Algorithm Engineering – Exercise 4

Team 3 (Dennis Weiss, Lisa Dörr, Khiem That Ton)

1. Exact Algorithm - Reduction to Other Solvers

We formulated Weighted Cluster Editing as an Integer Linear Programming (ILP) and Constraint Satisfaction (CS) problem. For the ILP version we used the IBM ILOG CPLEX 12.9 solver and for the CS formulation we used the CP-SAT solver from Google OR-Tools. In Fig. 1 we compare the running times of the ILP version, the CS version and our fastest search tree algorithm as described in our previous exercise.

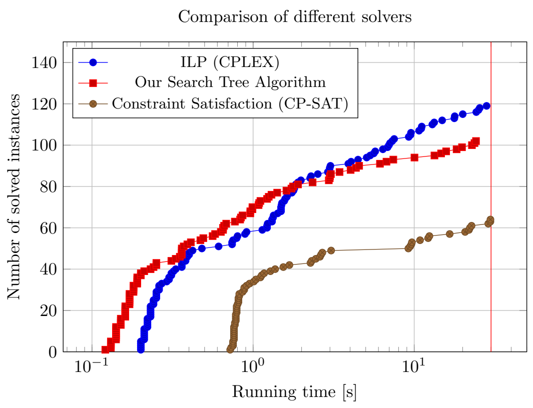

Figure 1: Comparison of running time on the provided problem instances of the ILP and CS solvers and our fastest search tree algorithm

2. Heuristics

We implemented the following greedy heuristics:

- *Greedy Heuristic 1*: The closed neighborhood heuristic mentioned in the lecture
- *Greedy Heuristic 2*: The greedy heuristic described in [1], where edges are iteratively removed such that a connected component is split into two disconnected components. The order in which edges are removed is defined by the score of an edge, as described in detail in [1]. If the cost of splitting is higher than the cost of making the connected component a clique, the latter is returned. Otherwise the procedure is recursively called on the two disconnected components.
- *Greedy Heuristic 3*: As Greedy Heuristic 2, but uses the vertex pair scoring strategy from Sec. 3 to decide on which edge to remove first (lowest score edges are removed first).
- *Greedy Heuristic 4*: An algorithm that firstly scores all vertex pairs as in Sec. 3. Starting from the graph with no edges, it iteratively adds edges (highest score first), takes the transitive closure of the created graph and updates the cost of editing the input graph to it. The solution with the minimum cost is saved and returned eventually. During the edge addition procedure an edge is only added with probability p. This allows to circumvent edges that are detrimental to the cost of the solution graph but received a high score.
- *Greedy Heuristic 5*: As *Greedy Heuristic 4*, but uses the signed edge weight as score and a probability p of adding the edge $\{u, v\}$ depending on the cost difference of editing the input graph to the current and to the graph with the clusters of u and v merged.
- *Greedy Heuristic 6*: We use Spectral Clustering [2] to find clusters which we connect to cliques in the solution graph.

We used Simulated Annealing (SA) [3] to further improve the solution found by a greedy heuristic and get closer to a globally optimal solution. A neighboring graph G' of the current graph G is generated by choosing a vertex v out of $V(G)$ uniformly at random, removing all edges of v and connecting v to $N[w]$, where w is also sampled uniformly at random from $V(G)$. Let $c(G)$ be the cost of editing the input graph to reach the graph G. During the SA process we change the current solution from G to G' with probability p_{accept}.

$$p_{accept} = \min\left(\exp\left(\frac{c(G) - c(G')}{T}\right), 1\right) \tag{1}$$

The temperature value T influences the exploration-exploitation trade-off. In our algorithm we employ a linear decrease of T from T_{start} to 0 over the iterations of SA.

3. Scoring Vertex Pairs Using Relaxed ILPs on Subgraphs

For l iterations the graph is split into random partitions G' of size $n' := \lceil \alpha \cdot \sqrt{n} \rceil$. If n is not an integer multiple of n', the last partition has size $n \mod n'$. Then we solve the relaxed ILP for Weighted Cluster Editing using Google's GLOP solver for each partition G' and sum the LP variable assignments in all l iterations to compute a score of every vertex pair.

4. Optimizing Parameters

In the following analysis we used the average[1] ratio between the exact solutions size and the solution size found by the heuristic as value to score the heuristic algorithm.

In Fig. 2 we evaluate for which combination of edge addition probability p and *minScore*[2] the best solution was found by *Greedy Heuristic 4*. In the vast majority of cases the optimal solution was found for an edge addition probability of 0.5 or greater. As a consequence, we only use edge addition probabilities in $[0.5, 1]$ for our algorithm.

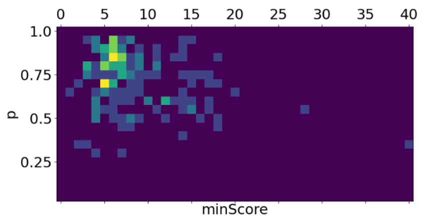

Figure 2: Occurrence of lowest cost solution for combinations of *minScore* and edge addition probability p

By analyzing the effect of α, as in Fig. 3, we decided to use $\alpha = 1$ for the fast version of the heuristic and $\alpha = 2.5$ for the thorough version, as no large increase in solution quality, but large increase in running time was observed for a value greater than 2.5.

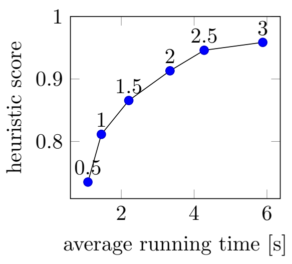

Figure 3: Average running times and heuristic scores for different values of α from 0.5 to 3 whilst using $l = 40$ and $k = 10$

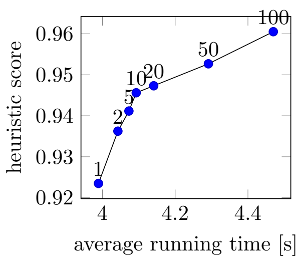

Figure 4: Average running times and heuristic scores for different k from 1 to 100 whilst using $l = 40$ and $\alpha = 2.5$

In Fig. 4 the average running times and heuristic scores for different values of k are plotted. We evaluate $k = 10$ to be a good trade-off between running time and solution quality for both versions of the heuristic algorithm.

Based on the observations in Fig. 5 we choose $l = 20$ for our fast algorithm and $l = 40$ for our thorough algorithm.

5. Our Productive Heuristic Algorithm

In our productive algorithm we loop through the heuristic algorithms multiple times in the order from fastest to slowest algorithm, as plotted in Fig. 7. Furthermore, we decided to not include *Greedy Heuristic 3* and *5*. As

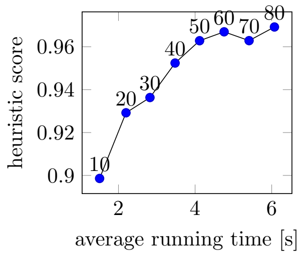

Figure 5: Average running times and heuristic scores for different l from 10 to 80 whilst using $k = 10$ and $\alpha = 2.5$

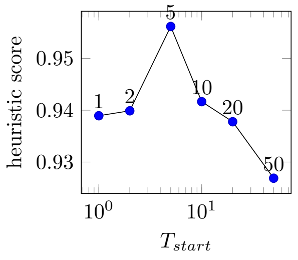

Figure 6: Heuristic scores for different T_{start} after the fast version of *Greedy Heuristic 4* and SA with 30'000 iterations

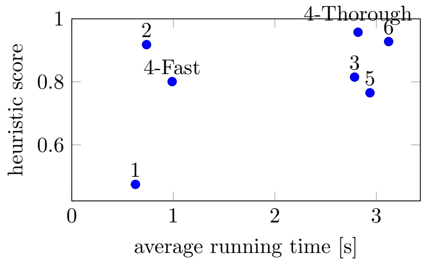

Figure 7: Average running times and heuristic scores for different greedy heuristics

Greedy Heuristic 2 is not randomized, the heuristic is only computed on the first call and in all consecutive calls just the SA process is repeated. On the publicly available instances with provided exact solution the algorithm achieved a heuristic score of $\sim$ **99.894**%.

Algorithm 1 Our Heuristic Algorithm

```
repeat
    for $1 \leqslant i \leqslant 256$ do
        Greedy Heuristic 1 with SA
    for $1 \leqslant i \leqslant 32$ do
        Greedy Heuristic 2 with SA
    for $1 \leqslant i \leqslant 8$ do
        Greedy Heuristic 4 (Fast) with SA
    for $1 \leqslant i \leqslant 2$ do
        Greedy Heuristic 4 (Thorough) with SA
    for $1 \leqslant i \leqslant 2$ do
        Greedy Heuristic 6 with SA
until Termination
return Lowest cost solution found
```

[1] Averages are taken over all instances that were solved within a 30s time limit by all algorithm versions considered for comparison.

[2] *minScore* is the value for which vertex pairs with a score greater or equal are considered for edge addition.

References

[1] Sven Rahmann et al. "EXACT AND HEURISTIC ALGORITHMS FOR WEIGHTED CLUSTER EDITING". In: *Computational Systems Bioinformatics*. 2007, pp. 391–401.

[2] Andrew Ng, Michael Jordan, and Yair Weiss. "On Spectral Clustering: Analysis and an algorithm". In: ed. by T. Dietterich, S. Becker, and Z. Ghahramani. Vol. 14. MIT Press, 2002.

[3] Peter J. M. van Laarhoven and Emile H. L. Aarts. "Simulated annealing". In: *Simulated Annealing: Theory and Applications*. Dordrecht: Springer Netherlands, 1987, pp. 7–15. ISBN: 978-94-015-7744-1.