Loop Detection in the PANDA Planning System

Daniel Höller[1] and **Gregor Behnke**[2]

[1]Saarland University, Saarland Informatics Campus, Saarbrücken, Germany
[2]University of Freiburg, Freiburg, Germany
hoeller@cs.uni-saarland.de, behnkeg@informatik.uni-freiburg.de

Loop Detection in HTN Planning

- HyperTension, the winner of the track on *Totally-Ordered HTN Planning* of the 2020 IPC comes with an interesting search technique
  - it tracks parts of the decomposition path in the state of the problem and
  - uses this information to leave decomposition structures that otherwise would have resulted in an infinite loop of its depth first search
- This can be seen as a preliminary form of loop detection
- In classical planning, most (search-based) planning systems use a loop detection/a graph search
- In HTN planning, this is computationally costly in the general case

Loop Detection in HTN Planning

- Search nodes contain a state *and* a task network
- We need to compare both to identify duplicates

→ Here we focus on task networks

- It has been shown that checking whether task networks are isomorphic is as hard as graph isomorphism

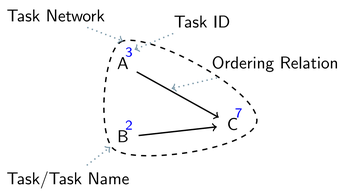

Figure 1: *Schema of task networks.*

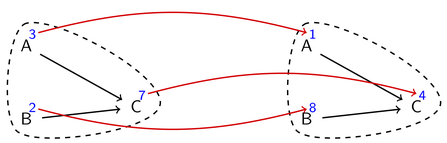

Figure 2: *ID mapping between isomorphic tasks.*

Loop Detection in HTN Planning

- We integrate loop detection into the progression search of the PANDA system
- We integrate it in the mechanics of the search, obtaining a graph search

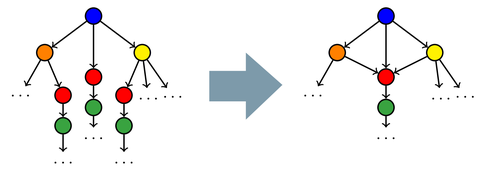

Figure 3: *From tree search to graph search.*

Loop Detection in HTN Planning

- To make the comparison feasible, we
  - introduce approximations and
  - exploit special cases

Task Hash Approximation

- Determine for each task how often it is in the task network
- Compare or hash the resulting table
- Non-isomorphic task networks might have same table and same hash

→ Search becomes incomplete

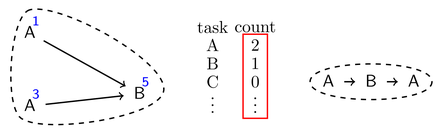

Figure 4: *Task Hash approximation.*

Task Layers Approximation

- Determine layers of tasks
- Sort and concatenate the layers
- Non-isomorphic task networks might have same layering and same hash

→ Search becomes incomplete

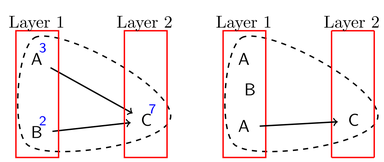

Figure 5: *Task Layers approximation.*

Ordering Relations Approximation

- Determine pairs of ordering relations
- Sort and concatenate them
- Non-isomorphic task networks might have same pairs and same hash

→ Search becomes incomplete

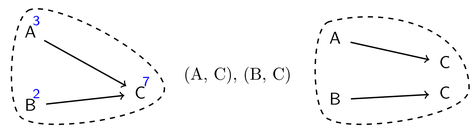

Figure 6: *Ordering Relations approximation.*

Exact Test for Totally Ordered Problems

- For totally ordered problems, we can compare the task sequences
- Results in an exact isomorphism test

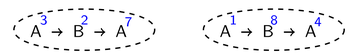

Figure 7: *Exact test for totally ordered problems.*

Exact Test for Partially Ordered Problems (1)

- For partially ordered problems, we can compare the task networks
- Results in an exact isomorphism test
- Is hard to compute → first hash, compute only for hash bucket

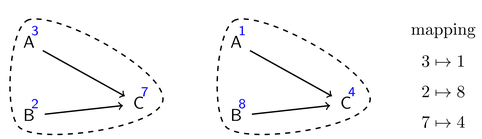

Figure 8: *Exact test for partially ordered problems (1).*

Exact Test for Partially Ordered Problems (2)

- Many partially ordered problems have a special structure:
  - the initial task network is totally unordered and
  - all methods are totally ordered
- This form is sufficient to encode semi-decidable problems
- All networks generated during search are sets of sequences
- We sort these sequences and treat them similar to TO networks

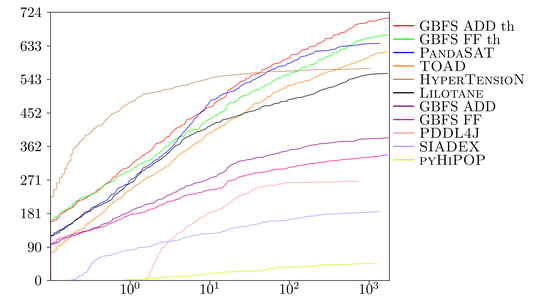

Figure 9: *Empirical Results – Totally Ordered Planning.*

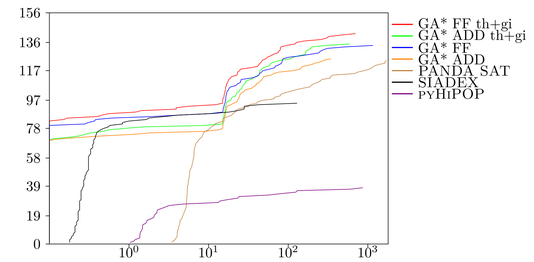

Figure 10: *Empirical Results – Partially Ordered Planning.*

Conclusion

- We realized a graph search based on the progression search of the PANDA system
- To do so, we need to check for duplicate search nodes
- To make this feasible,
  - we realized several approximations that render the search incomplete
  - we exploit special cases during the exact calculation
- Especially on the benchmark set of the Totally Ordered Track, this highly improves PANDA's performance

UNIVERSITÄT DES SAARLANDES

UNI FREIBURG