Hybrid Planning
Theoretical Foundations and Practical Applications

Pascal Bercher

Supervisor: Susanne Biundo

Institute of Artificial Intelligence, Ulm University, Germany

firstName.lastName@uni-ulm.de

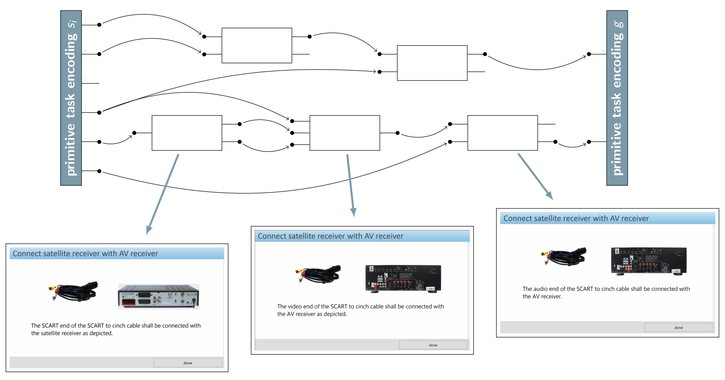
Main Contributions

- A set-theoretic formalization of HTN/TIHTN and hybrid planning.
- A comprehensive overview of the complexity results for the plan existence problem for these problem classes and sub problems thereof, such as Partial-Order Causal-Link (POCL) planning.
- A plan-space-based hybrid search algorithm suitable for all presented problem classes, called PANDA$_2$.
- Informed heuristics for hybrid planning in general, and POCL planning in particular. Some of these heuristics are the first admissible heuristics for the respective problem class.
- The results are applied in an assistance system that helps a user in the task of setting up a home theater.

Home Theater

We implemented a prototypical companion system that assists a human user in the task of setting up a complex home theater. For that purpose, the sub devices must be connected with each other using the correct cables and adapters. The assistance system has the following capabilities:

- Its instructions are based upon a plan that is automatically generated from a formal description of the hardware. The instructions explain how to set up the theater.
- Based on plan repair, it can explain any instruction in question.
- The system can cope with execution errors (broken cables).



Definition (Plan)

A *plan* $P = (PS, \prec, CL)$ 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 $l : t$.

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 4-tuple $(prec^+, prec^-, eff^+, eff^-)$ consisting of positive and negative preconditions and effects over the set of state variables.
- M is a finite set of *(decomposition) methods*. A method $m = (t, P)$ maps an abstract task $t \in T_a$ 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 iff:

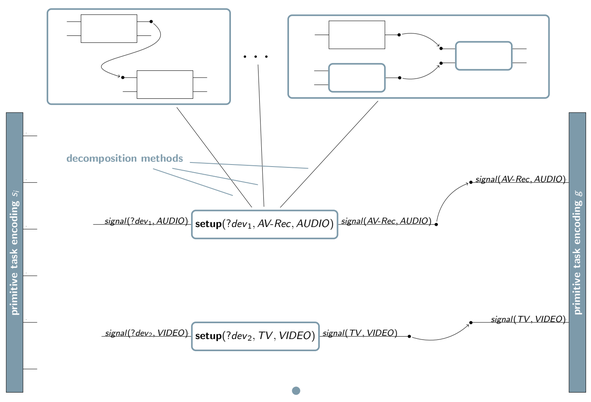
- P is a refinement P_i, i.e., P can be obtained from P_i via
  - Decomposition: given a plan $P' = (PS, \prec, CL)$, use method $(t, P'') \in M$ to replace $l : t \in PS$ by P''. Causal links and orderings are inherited.
  - Insertion of ordering constraints.
  - Insertion of tasks. *This refinement is optional!*
  - Insertion of causal links.
- Every linearization of P is executable in s_i and satisfies g.

In the absence of causal links, the respective problem classes are called HTN planning or TIHTN planning (HTN planning with task insertion), depending on whether insertion of tasks is allowed.

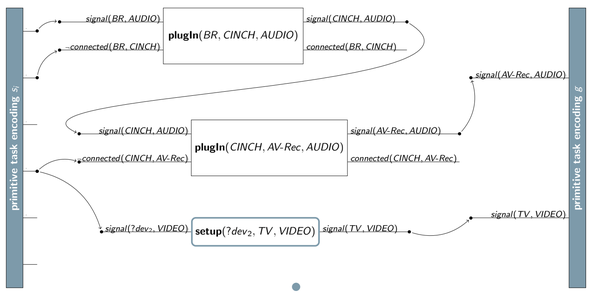
Search Algorithm PANDA

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

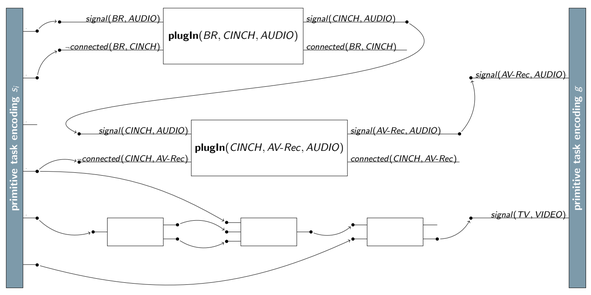
initial plan $\mathbf{P_i}$
(the graphic also shows some decomposition methods)



intermediate plan



solution plan



Complexity Results

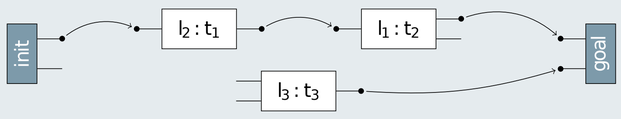
Restrictions on Hierarchy	Task Networks Totally Ordered?	Task Insertion?	Complexity Class
Arbitrary	yes	yes	PSPACE
Arbitrary	yes	no	EXPTIME
Arbitrary	no	yes	NEXPTIME
Arbitrary	no	no	undecidable and semi-decidable
Tail-Recursive	yes	yes	PSPACE
Tail-Recursive	yes	no	PSPACE
Tail-Recursive	no	yes	NEXPTIME
Tail-Recursive	no	no	EXPSPACE
Regular	yes	yes	PSPACE
Regular	yes	no	PSPACE
Regular	no	yes	PSPACE
Regular	no	no	PSPACE
Acyclic	yes	yes	PSPACE
Acyclic	yes	no	PSPACE
Acyclic	no	yes	NEXPTIME
Acyclic	no	no	NEXPTIME
No Hierarchy	yes	yes	PSPACE
No Hierarchy	yes	no	in P
No Hierarchy	no	yes	PSPACE
No Hierarchy	no	no	NP

All results are for HTN and TIHTN planning (i.e., where there are no causal links).

POCL Heuristic based on Partial Delete-Relaxation

Idea: perform delete-relaxation only for the domain, but let the given plan unaltered, as it describes the current planning process.

- The given plan contains valuable information, such as the the negative effects of the actions and the causal links.
- However, finding a delete-relaxed solution for it is NP-complete due to the partial order. Thus, we sample n (fixed) linearizations.



For each linearization, solve the respective sub problems using the FF heuristic. Here, there are four sub problems per linearization. In each sub problem, the set of applicable actions may be different because of the causal links than "span over the respective problems".

POCL Heuristic based on Partial Delete-Relaxation – Results

Evaluation was done using benchmarks from IPC 1 to 5. We compared Sample FF with the Add and the Relax Heuristic.

- Number of solved problem instances in 15 minutes (of a total of 446). Add: 292, Relax: 194, Sample FF: between 127 and 187.
- Sample FF was the only heuristic that was able to prove the unsolvability of an unsolvable planning instance.
- For the most informed variant of Sample FF, there is no relaxed solution among 30 samples for 36 % of all search nodes.

State-based heuristics for POCL planning

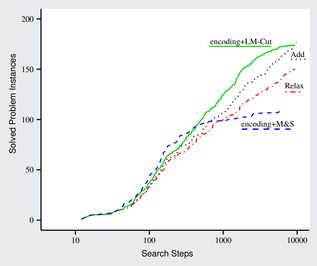
Idea: encode a plan into a state. Then, use state-based heuristics!

- $T'_p := T_p \cup \{enc(l_1 : t_1), enc(l_2 : t_2)\}$ with
  - $enc(l_1 : t_1) = \langle prec^+_{t_1} \cup \{l_2\}, prec^-_{t_1} \cup \{l_1\}, eff^+_{t_1} \cup \{l_1\}, eff^-_{t_1} \rangle$
  - $enc(l_2 : t_2) = \langle prec^+_{t_2}, prec^-_{t_2} \cup \{l_2\}, eff^+_{t_2} \cup \{l_2\}, eff^-_{t_2} \rangle$
- $s'_i := s_i$ and $g' := g \cup \{l_1, l_2\}$

Here, causal links are not compiled away, but this can also be done in P.

Then, $h_{\mathscr{D}}(P)$ can be defined via $h_{\mathscr{D}'}(s_i)$.

State-based heuristics for POCL planning – Results



- The encoding plus an admissible state-based heuristic constitutes the first admissible POCL heuristic.
- The encoding plus LM-Cut acts clearly more informed than the Add and the Relax heuristics. This is quite interesting, since LM-Cut is admissible whereas Add and Relax are not.

MME Heuristic

The MME (Minimal Modification Effort) heuristic relies on a TDG (Task Decomposition Graph). An example TDG is depicted below.

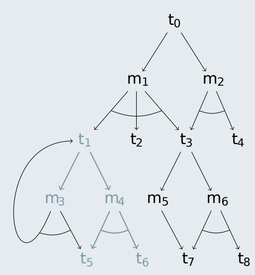
For task nodes $n_t = \langle prec^+, prec^-, eff^+, eff^- \rangle$, $h_T(n_t)$ is given by

$$h_T(n_t) := \begin{cases} |prec^+(n_t)| + |prec^-(n_t)| & \text{if } n_t \text{ primitive} \\ 1 + \min\limits_{(n_t, n_m) \in E_{T \rightarrow M}} h_M(n_m) & \text{else} \end{cases}$$

For method nodes $n_m = (m, P)$ with $P = (PS, \prec, CL)$, $h_M(n_m)$ is given by

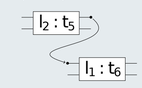
$$h_M(n_m) := \sum_{(n_m, n_t) \in E_{M \rightarrow T}} h_T(n_t) - |CL|$$

Example:



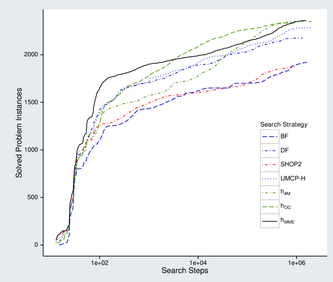
$h_T(t_1) = 1 + min\{h_M(m_3), h_M(m_4)\}$

let $m_4 = (t_1, P)$ with P given by



$$\begin{aligned} h_M(m_4) &= h(t_5) + h(t_6) - |CL| \\ &= |prec^+(t_5)| + |prec^-(t_5)| + \\ &= |prec^+(t_6)| + |prec^-(t_6)| - 1 \\ &= (1+1) + (2+0) - 1 = 3 \end{aligned}$$

MME Heuristic – Results



- MME is admissible with respect to the number of required modifications.
- It acts more informed than most other hybrid or hierarchical planning heuristics.
- It can be easily adapted to take action costs into account.

References

[1] Ron Alford, Pascal Bercher, and David Aha. Tight bounds for HTN planning. In *ICAPS 2015*. AAAI Press.

[2] Ron Alford, Pascal Bercher, and David Aha. Tight bounds for HTN planning with task insertion. In *IJCAI 2015*. AAAI Press.

[3] Gregor Behnke, Denis Ponomaryov, Marvin Schiller, Pascal Bercher, Florian Nothdurft, Birte Glimm, and Susanne Biundo. Coherence across components in cognitive systems: One ontology to rule them all. In *IJCAI 2015*. AAAI Press.

[4] Pascal Bercher, Susanne Biundo, Thomas Geier, Thilo Hoernle, Florian Nothdurft, Felix Richter, and Bernd Schattenberg. Plan, repair, execute, explain - how planning helps to assemble your home theater. In *ICAPS 2014*, pages 386–394. AAAI Press.

[5] Pascal Bercher, Thomas Geier, and Susanne Biundo. Using state-based planning heuristics for partial-order causal-link planning. In *KI 2013*, pages 1–12. Springer.

[6] Pascal Bercher, Thomas Geier, Felix Richter, and Susanne Biundo. On delete relaxation in partial-order causal-link planning. In *ICTAI 2013*, pages 674–681. IEEE Computer Society.

[7] Pascal Bercher, Shawn Keen, and Susanne Biundo. Hybrid planning heuristics based on task decomposition graphs. In *SoCS 2014*, pages 35–43. AAAI Press.

[8] Pascal Bercher, Felix Richter, Thilo Hörnle, Thomas Geier, Daniel Höller, Gregor Behnke, Florian Nothdurft, Frank Honold, Wolfgang Minker, Michael Weber, and Susanne Biundo. A planning-based assistance system for setting up a home theater. In *AAAI 2015*, pages 4264–4265. AAAI Press.

[9] Mohamed Elkawkagy, Pascal Bercher, Bernd Schattenberg, and Susanne Biundo. Improving hierarchical planning performance by the use of landmarks. In *AAAI 2012*, pages 1763–1769. AAAI Press.

[10] Thomas Geier and Pascal Bercher. On the decidability of HTN planning with task insertion. In *IJCAI 2011*, pages 1955–1961. AAAI Press.

[11] Daniel Höller, Gregor Behnke, Pascal Bercher, and Susanne Biundo. Language classification of hierarchical planning problems. In *ECAI 2014*, volume 263, pages 447–452. IOS Press.