Linear-Time Filtering Algorithms for the Disjunctive Constraint

Hamed Fahimi, Claude-Guy Quimper
Université Laval
Department of Computer science and Software engineering
hamed.fahimi.1@ulaval.ca, claude-guy.quimper@ift.ulaval.ca

Abstract
We present three new filtering algorithms for the DISJUNCTIVE constraint that all have a linear running time complexity in the number of tasks. The first algorithm filters the tasks according to the rules of the time tabling. The second algorithm performs an overload check that could also be used for the CUMULATIVE constraint. The third algorithm enforces the rules of detectable precedences. We introduce the new data structure time line for the last two algorithms. The time line provides some constant time operations that were previously implemented in logarithmic time by the Θ-tree data structure. Experiments show that these new algorithms are competitive even for a small number of tasks and outperform existing algorithms as the number of tasks increases.

Disjunctive constraint
Let I = {A1, …, As} be a set of tasks with unknown starting times si and known processing times pi. The constraint DISJUNCTIVE((A1, …, As)) is satisfied, if for all pairs of tasks i and j (i ≠ j), where
si + pj ≤ sj + pi.

Time-tabling
This rule exploits the fact that a task i must execute within its compulsory part [lsti, ecti], if lsti < ecti. Consequently, if there exists a task j with lstj < ectj and there exists a task k that satisfies ectj ≥ lstj, then j has to execute after k.

Strategy of the algorithm: The algorithm sorts the tasks in non-decreasing order of the processing times. While iterating through the sorted tasks, it jumps over the tasks which have compulsory parts and schedules the earliest one possible point. Afterwards, the union find merges the traversed compulsory parts. This operation reduces the number of jumps and leads to a linear time running after processing all the tasks.

Overload check
This rule does not filter the space. Rather, it detects an inconsistency and triggers a backtrack during the search process.

Strategy of the algorithm: The tasks are scheduled in non-decreasing order of the latest completion times with the time line. If after scheduling a task i, the earliest completion time is greater than lstj, then the overload check fails.

Detectable Precedences
This technique consists of finding the following set for a task i:
Ωi = {j ∈ I \ {i} | ectj > lsti}

Once this set is found, one can delay the earliest starting time of i up to
Est′ = max(Estj, ectj(Ωi \ {i}| ectj > lstj).

Strategy of the algorithm: We introduced a new algorithm to enforce the rule of detectable precedences. The algorithm simultaneously iterates over all the tasks in non-decreasing order of ectj and on all the tasks k in non-decreasing order of lstk finds the detectable precendences and filters the domains at the end of the iteration through i.

Experiments
Here are the tables of our experiments on the benchmarks for the open-shop problem and job-shop problem with a jobs and m tasks per job. The numbers indicate the ratio of the cumulative number of backtracks performed by our algorithms and the number of backtracks performed by the state-of-the-art algorithms. A ratio greater than 1 means that, within 10 minutes of computations, the state-of-the-art algorithms explore a larger portion of the search tree.

Conclusion
Thanks to the constant time operation of the Union-Find data structure, we designed the new time line data structure, to speed up filtering algorithms for the Disjunctive constraint.

We came up with three faster algorithms to filter the disjunctive constraint. The following table exhibits the results.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Previous complexity</th>
<th>Improved complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time-Tabling</td>
<td>O(n log(n))</td>
<td>O(n)</td>
</tr>
<tr>
<td>Overload check</td>
<td>O(n log(n))</td>
<td>O(n)</td>
</tr>
<tr>
<td>Detectable precedences</td>
<td>O(n log(n))</td>
<td>O(n)</td>
</tr>
</tbody>
</table>

Thanks to the constant time operation of the Union-Find data structure, we designed the new time line data structure, to speed up filtering algorithms for the Disjunctive constraint.