An Admissible HTN Planning Heuristic

Pascal Bercher, Gregor Behnke, Daniel Höller, Susanne Biundo

Institute of Artificial Intelligence, Ulm University, Germany

firstName.lastName@uni-ulm.de

Main Contributions

Our formalization bases upon *hybrid planning*, a hierarchical planning formalism that fuses *hierarchical task network (HTN) planning* with *partial order causal link (POCL) planning*.

Our main contributions are:

- Exploitation of the Task-Decomposition Graph (TDG) for pruning and heuristics. We introduce:
  - A cost-aware admissible heuristic
  - A modification-aware heuristic (which is admissible in the number of required modifications)
- TDG recomputation:
  - So far, all previous TDG-based techniques and heuristics were preprocessing heuristics.
  - We show that rebuilding the TDG can improve heuristic estimates and very often prunes the search space.

Definition (Plan)

A *plan* $P = (PS, \prec, CL, VC)$ is a partially ordered sequence of tasks:

- PS is a finite and possibly empty set of plan steps. Each plan step $l : t$ is a task t with a unique label l.
- $\prec \subseteq PS \times PS$ is a strict partial order on PS.
- $CL \subseteq PS \times V \times PS$, where V indicates the set of all positive and negative state variables, is a set of causal links between the plan steps. A causal link $l : t \rightarrow_\varphi l' : t'$ denotes that the precondition φ of the plan step $l' : t'$ is supported by the plan step $l : t$.
- The set VC is a set of variable constraints.

Definition (Planning Domain and Problem)

A *planning domain* is a tuple $\mathscr{D} = (T_a, T_p, M)$, where:

- T_a, T_p are finite sets of abstract and primitive tasks, respectively. Each (primitive or abstract) task is a triple $(t(\bar{\tau}), pre(\bar{\tau}), eff(\bar{\tau}))$ consisting of a parametrized name, and the task's preconditions and effects.
- M is a finite set of *(decomposition) methods*. A method $m = (t(\bar{\tau}), P)$ maps an abstract task $t(\bar{\tau}) \in T_a$ to a plan P.

A *planning problem* is a tuple $\mathscr{P} = (\mathscr{D}, s_i, P_i, g)$, where:

- $\mathscr{D}$ is the planning domain.
- s_i and g are the initial state and the goal description, respectively.
- P_i is the initial plan. As usual in POCL planning, it contains two special actions that encode s_i and g, respectively.

Definition (Solution Plan)

A plan P is a solution if and only if:

- P is a refinement P_i, i.e., P can be obtained from P_i via
  - Decomposition:
    given a plan $P' = (PS, \prec, CL, VC)$, use method $(t(\bar{\tau}), P'') \in M$ to replace $l : t(\bar{\tau}') \in PS$ by P''. Causal links and orderings are inherited.
  - Insertion of ordering constraints.
  - Insertion of causal links. Task insertion is prohibited.
- P is a solution in the standard POCL sense, i.e.,
  - it is primitive and ground,
  - there are no open preconditions, and
  - there are no causal threats

on the Difference on Hybrid and HTN planning

Hybrid problems are a generalization of HTN problems. That is, every HTN planning problem is also a hybrid problem:

- In HTN planning, *plans* are referred to as *task networks*. Task networks do not contain causal links. Note that this has an effect of the *problem description*, since plans are part of the model's decomposition methods.
- In HTN planning, *abstract tasks* are referred to as *compound tasks*. They do not have preconditions or effects.
- In HTN planning, solution task networks only need to *possess* an executable linearization. In hybrid planning, all linearizations need to be executable.

Our cost-aware heuristic is applicable to:

- HTN problems and hybrid problems, and
- for *all* search-based planners, both state-based progression planners and and decomposition-based planners.

Our modification-aware heuristic is applicable to:

- HTN problems and hybrid problems, and
- for hybrid (i.e., decomposition- and POCL-based) planners such as PANDA.

Search Algorithm PANDA

$F \leftarrow \{P_i\}$
while $F \neq \emptyset$ **do**
  $P \leftarrow$ `planSel` (F)
  if `Flaws`$(P) = \emptyset$ **then return** P
  $f \leftarrow$ `flawSel` (`Flaws`(P))
  $F \leftarrow (F \cup \{$ `modify`$(m, P) \mid m \in$ `Mods`$(f, P)\}) \setminus \{P\}$
return *fail*

TDG-based heuristics

Both proposed heuristics exploit the AND/OR structure of a TDG:

- The refinement effort of an abstract task can be estimated relying on its *cheapest* decomposition method.
- The refinement effort of a decomposition method can be estimated relying on the refinement efforts for *all* its tasks.

What is the refinement effort of a task? What do we want to estimate?

- The number of missing tasks or their costs (to estimate the size or cost of a solution plan) $\rightarrow$ cost-aware TDG heuristic
- The number or required plan modifications (to estimate the required search effort a planner still needs to perform) $\rightarrow$ modification-aware TDG heuristic

Cost-aware TDG heuristic

Prior to planning, we *once* pre-calculate a ground TDG and cost estimates for each of its nodes. Let $\langle V_T, V_M, E_{T\rightarrow M}, E_{M\rightarrow T}\rangle$ be a ground TDG. Then,

$$h_T(v_t) := \begin{cases} cost(v_t) & \text{if } v_t \text{ is primitive} \\ \min\limits_{(v_t, v_m) \in E_{T\rightarrow M}} h_M(v_m) & \text{else} \end{cases}$$

For a method vertex $v_m = \langle PS, \prec, CL, VC\rangle$, we set:

$$h_M(v_m) := \sum_{(v_m, v_t) \in E_{M\rightarrow T}} h_T(v_t)$$

During planning, let $P = \langle PS, \prec, CL, VC\rangle$ be a plan. Then,

$$h_{TDG_c}(P) := \sum_{\substack{l:t(\bar{\tau}) \in PS \\ t(\bar{\tau}) \text{ abstract}}} \left(\min_{v_t \in comp(t(\bar{\tau}))} h_T(v_t) \right)$$

Here, $comp(t(\bar{\tau}))$ is the set of all compatible groundings of the lifted abstract task $t(\bar{\tau})$ that are contained in the TDG.

Modification-aware TDG heuristic

Again, we pre-calculate a ground TDG $\langle V_T, V_M, E_{T\rightarrow M}, E_{M\rightarrow T}\rangle$ and modification estimates for its nodes.

$$h_T(v_t) := \begin{cases} |pre(v_t)| & \text{if } v_t \text{ is primitive} \\ 1 + \min\limits_{(v_t, v_m) \in E_{T\rightarrow M}} h_M(v_m) & \text{else} \end{cases}$$

For a method vertex $v_m = \langle PS, \prec, CL, VC\rangle$, we set:

$$h_M(v_m) := \sum_{(v_m, v_t) \in E_{M\rightarrow T}} h_T(v_t) - |CL|$$

Let $P = \langle PS, \prec, CL, VC\rangle$ be a plan. Then,

$$h_{TDG_m}(P) := \sum_{l:t(\bar{\tau}) \in PS} \left(\min_{v_t \in comp(t(\bar{\tau}))} h_T(v_t) \right) - |CL|$$

$comp(t(\bar{\tau}))$ is defined as above, and $pre(t(\bar{\tau}))$ is the precondition of the task $t(\bar{\tau})$.

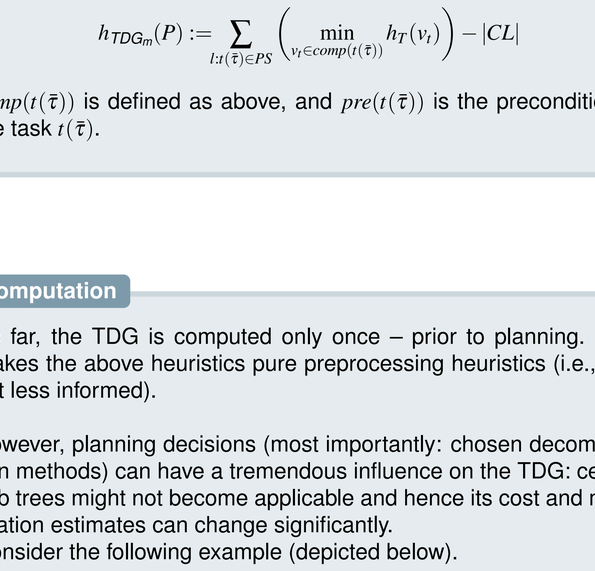
Recomputation

So far, the TDG is computed only once – prior to planning. This makes the above heuristics pure preprocessing heuristics (i.e., fast, but less informed).

However, planning decisions (most importantly: chosen decomposition methods) can have a tremendous influence on the TDG: certain sub trees might not become applicable and hence its cost and modification estimates can change significantly.
Consider the following example (depicted below).

Let $cost(p_3) = i$ and $h_M(P_{m_4}) = h_T(p_4) + h_T(a_3) = j > i$.
Then, we get $h_T(a_2) = i$. Let us now consider the heuristic values for P_1 and P_2 resulting from decomposing a_1 using m_1 or m_2, respectively.

Without recomputation, we get $h(P_1) = h(P_2) = i$. With recomputation, we get $h(P_1) = j$ and $h(P_2) = i$, so we get improved heuristic accuracy due to updated reachability information in the TDG.



Recomputation and Incremental Heuristic Computation

Let P be a plan and *mod* a modification, and P' the plan resulting from applying *mod* to P. In the case *mod* is a decomposition $m = (t, P_m)$ and there are also further methods for t, then we recompute the TDG. Otherwise, we perform an incremental heuristic calculation.

Cost-aware TDG heuristic:

1. *mod* is not a decomposition (i.e., an insertion of a causal link, an ordering, or a variable constraint). Then, we get:
   $h_{TDG_c}(P) = h_{TDG_c}(P')$
2. *mod* is a method $m = (t, P_m)$ (without alternatives). We can set:
   $h_{TDG_c}(P') = h_{TDG_c}(P) - \sum_{l:t' \in PS_m, t' \text{primitive}} cost(t')$

Modification-aware TDG heuristic:

1. *mod* is an ordering or variable insertion. Then, we get:
   $h_{TDG_m}(P) = h_{TDG_m}(P')$
2. *mod* is a causal link insertion or a decomposition (without alternatives). Then, we get:
   $h_{TDG_m}(P) = h_{TDG_m}(P') - 1$

Results (Coverage)

Table 1: *Per domain and strategy, we present the number of solved problem instances (#s), the number of optimally solved instances (#o), and the maximal plan cost over all solved problem instances relative to the optimal solution (cost).*

	Strategy	UM-Tr. (21 inst.) #s	#o	cost	SmartPh. (5 inst.) #s	#o	cost	Satellite (22 inst.) #s	#o	cost	Woodw. (11 inst.) #s	#o	cost	Summary (59 inst.) #s	#o	cost
blind	Uniform	21	21	1.00	4	4	1.00	17	17	1.00	8	8	1.00	50	50	**1.00**
	BF	21	21	1.00	4	4	1.00	15	15	1.00	7	7	1.00	47	47	**1.00**
	DF	21	21	1.00	5	1	1.60	19	7	2.09	8	4	1.44	53	33	2.09
systems	UMCP$_{BF}$	21	21	1.00	4	4	1.00	15	15	1.00	7	7	1.00	47	47	**1.00**
	UMCP$_{DF}$	21	21	1.00	4	1	1.60	17	6	2.09	6	4	1.29	48	32	2.09
	UMCP$_{h}$	21	21	1.00	5	4	1.40	19	11	1.50	7	7	1.00	52	43	1.50
	Compile	18	18	1.00	5	5	1.00	21	18	1.10	5	5	1.00	49	46	1.10
	Compile$_{opt}$	16	16	1.00	5	5	1.00	9	9	1.00	5	5	1.00	35	35	**1.00**
A*	ADD	21	21	1.00	4	1	1.20	21	21	1.00	10	9	1.17	56	52	1.20
	ADD-r	21	21	1.00	5	5	1.00	19	18	1.08	9	4	1.25	54	48	1.25
	Relax	21	21	1.00	5	5	1.00	18	18	1.00	10	8	1.17	54	52	1.17
	OC	21	21	1.00	4	4	1.00	21	21	1.00	10	7	1.17	56	53	1.17
	TDG$_m$/-rec	21	21	1.00	5	5	1.00	22	21	1.31	9	9	1.00	**57**	56	1.31
	TDG$_c$/-rec	21	21	1.00	5	5	1.00	18	18	1.00	8	8	1.00	52	52	**1.00**
A$^*_2$	ADD	21	21	1.00	4	0	1.20	21	20	1.09	10	9	1.17	56	50	1.20
	ADD-r	21	21	1.00	5	5	1.00	20	17	1.10	10	4	1.25	56	47	1.25
	Relax	21	21	1.00	5	5	1.00	18	15	1.10	10	4	1.25	54	45	1.25
	OC	21	21	1.00	4	4	1.00	22	21	1.09	10	7	1.22	**57**	53	1.22
	TDG$_m$/-rec	21	21	1.00	5	5	1.00	22	17	1.31	9	8	1.08	**57**	51	1.31
	TDG$_c$	21	21	1.00	5	5	1.00	20	20	1.00	10	10	1.00	56	56	**1.00**
	TDG$_c$-rec	21	21	1.00	5	5	1.00	20	20	1.00	11	11	1.00	**57**	**57**	**1.00**

Results (Recomputation)

Table 2: *Per domain, we present the number of recomputations divided by number of decompositions (*rec/dec*) and the number of improved heuristic estimates divided by number of recomputations (*h-im/rec*). For each of these values we report the minimum (*min*), maximum (*max*), and mean of means (μ).*

	Strategy	rec/dec min	rec/dec max	rec/dec μ	h-im/rec min	h-im/rec max	h-im/rec μ	rec/dec min	rec/dec max	rec/dec μ	h-im/rec min	h-im/rec max	h-im/rec μ
		UM-Translog						SmartPhone					
A*	TDG$_m$	.027	.188	.086	.000	.333	.032	.300	.691	.476	.000	.117	.023
	TDG$_c$	.027	.188	.086	.000	.333	.032	.300	.647	.473	.000	.041	.008
A$^*_{2'}$	TDG$_m$	.027	.188	.086	.000	.333	.032	.300	.713	.484	.000	.121	.024
	TDG$_c$	.027	.188	.086	.000	.333	.032	.300	.647	.471	.000	.041	.008
		Satellite						Woodworking					
A*	TDG$_m$	.857	1.00	.956	.110	.608	.248	.294	.932	.581	.000	.548	.246
	TDG$_c$	.750	1.00	.913	.087	.592	.264	.294	.943	.600	.000	.592	.330
A$^*_{2'}$	TDG$_m$	.857	1.00	.953	.110	.617	.268	.294	.961	.611	.000	.721	.306
	TDG$_c$	.814	1.00	.934	.049	.609	.256	.294	.943	.615	.000	.587	.333

Table 3: *For each domain, we summarize in how many problem instances the search space or time was deduced (<), unchanged (=), or increased (>), due to TDG recomputation.*

Strategy	space <	space =	space >	time <	time =	time >	space <	space =	space >	time <	time =	time >	space <	space =	space >	time <	time =	time >
	UM-Translog						SmartPhone						Satellite					
A* TDG$_m$	2	19	0	1	15	5	1	4	0	1	4	0	22	0	0	0	17	5
A* TDG$_c$	2	19	0	0	13	8	1	4	0	0	4	1	18	0	0	0	10	8
A$^*_{2'}$ TDG$_m$	2	19	0	3	16	2	1	4	0	0	4	1	22	0	0	0	13	9
A$^*_{2'}$ TDG$_c$	2	19	0	4	11	6	1	4	0	1	4	0	20	0	0	0	11	9

Strategy	time <	time =	time >	space <	space =	space >	time <	time =	time >
	Woodworking								
A* TDG$_m$	7	2	0	1	6	2			
A* TDG$_c$	6	2	0	4	3	1			
A$^*_{2'}$ TDG$_m$	4	5	0	1	6	2			
A$^*_{2'}$ TDG$_c$	5	5	0	4	3	3			

Conclusion and Future Work

- To the best of our knowledge, TDG$_c$ is the first domain-independent heuristic for standard HTN planning.
- TDG$_c$ is admissible and thereby guarantees finding optimal solutions in combination with A^* (cf. evaluation).
- Both heuristics perform well in terms of plan quality and search guidance.
- Rebuilding the TDG often reduces the search space (see Tab. 3), sometimes significantly. However, many recomputations do not improve heuristic accuracy (see Tab. 2). So, in future work, we want to identify more unnecessary recomputations.
- Due to the partial order of plans (in search nodes and in the TDG), it is non-trivial to generate a relaxed plan. Instead, our heuristics calculate the cheapest set of actions that can be reached from a given search node while ruling out unreachable actions. Computing relaxed plans remains future work.