On the Discovery and Utility of Precedence Constraints in Temporal Planning

Yanmei Hu[1], Minghao Yin[1*], Dunbo Cai[2]

[1]School of Computer Science and Information Technology, Northeast Normal University, Changchun, P. R. China, 130117
[2]School of Computer Science and Engineering, Wuhan Institute of Technology, Changchun, P. R. China, 430022
huym260@gmail.com, ymh@nenu.edu.cn, dunbocai@gmail.com

Introduction

Temporal planning considers temporal dependencies and numeric resources. As a further step towards real-world applications, it has been attracting many attentions and triggered some great work. Recently, most of those works are based on heuristic search, e.g., in the work of Sapa (Do and Kambhampati 2003), LPG (Gerevini et al. 2008), SGP (Chih-wei et al. 2008) and TFD (Eyerich et al. 2009). In this line, the design of reasonable and informative heuristics shows a great progress, especially in the planner TFD.

TFD proposed an extension of the *SAS+* formulism, which is called *temporal numeric SAS+*. The formulism basically has two advantages: leading to a smaller search space in comparison with the STRIPS representation, and building a convenient base for designing *causal graph* (Helmert 2006) based heuristics. Specifically, TFD proposed an extension of the heuristic h^{cea} (Helmert and Geffner 2008), which is potentially more informative than the causal graph heuristic h^{CG} (Helmert 2006) and can lead to plans of high quality. Noted that Cai et al. (2009) proposed an extension of h^{cea}, which is called h^{pcc}. h^{pcc} enhanced h^{cea} with the so called *precedence constraints*, and is potentially more informative than h^{cea}. So, a natural question is, in a temporal and numeric setting, how h^{pcc} can be generalized and what the result is. In this preliminary work, we make an attempt to answer the question and report some initial results.

Let's first consider an example where h^{tcea} could be improved. We follow the example used by Patrick et al (2009). Assume that there are two locations l_0 and l_1, a robot r_1 located at l_1 with a water tank whose capacity is $c = 150$ units and is initially empty, i.e., $w = 0$. Additionally, r_1 can only refill its tank at l_0 and there is a path between l_0 and l_1 with cost d_{01}. Now we want to water flower f_1 at location l_1. f_1 has a current water level $h_1 = 0$, and needs to be watered until level $n_1 = 10$. The rule for watering some flower f_i at l_i, have the form: f_i_*unwatered*, $at(l_i)$, $w - (n_i - h_i) \geq 0 \rightarrow f_i$_*watered*. Let's consider the cost of reaching f_0 = *watered*. To water f_0, $w - (n_0 - h_0) \geq 0$ must be fulfilled. So, the water tank must be filled with at least 10 units at l_0. As the pivot condition f_0_*unwatered* is satisfied in the current state, the estimated cost of reaching f_0_*watered* by h^{tcea} is $d_{01} + 10$. However, it is not a reasonable estimation. In fact, we should first achieve the condition $w - (n_0 - h_0) \geq 0$, and then achieve $at(l_1)$ in the context state satisfying $w - (n_0 - h_0) \geq 0$. Therefore, we can obtain a more precise estimation: $2 \times d_{01} + 10$.

Based on the work of Eyerich et al. (2009), we extend h^{pcc} (Cai et al. 2009) to obtain a more reasonable cost for temporal planning, which results a heuristic called *temporal precedence constrains contexts* (h^{tpcc}). In h^{tpcc} considers precedence constraints over both logical fluents and derived comparison variables. Note that Eyerich et al., (2009) only considers dependencies among fluents and the methods of Cai et al. (2009) only suit for fluents. However, in our setting, there may be comparison variables in the precondition of an action. For example, to model $w - (n_0 - h_0) \geq 0$, Eyerich et al. (2009) introduces a new boolean comparison variable v_c, where v_c is *true iff.* $w - (n_0 - h_0) \geq 0$ holds. To handle comparison variables, we proposed methods for accounting precedence constraints over both fluents and comparison variables.

Temporal Heuristic with Precedence Constraints

We follow the notation and definition of Eyerich et al. (2009) and Cai et al. (2009). For our purpose, a rule r corresponding to an *instant operator (io),* (Eyerich et al. 2009) has the form $r : Z_r \rightarrow x_r$ (or r: $Z_r \rightarrow \underline{e}_r$) with $cost(r)$

Partially supported by the NSFC grants (60803102 and 60973033).
Correspondence should be given to Minghao Yin (ymh@nenu.edu.cn)

= *cost*(*io*), where x_r is an atom associated with some logical variable and e_r is a numeric expression of the form $v_1 \circ v_2$. Z_r is a set of atoms that composed by logical variables or comparison variables.

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

In Eq. (1), we extend h^{pcc} (Cai et al. 2009) into a temporal and numeric setting using instant actions (Eyerich et al. 2009), which results the heuristic function h^{tpcc}. $h^{tpcc}(x|s)$ is the estimated cost of reaching an atom *x* from a state *s*. In Eq. (1), *s* is the state corresponding to the rule *r*, *s′* is the state corresponding to the rule *r′*, and $c^{tc}(y, r, s)$ is the context state associated with the condition *y* of *r*, with respect to *s*, which is computed like the equations (5) and (6) in the paper of Cai et al. (2009).

We design a *context function* (Cai et al. 2009) *ctx* to account the precedence constraints over preconditions of a rule *r*, where $ctx(r,q) = p$ indicates that the context of $q \in Z_r$ should be the state that results from achieving $p \in Z_r$. To obtain precedence constraints, we build the following rules:

Rule 1 For *p*, $q \in Z_r$, if *p* and *q* are landmarks, there are orderings $q \to p$, $q \to_{\mathrm{n}} p$ or $q \to_{\mathrm{gn}} p$, then *ctx*(*p*, *r*) = *q*.

Rule 2 For *p*, $q \in Z_r$, if *p* is a comparison atom and *q* is a logical atom, $\exists r' \in prom(p, s) \wedge \exists x' \in Z_{r'} \wedge var(x) = var(q)$, then *ctx*(*q*, *r*) = *p*.

Note that ***Rule 2*** is for accounting the precedence among comparison variables and logical variables. To consider the reasonability of ***Rule 2***, we may think cases where the value change of *p* involves the value change of *q*.

Experimental Results and Conclusions

To evaluate h^{tpcc}, we implement it on top of the code of Fast Downward (Helmert 2006) and LAMA (Richter et al. 2008), and test it on 12 benchmarks used in the temporal satisficing track of IPC 2008, with h^{tcea} as a reference. All experiments are done on a PC with a 2.4GHz CPU and 3GB memory. The limit on time is 30 miniutes and on memory is 2GB. Table 1 shows, for each heuristic, the number of solved problems on each domain. Table 2 compares the plan quality resulted from using h^{tpcc} and that resulted from using h^{tcea} on problems in each domain. In Tab. 2, +*n*/-*m* indicates that h^{tpcc} results better plans on *n* problems while results worse plans on *m* problems when compared with h^{tcea}.

From Tab. 1, we can see h^{tpcc} is worse than h^{tcea} totally, which is mostly due to our currently very rough implementation. From Tab. 2, we can see that h^{tpcc} leads to better plans than h^{tcea} dose on 6 domains. Therefore, we consider h^{tpcc} as a promising heuristic and will improve our implementation in the future work.

Domain	h^{tcea}	h^{tpcc}
Elevators-numeric	23	29
Elevators-strips	23	26
Crewplanning-strips	29	6
Openstacks-adl	30	30
Openstacks-numeric	30	30
Openstacks-numadl	30	30
Openstacks-strips	30	30
Parcprinter-strips	13	3
Pegsol-strips	29	30
Sokoban-strips	14	7
Transport-numeric	11	11
Woodwork-num	29	27
Total	291	259

Table 1: Number of instances solved with h^{tcea} and h^{tpcc}.

Domain	h^{tpcc} *vs.* h^{tcea}
Elevators-numeric	+11/-7
Elevators-strips	+7/-21
Crewplanning-strips	+0/-23
Openstacks-adl	+3/-7
Openstacks-numeric	+7/-0
Openstacks-numadl	+10/-10
Openstacks-strips	+1/-0
Parcprinter-strips	+0/-10
Pegsol-strips	+7/-4
Sokoban-strips	+1/-8
Transport-numeric	+5/-4
Woodwork-num	+8/-6

Table 2: Plan quality comparison.

References

Cai, D., Hoffmann, J., and Helmert, M. 2009. Enhancing the context-enhanced additive heuristic with precedence constraints. In *Proc. ICAPS'09*, 50-57.

Chih-wei, Hsu., and Benjamin W, Wah. 2008. The SGPlan planning system*.* In *IPC-6*.

Do, M. B., and Kambhampati, S. 2003. Sapa: A multi-objective metric temporal planner. *JAIR* 20:155–194.

Gerevini, A., Saetti, A., and Serina, I. 2008. An approach to efficient planning with numerical fluents and multi-criteria plan quality. *AIJ* 172(8-9):899–944.

Helmert, M. 2006. The Fast Downward planning system. *JAIR* 26:191–246.

Helmert, M., and Geffner, H. 2008. Unifying the causal graph and additive heuristics. In *Proc. ICAPS'08*, 140-147.

Eyerich, P.; Mattmuller, R.; and Gabriele, R. 2009. Using the context-enhanced additive heuristic for temporal and numeric planning. In *Proc. ICAPS'09*, 130-137.

Richter, S., Helmert, M., and Westphal, M. 2008. *Landmarks Revisited*. In *Proc. AAAI'08*, 975-982.