Improved Pattern Selection for PDB Heuristics in Classical Planning
(Extended Abstract)

Sascha Scherrer and Florian Pommerening and Martin Wehrle
University of Basel, Switzerland
mail@sascha-scherrer.ch, florian.pommerening@unibas.ch, martin.wehrle@unibas.ch

The iPDB approach (Haslum et al. 2007) represents
the state-of-the-art algorithm to compute pattern databases
(PDBs) (Culberson and Schaeffer 1998) for domain inde-
pendent optimal planning. In a nutshell, iPDB selects pat-
terns based on a hill-climbing search in the space of pat-
tern collections. The iPDB heuristic admissibly combines
pattern database heuristics for each pattern in the resulting
collection. Despite its success, the overall iPDB approach
suffers from a problem: the hill-climbing search often gets
stuck in local optima, which limits the quality of the heuris-
tic based on the resulting pattern collection. This problem
has been known for years, and in fact, Haslum et al. (2007)
already pointed it out as a direction for further improve-
ments. However, it has not been addressed in the planning
literature so far. Searching in a larger neighborhood to es-
cape from a local optimum quickly exhausts available re-
sources because of the large number of successors to con-
sider. A successful approach thus must manage resources
carefully. In this research abstract, we propose variable neigh-
bored search (VNS) (Mladenovic and Hansen 1997) to ad-
dress the problem of local optima. Initial experiments are
couraging, in particular showing that the resulting heuris-
tic is competitive with the LM-Cut heuristic (Helmert and
Domshlak 2009) across a wide range of planning domains.

Background

We consider SAS + planning tasks \((\mathcal{V}, \mathcal{O}, s_0, s_*)\) that consist
of a finite set of finite-domain variables \(\mathcal{V}\), a finite set of
operators \(\mathcal{O}\), an initial state \(s_0\), and a goal \(s_*\). For details,
we refer the reader to the literature (e.g., Sievers, Ortlieb,
and Helmert 2012). For a given planning task \(\Pi\), a plan
is a sequence of operators \(\pi\) such that the operators in \(\pi\)
are sequentially applicable starting in \(s_0\), and they lead to
a state \(s_n\) that complies with the goal. A plan is optimal if
it is a shortest plan among all plans (for simplicity, we only
consider unit cost planning tasks). In this paper, we focus on
finding optimal plans. For variables \(v, v' \in \mathcal{V}\), we say that
\(v\) is causally related to \(v'\) if there is an operator \(o \in \mathcal{O}\) that
has a precondition on \(v\) and an effect on \(v'\).

The iPDB approach computes a heuristic for a pattern
collection, i.e., a set of patterns. This pattern collection is
obtained as a result of a local search in the pattern space:
The extensions of a pattern \(P\) are patterns that extend \(P\) by
one variable that is causally related to a variable in \(P\). For
each pattern \(P\) in a collection \(C\) and each extension \(P'\) of
\(P, C\) has one successor, \(C \cup \{P'\}\). The hill-climbing search
starts with a collection that contains one pattern for every
goal variable and greedily selects successors that maximize
the heuristic’s improvement on a set of sample states. The
improvement measures for how many sample states the re-
sulting heuristic now has higher values. The hill-climbing
search terminates once the improvement no longer exceeds
a given threshold. For a more detailed description of iPDB’s
framework beyond the pattern selection, we again refer the
reader to the literature (Haslum et al. 2007).

Variable Neighborhood Search

We investigate a variant of variable neighborhood search
(VNS) (Mladenovic and Hansen 1997) to address the prob-
lem of finding local optima. In a nutshell, VNS tries to find
successor collections in neighborhoods of successively in-
creasing size until a better collection is found. In more de-
ctail, we run iPDB as described above storing all candidate
patterns \(P'\) in a candidate collection \(\mathcal{P}\). If no successor
leads to a sufficient improvement, we do not stop the hill-
climbing as standard iPDB would do, but instead consider
larger neighborhoods, i.e., the extensions of candidate pat-
terns by another causally related variable. We add every ex-
tension of every candidate so far to \(\mathcal{P}\) and continue to look
for a candidate with sufficient improvement. This effectively
searches for improving candidates that are extensions of a
pattern in the collection by more and more variables. We
continue until an improving candidate is added to the col-
lection. At this point, we reset the list of candidates and
start with extensions of the patterns in the collection again.

VNS for iPDB can consider large sets of pattern collec-
tions with increasing neighborhood size. Hence, from a
practical point of view, limits for the time and memory used
by VNS are needed. We build on the implementation of
Sievers et al. (2012), which already limits the size of each
individual PDB and the total size of all PDBs for patterns
in the collection. We additionally limit the total size of all
PDBs for patterns in the candidate collection \(\mathcal{P}\) and only add
patterns to \(\mathcal{P}\) if \(\mathcal{P}\) still respects all limits afterwards. In addi-
tion, we limit the time spent by the hill-climbing algorithm.
Figure 1: Expansions of A* (without the last $f$ layer to ignore tie-breaking issues) using iPDB with and without VNS.

**Evaluation**

We have implemented VNS on top of the iPDB implementation (Sievers, Ortlieb, and Helmert 2012) in the Fast Downward planner (Helmert 2006). We evaluated our approach for optimal planning with A* on the IPC 1998–2011 benchmarks. Our experiments were performed on machines with Intel Xeon E5-2660 CPUs running at 2.2 GHz, with a global time limit of 30 minutes and a memory bound of 2 GB per run. We left the memory limits for the number of abstract states per PDB and for the collection at their defaults (2 million and 20 million, respectively). For VNS, we additionally limit hill-climbing time to 900 s and memory usage of the candidate collection to 400 million abstract states ($\approx 80\%$ of the available memory). Limiting hill-climbing time to 450 s and 1350 s yields similar results as for 900 s. Without limiting hill-climbing time, iPDB exceeds the global time limit in 281 cases before starting the A* search. Considering the memory limit, due to over-allocation and overhead of other data structures, we cannot use all memory to store PDBs. Experiments with 40% and 20% of the available memory solve 3–5 fewer tasks in total.

Figure 1 shows that VNS can further improve iPDB’s quality. This improvement highlights that the original iPDB procedure indeed often gets stuck, and it shows that VNS is a way around this problem. In addition, the per-domain results in Table 1 show that this variant is on par with the LM-Cut heuristic (better in 21 out of 44 domains, worse in 14). Although the number of solved tasks (the coverage) of LM-Cut is higher, we observe that this is primarily due to one domain (Miconic), where LM-Cut solves 86 tasks more than VNS. In contrast, the fraction of solved tasks by domain, averaged over all domains (the coverage score) normalizes the coverage over the number of tasks by domain and is more robust to differences in domains with many tasks. For VNS, this score is about two percentage points higher than that of LM-Cut, which is a considerable improvement.

Table 1: Coverage for different variants of iPDB and LM-Cut. Subscripts for iPDB heuristics show time and memory limits. Per-domain results are shown for Miconic, and where coverage for $h_{\infty,iPDB}$ and $h_{900s,400M,iPDB}$ differs by more than one.

<table>
<thead>
<tr>
<th>Domain</th>
<th>$h_{\infty,iPDB}$</th>
<th>$h_{900s,400M,iPDB}$</th>
<th>$h_{\infty,iPDB}$</th>
<th>$h_{900s,400M,iPDB}$</th>
<th>$h_{\infty,LM-Cut}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Airport (50)</td>
<td>25</td>
<td>38</td>
<td>38</td>
<td>28</td>
<td></td>
</tr>
<tr>
<td>Depot (22)</td>
<td>8</td>
<td>11</td>
<td>11</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>Elevators (50)</td>
<td>36</td>
<td>43</td>
<td>43</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>Floortile (20)</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>Miconic (150)</td>
<td>55</td>
<td>55</td>
<td>55</td>
<td>141</td>
<td></td>
</tr>
<tr>
<td>Parcprinter (50)</td>
<td>22</td>
<td>28</td>
<td>28</td>
<td>31</td>
<td></td>
</tr>
<tr>
<td>TPP (30)</td>
<td>6</td>
<td>8</td>
<td>8</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>Transport (50)</td>
<td>17</td>
<td>24</td>
<td>24</td>
<td>17</td>
<td></td>
</tr>
<tr>
<td>Trucks (30)</td>
<td>8</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Woodworking (50)</td>
<td>13</td>
<td>23</td>
<td>23</td>
<td>29</td>
<td></td>
</tr>
</tbody>
</table>

Sum (502)        | 192               | 221                   | 244               | 317                   |                     |
Sum in other domains (894) | 474 | 473 | 481 | 452 |
Total sum (1396)  | 666               | 694                   | 725               | 769                   |                     |
Coverage score (in %) | 50.72 | 52.68 | 55.45 | 53.60 |

Conclusions

Our initial evaluation of VNS for iPDB is encouraging, showing significant improvements compared to basic iPDB, and competitive results with LM-Cut. In the future, it will be interesting to investigate more informed neighborhood extension algorithms, for example, preferring promising patterns depending on the problem structure.

Acknowledgments

This work was supported by the Swiss National Science Foundation as part of the project “Abstraction Heuristics for Planning and Combinatorial Search (AHPACS)”.

References


