# On the Verification of Totally-Ordered HTN Plans

Roman Barták,Simona Ondrčková, Gregor Behnke, Pascal Bercher

## On the Verification of Totally-Ordered HTN Plans

Improved version of plan verification algorithm that solves totally ordered domains faster.
Parsing based bottom up approach.

This approach supports:

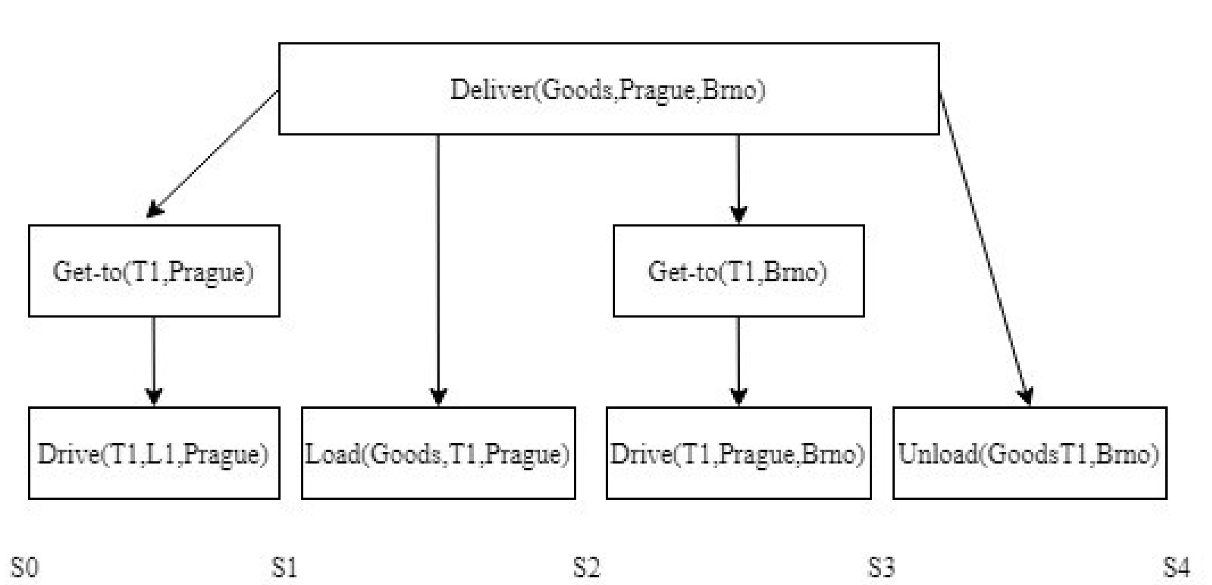
- Empty tasks
- Before, between conditions
- Recursive tasks
- Ordering conditions

Decomposition rules:

Deliver(G,L1,L2) ->Get-to(T,L1), Load(G,T,L1), Get-to(T,L2), Unload(G,T,L2)
conditions: get-to1<load, load<get-to2, get-to2<unload

Get-to(T,L1) ->Drive(T,L0,L1)
conditions: at(T,L0)

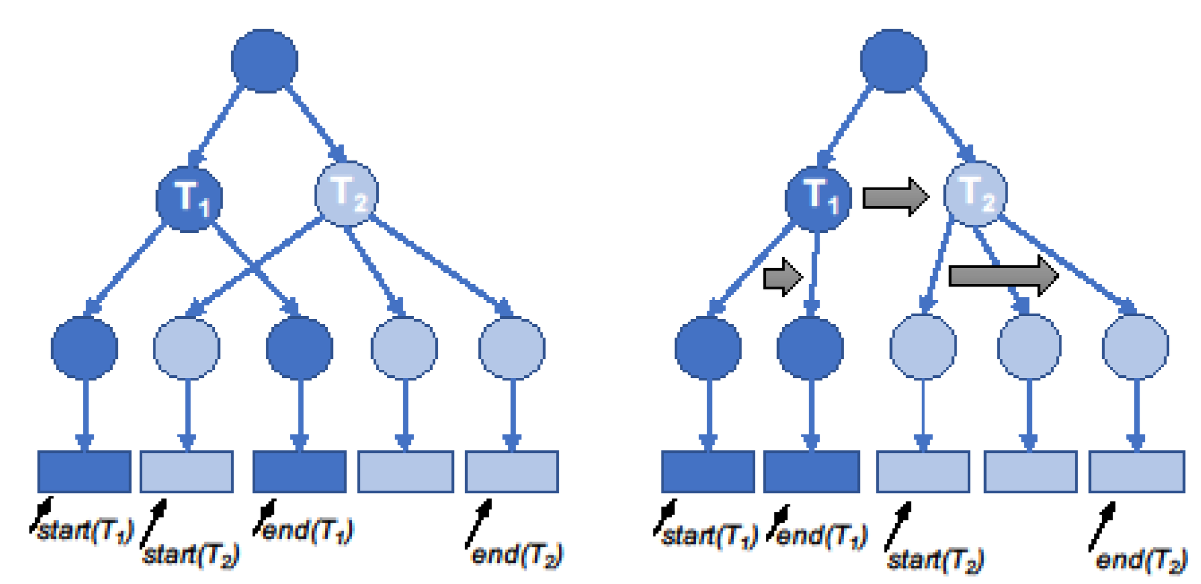
Get-to(T,L1) -> (empty task)
conditions: at(T,L1)




Deliver(Goods,Prague,Brno)
Action vector: 1,1,1,1
Start index: 1
End index: 4

## Improvement

If a domain is TO, it doesn't allow interleaving.
Subtasks must form a contigous sequence.

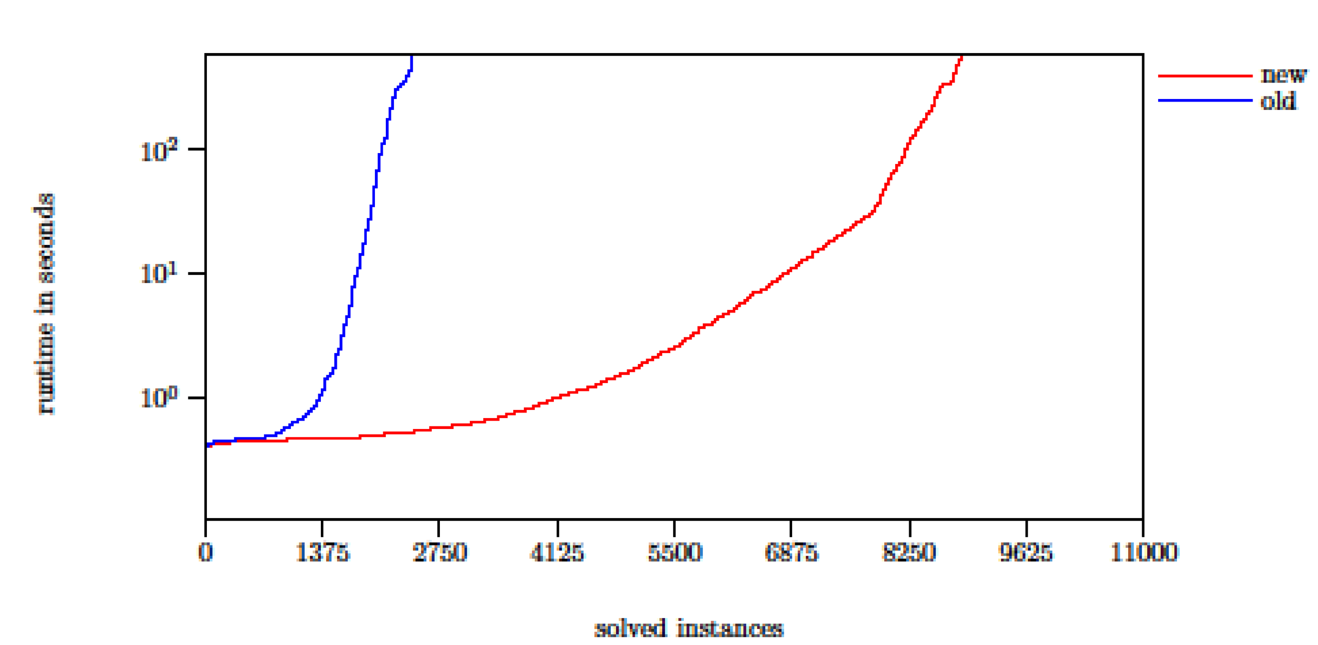



Main improvement:

$$end(T_i) + 1 = start(T_{i+1})$$

## Evaluation

Tested on 10963 domains.
Time limit 10 minutes.
Average plan length 239 actions.
Red – new algorithm
Blue – old algorithm




## Algorithm

**Data:** a plan $\mathbf{P} = (a_1, ..., a_n)$, an initial state $S_0$, and a set of decomposition methods (domain model); $TO = true$ if the domain is totally ordered,
**Result:** `true` if the plan can be derived from some compound task, `false` otherwise

**Function** VERIFYPLAN
  **for** $i$ = *1* **to** $n$ **do**
    **if** $\neg(\text{pre}(a_i) \subseteq S_{i-1})$ **then**
      **return false**
    $S_i = (S_{i-1} \setminus \text{eff}^-(a_i)) \cup \text{eff}^+(a_i)$
  **sp** $\leftarrow \emptyset$; new $\leftarrow \{(A_i, i, i, I_i) \mid i \in 1..n\}$
  **Data:** $A_i$ is a primitive task corresponding to action $a_i$, $I_i$ is a Boolean vector of size $n$, such that $\forall i \in 1..n, I_i(i) = 1$, $\forall j \neq i, I_i(j) = 0$
  **while** new $\neq \emptyset$ **do**
    **sp** $\leftarrow$ **sp** $\cup$ new; new $\leftarrow \emptyset$
    **foreach** *decomposition method* $R$ *of the form* $T_0 \rightarrow T_1, ..., T_k\ [\prec, \text{bef}, \text{btw}]$ *such that* $\{(T_j, b_j, e_j, I_j) \mid j \in 1..k\} \subseteq$ **sp do**
      **if** $\exists (i,j) \in \prec : \neg(e_i < b_j)$ **then**
        **continue with the next method**
      **if** $TO \wedge \exists i : \neg(e_i + 1 = b_{i+1})$ **then**
        **continue with the next method**
      $b_0 \leftarrow \min\{b_j \mid j \in 1..k\}$
      $e_0 \leftarrow \max\{e_j \mid j \in 1..k\}$
      **for** $i$ = *1* **to** $n$ **do**
        $I_0(i) \leftarrow \sum_{j=1}^{k} I_j(i)$;
        **if** $I_0(i) > 1$ **then**
          **continue with the next method**
      **if** $\exists (p, U) \in \text{bef} : p \notin S_{\min\{b_j \mid j \in U\}-1}$ **then**
        **continue with the next method**
      **if** $\exists (U, p, V) \in \text{btw}\ \exists i \in \max\{e_j \mid j \in U\}, \ldots, \min\{b_j \mid j \in V\} - 1 : p \notin S_i$ **then**
        **continue with the next method**
      new $\leftarrow$ new $\cup \{(T_0, b_0, e_0, I_0)\}$
      **if** $\forall k, I_0(k) = 1$ **then**
        **return true**
  **return false**

**Algorithm 1:** Parsing-based HTN plan verification

**This is a small extension of the algorithm but a giant improvement of the efficiency.**