Algorithm Engineering – Exercise 5

Team 02: Julian Fechner, Julio Cesar Perez Duran, and Denis Koshelev

1. Implemented Features

Branch-and-reduce algorithm: We implemented the new branching algorithm with the two constraints from the lecture. As depicted in Figure 1, the new algorithm was overall faster and the number of recursive steps was greatly reduced due to the constraints and the heuristic upped bound.

Reduction rules: We implemented three new reduction rules: *Twin Rule* [1], *Degree-Three-Independent-Set Rule* and *improved LP relaxation*. Unfortunately, none of the rules proved to be sufficiently efficient, but they still had some impact in reducing the number of recursive steps. As demonstrated by Figure 2, improved LP relaxation performed best, however its computation cost was too high. Other rules, also compared with constraints, also reduced the number of recursive steps, but not significantly.

Maximum Clique Solver: We used the strategy of applying Maximum Clique Solver from the winner solver "WeGotYouCovered" of the PACE challenge for Vertex Cover [2]. Our algorithm primarily utilized a branch-and-reduce technique, leaving a small time slot to resolve challenging instances by computing the maximum clique on the complement graph. We employed the MoMC Algorithm, which performs well on sparse instances [3]. As expected, this resulted in a successful performance on the complement dense graphs, since it solves 35 additional instances, as depicted in Figure 3. Our program faced challenges running on the university server due to the path configurations, so we chose not to include that version in the submission.

Bottlenecks: Three main bottlenecks were identified with the VisualVM Profiler: the calculation of the complement graph, the creation of new graphs for components, and the domination rule. We solved them by utilizing our data structures and updating them on the run. Here is an example of the speedup for two graphs: `05-football.graph` and `vc035`:

Change	Complement	Components	Domination
Before	6453 ms	1041 ms	12180 ms
After	733 ms	367 ms	9298 ms

2. Data Structures

In order to utilize the improved LP relaxation, we introduced a new class called `FlowGraph` for storing edges, their flow and capacity. However, it was not optimized, as it used an adjacency map with vertices and corresponding edges, as well as a hashset of all edges, for both the flow graph and the residual graph. This resulted in the LP relaxation being too slow to include in the final submission, despite its ability to significantly decrease the number of recursive steps, as previously noted.

3. Highlights

As seen in Figure 3, the new solver without Maximum Clique integration can only solve one more test case. Despite individual features reducing recursive steps, as shown in Figure 2, their combination did not result in a significant improvement, as all of the previous iteration rules were still enabled, already reducing the graphs, and the implementation was also slow.

There are still some challenges in the implementation of the solver, such as updating the bipartite graph for the LP bound, the unconfined rule, and the old LP Reduction. For instance, in test case `vc035` these challenges resulted in 45%, 17% and 2% of the running time respectively, not considering branching algorithm itself.

4. Experiments

SMAC: We added on/off toggles and depth-parameters to all reduction rules and lower-bounds, then trained an ML-model with SMAC. After 24 hours of training, the model was stopped due to a timeout that we set, and the "incomplete" result was used in the solver. The optimized parameters found by SMAC improved the runtime for the training dataset but worsened it for testcases not included in the training dataset.

To avoid an exponential increase from each additional toggle in the configuration space, we conducted another SMAC testrun with only depth parameters for the two lower-bounds on the same training dataset. This resulted in different depth threshold parameters (bounds=10, reduction=5), which still provided a small speed-up for the training dataset.

References

[1] M. Xiao, H. Nagamochi, Confining sets and avoiding bottleneck cases: A simple maximum independent set algorithm in degree-3 graphs doi:https://doi.org/10.1016/j.tcs.2012.09.022.

[2] D. Hespe, S. Lamm, C. Schulz, D. Strash, WeGotYouCovered: The Winning Solver from the PACE 2019 Challenge, Vertex Cover Track, pp. 1–11. doi:10.1137/1.9781611976229.1.

[3] C.-M. Li, H. Jiang, F. Manyà, On minimization of the number of branches in branch-and-bound algorithms for the maximum clique problem doi:https://doi.org/10.1016/j.cor.2017.02.017.

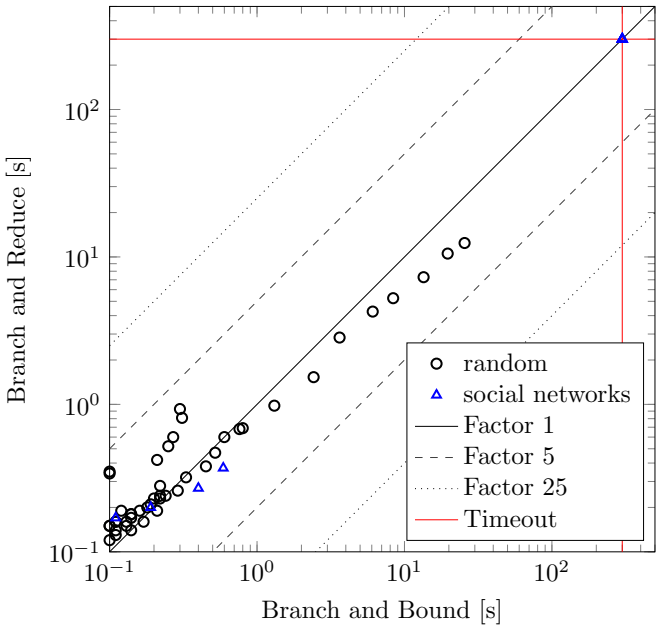

Figure 1: Comparison of the branch-and-bound and branch-and-reduce algorithms.

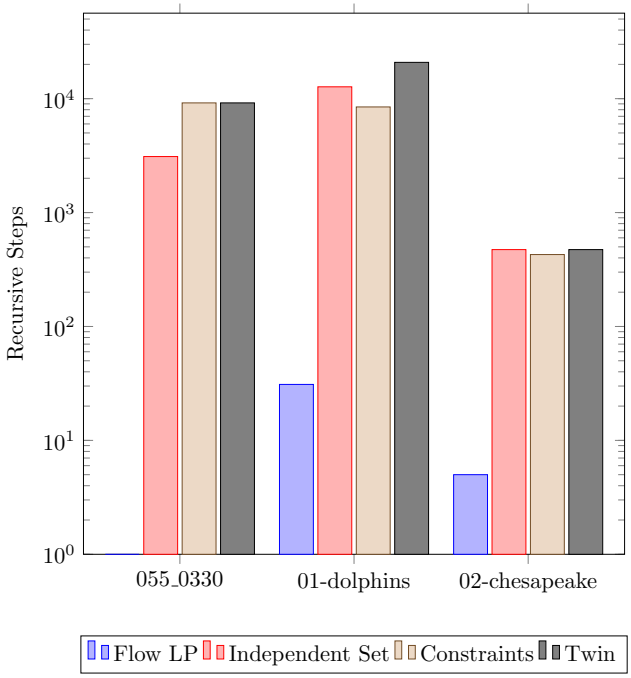

Figure 2: Overview of recursive steps for new reduction rules on specific graph instances.

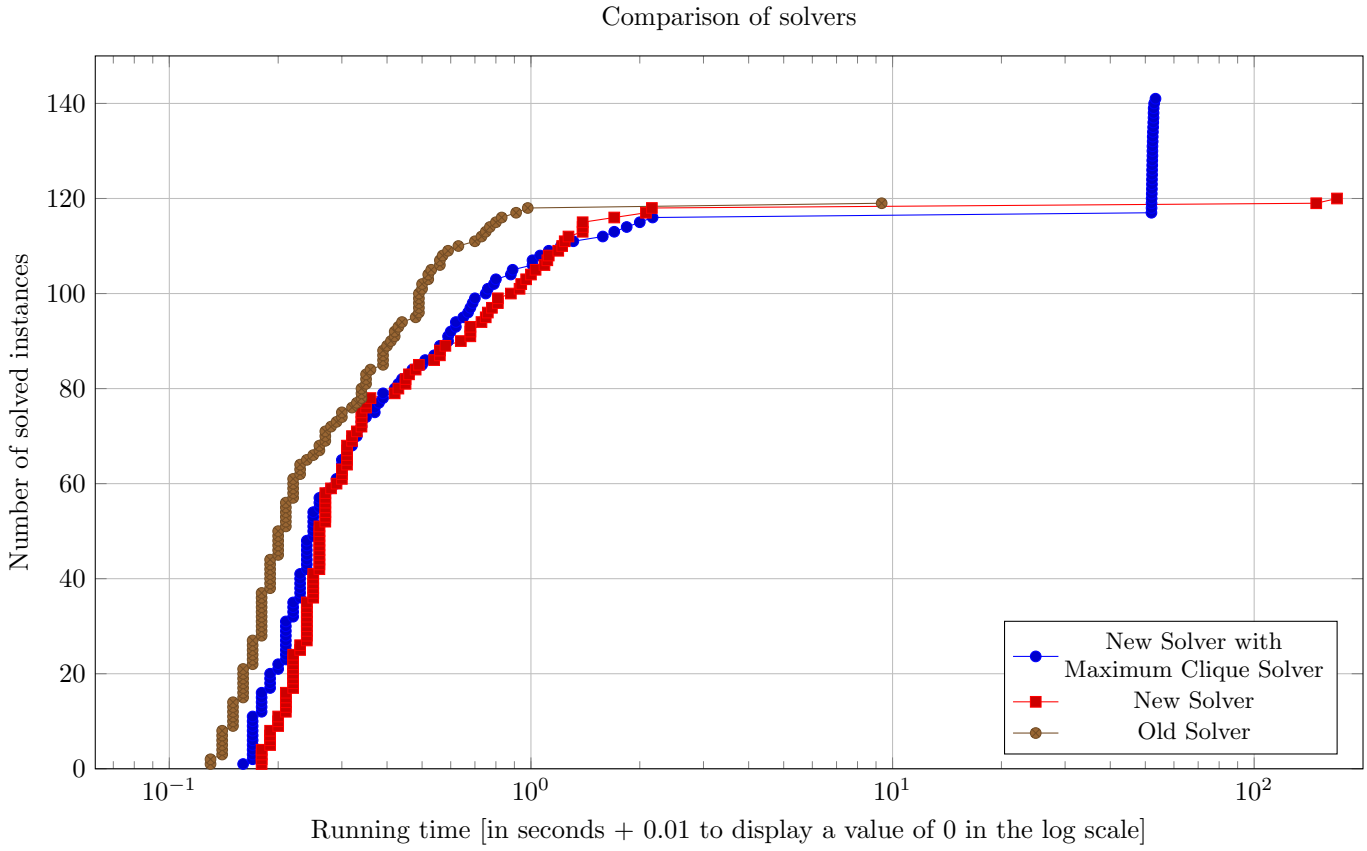

Figure 3: Comparison of different versions of our solvers. Timeout was set to 60 sec and 5 instances were allowed to fail.