. 2008). Additionally, we consider the estimated cost of comparison variable just like that in h^{tcea} for simple. To evaluate it, we test some benchmarks used in the temporal satisficing track of IPC 2008 and obtain some primary results. From the primary results, we can see the potential power of h^{tpcc} on the improving of solutions, although it works worse on several benchmarks, which is mostly due to our currently very rough implementation. Therefore, we consider h^{tpcc} as a promising heuristic and will improve and perfect our implementation in the future work.

References

Chih-wei, Hsu., and Benjamin W, Wah. 2008. *The SGPlan planning system in IPC-6*. http://manip.crhc.uiuc.edu/Wah/papers/C168/C168.pdf.

Dunbo, C., Hoffmann, J., and Helmert, M. 2009. *Enhancing the context-enhanced additive heuristic with precedence constraints*. In Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS 2009), Thessaloniki, Greece, September 19-23 2009.

Helmert, M. 2004. *The fast Downward planning system*. http://www.informatik.uni-freiburg.de/~helmert/pub.

Helmert, M., and Geffner, H. 2008. *Unifying the causal graph and additive heuristics*. In Proc. ICAPS 2008:140-147.

Helmert, M., and Domshlak, C. 2009. *Landmarks, Critical Paths and Abstractions: What's the Difference Anyway?* 19th International Conference on Automated Planning and Scheduling (ICAPS 2009), Thessaloniki, Greece, September 19-23 2009.

Hoffmann, J., Porteous, J., and Sebastia, L. 2003. *Ordered landmarks in planning*. Journal of Artificial Intelligence Research 22:215–278.

Koehler, J., and Hoffmann, J. 2000. *On reasonable and forced goal orderings and their use in an agenda-driven planning algorithm*. JAIR 12:338–386.

Patrick, E., Robert, M., and Gabriele, R. 2009. *Using the context-enhanced additive heuristic for temporal and numeric planning*. In Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS 2009):130-137. AAAI Press 2009.

Porteous, J.; Sebastia, L.; and Hoffmann, J. 2001. *On the extraction, ordering, and usage of landmarks in planning*. In Cesta, A., and Borrajo, D., eds., Pre-proceedings of the Sixth European Conference on Planning (ECP'01), 37–48.

Sebastia, L.; Onaindia, E.; andMarzal, E. 2006. Decomposition of planning problems. AI Communications 19(1):49–81.

Richter, S., Helmert, M., and Westphal, M. 2008. *Landmarks Revisited*. Proceedings of the Twenty-Third AAAI conference on Artificial Intelligence.