PACE 2024 Solver Description

Martin Josef Geiger
University of the Federal Armed Forces Hamburg, Germany

Abstract

This extended abstract outlines our contribution to the Parameterized Algorithms and Computational Experiments Challenge (PACE), which invited to work on the one-sided crossing minimization problem. Our ideas are primarily based on the principles of Iterated Local Search and Variable Neighborhood Search. For obvious reasons, the initial alternative stems from the barycenter heuristic. This first sequence (permutation) of nodes is then quickly altered/ improved by a set of operators, keeping the elite configuration while allowing for worsening moves and hence, escaping local optima.

2012 ACM Subject Classification Applied computing

Keywords and phrases PACE 2024, one-sided crossing minimization, Variable Neighborhood Search, Iterated Local Search

Digital Object Identifier 10.4230/LIPIcs.CVIT.2016.23

1 Problem description and some reflections

1.1 The problem

In the one-sided crossing minimization problem, a graph $G = (V, E)$ is given, which consists of a vertex set V and an edge set E. G is bipartite as there is a partition of V into two disjoint subsets V_1, V_2 (hence, $V = V_1 \cup V_2$, $V_1 \cap V_2 = \emptyset$, and $E \subseteq V_1 \times V_2$). We now assume that the nodes of V_1 are arranged in a linear order and placed in one layer, while the ones of V_2 appear in another layer parallel to the first one. Therefore, edges between V_1 and V_2 may cross, depending on the sequence of nodes in V_1, V_2. In it's *one-sided* variation, the crossing minimization problem lies in arranging (ordering) the nodes in V_2 – while assuming a fixed linear order $<_1$ of V_1 – such that the total number of edge crossings is minimal.

Several applications for this problem can be found in the literature, with graph drawing as a prominent example [3].

1.2 A lower bound and a corollary

It follows that the solution to the problem can be characterized as finding a (cost-minimal) linear order $<_2$ for V_2. In any such order, two nodes $a, b \in V_2$ can appear either ordered $a < b$ or $b < a$, and the crossings count c_{ab} or (XOR) c_{ba} are part of the optimal value. A trivial lower bound is obtained by considering all distinct pairs $a, b \in V_2$, and computing the sum over all $\min\{c_{ab}, c_{ba}\}$-values.

Concept 1. Based on this lower bound computation, we can construct a digraph on V_2, introducing arcs (a, b) iff $c_{ab} < c_{ba}$, and arcs (b, a) iff $c_{ba} < c_{ab}$. In some ideal cases, this digraph is acyclical, and an optimal ordering $<_2$ is quickly computed based on this preliminary input. Unfortunately, acyclicity is not always present. It follows that, in those cases, any linear order $<_2$ breaks at least one (often: some, several) cycles, and the problem can be reformulated as finding a minimal-cost cycle-breaking of the constructed digraph. Part of the process now becomes identifying the elementary circuits of the digraph, e. g. by means of [4], and breaking them in an optimal manner. In our experience, if G becomes 'large', this process becomes computationally difficult.

Concept 2. Alternative approaches directly construct and manipulate the linear order $<_2$ by considering a permutation of the nodes in V_2, and hence *implicitly* break existing cycles by forcing transitivity over all binary relations of $a, b \in V_2$. As the transition from the digraph into the permutation is a mapping from a higher into a lower dimensional space (i. e., a lossy compression), such approaches are more direct but fail to enumerate the cycles in a structured manner.

2 Submitted algorithm

Our approach is primarily based on the principles of Variable Neighborhood Search [2] and Iterated Local Search [5]. In the spirit of the classification above, we follow Concept 2, and consider permutations of nodes of V_2.

2.1 Preprocessing and reductions

Reducing the size of the instance is beneficial. First, we exclude isolated nodes in V_2, i. e., nodes that have no edges. Then, and excluding the very large instances, all c_{ab}-values are pre-computed. On this basis, the reduction rules RR1 and RR2, as given in [1], are applied.

If possible, V_2 is broken down into linearly ordered, disjoint subsets, such that the nodes of each subset must precede the ones of the following subset in the permutation, etc. Each subset can then be treated as an independent sub-problem, and the search process is therefore accelerated. This partitioning can be computed in $\mathcal{O}(|V_1| + |E|)$, and therefore feasible in cases in which pre-computing the crossings-matrix is too expensive.

2.2 Initial permutation of V_2

The starting solution stems from the barycenter-heuristic [6]. This is important, as the challenge organizers have published some instances for which this approach yields the optimal solution. In those cases, our program terminates early. In the Heuristics-Track of the competition, this applies to 12 of the 100 instances.

2.3 Improvement moves

We exhaustively search for improving moves until a local optimum is reached.

- First, the *single node move* tries to remove a node from it's current position and re-insert in some other place in the permutation.
- Then, *block moves* try to move entire blocks of subsequent nodes. The size of the blocks range from 2 to 5 nodes. Our experiments indicate that block moves contribute to the performance of the algorithm only a little – but still they do.

Improving moves are always accepted, and moves that do not change the quality of the current solution are considered with a certain probability in order to diversify the search.

Several truncation-techniques are implemented in order to speed-up the search. Obviously, moves that contradict the order given by the reduction rules RR1 and RR2 are omitted. Also, when moving a node (or a block), movements are stopped once their cumulative change in the objective function value exceeds a certain threshold: In those cases, we do not hope for an improvement to show up.

For the larger instances, i. e. the ones in which computing the crossings matrix is considered to be computationally too expensive, we truncate the movements further by introducing a maximum range (change of positions) for shifting nodes in the permutation. This is important as the algorithm otherwise spends too much time re-inserting a give node before moving on to the next node.

2.4 Diversification move

Once a local optimum is reached, a subset of the permutation is reversed and search continues from here. We allow for a maximum of 20% of the permutation to be reversed. Based on our experiments, this value presents a good compromise between diversifying and intensifying the search.

3 Source-code

The source-code of our contribution has been published under the Creative Commons Attribution 4.0 International Public License and made available under `https://doi.org/10.5281/zenodo.11465516`.

References

1 Vida Dujmović, Henning Fernau, and Michael Kaufmann. Fixed parameter algorithms for ONE-SIDED CROSSING MINIMIZATION revisited. *Journal of Discrete Algorithms*, 6:313–323, 2008.
2 Pierre Hansen and Nenad Mladenović. Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, 130:449–467, 2001.
3 Patrick Healy and Nikola S. Nikolov. Hierarchical drawing algorithm. In Roberto Tamassia, editor, *Handbook of Graph Drawing and Visualization*, Discrete Mathematics and its Applications, chapter 13, pages 409–453. CRC Press – Taylor Francis Group, Boca Raton, FL, USA, June 2013.
4 Donald B. Johnson. Finding all elementary circuits in a directed graph. *SIAM Journal on Computing*, 4(1):77–84, 1975.
5 Helena R. Lourenço, Olivier Martin, and Thomas Stützle. Iterated local search. In Fred Glover and Gary A. Kochenberger, editors, *Handbook of Metaheuristics*, volume 57 of *International Series in Operations Research & Management Science*, chapter 11, pages 321–353. Kluwer Academic Publishers, Boston, Dordrecht, London, 2003.
6 Kozo Sugiyama, Shojiro Tagawa, and Mitsuhiko Toda. Methods for visual understanding of hierarchical system structures. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-11(2):109–125, February 1981.