



guilhermefonseca

Guilherme D. da Fonseca  

LIS, Aix-Marseille Université

Abstract

We describe the solvers used by the Shadoks team in the PACE 2024 challenge. The challenge considers solvers for the one-sided crossing minimization problem (OCM). Each instance contains a bipartite graph with two partitions called *top* and *bottom*. The top partition comes with a vertex order. The output is the vertex order of the bottom partition and the goal is to minimize the number of edge crossings when the vertices of the two partitions are placed on two horizontal lines in order and the edges are drawn as line segments.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Graph drawing, crossing number, heuristics, local search.

Related Version <https://doi.org/10.5281/zenodo.11452654>

Supplementary Material (*Source Code*): <https://github.com/gfonsecabr/shadoks-PACE2024>

Funding Work supported by the French ANR PRC grant ADDS (ANR-19-CE48-0005).

Acknowledgements We would like to thank the challenge organizers and other competitors for their time, feedback, and making this whole event possible. We would also like to thank Yan Gerard for the insightful discussions.

1 Introduction

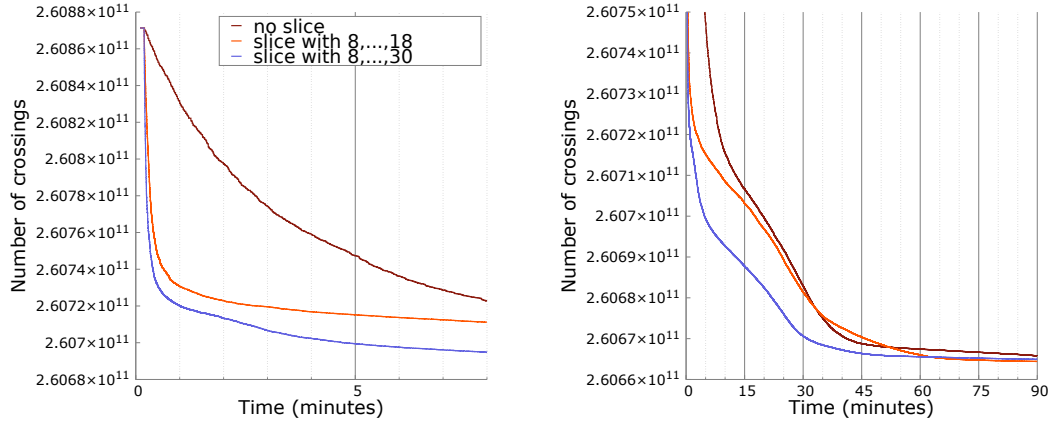
The PACE challenge is an annual optimization challenge. The 2024 edition considers the problem of the following graph drawing problem. The input is a bipartite graph $G = (V_0, V_1, E)$ where the vertices V_0 are ordered. The output is an ordering of V_1 . The goal is to minimize the number of edge crossings when both V_0 and V_1 are placed on two horizontal lines in order and the edges are drawn as line segments. For an extensive overview of the problem, see [7].

Numerous heuristics have been compared in [5]. We use three of these heuristics:

- **Barycenter:** The Barycenter heuristic [6] considers that the vertices of V_0 have x -coordinates $1, \dots, |V_0|$ and assigns the coordinate of each vertex $v \in V_1$ as the average of the coordinate of its neighbors.
- **Median:** The Median heuristic [4] is similar to the barycenter heuristic, except that the median is used instead of the average.
- **Split:** We could not obtain the article describing the Split heuristic [3], but we believe it works as follows. We randomly choose a *pivot* from V_1 . We then split the remaining vertices of V_1 into two sets, depending on whether there are fewer crossings when they are placed to the left or to the right of the pivot. Each side is then solved recursively.

The barycenter heuristic often gives the best results in practice and is very fast. The median heuristic is shown to provide a 3-approximation [4], but is often worse in practice. Both heuristics can easily be coded in $O(|E| + |V_1| \log |V_1|)$ time.

The split heuristic is slower and the solutions are often worse than the average heuristic, but may provide very different solutions depending on the random numbers used. In order to efficiently code the split heuristic, we pre-compute the following function $M : V_1 \times V_1 \rightarrow \mathbb{Z}$. The value of $M(u, v)$ is the difference between the number of crossings between the edges



■ **Figure 1** Number of crossings over 8 and 90 minutes of running time of the heuristic solver for one solution of heuristic instance **44.gr** using different parameters. Two scales of the same graph are shown. The instance has $|V_0| = 65536$, $|V_1| = 65536$, and $|E|/|V_1| = 17$.

adjacent to u and v if u is placed to the left and to the right of v . After precomputing M , the split heuristic takes $O(|V_1| \log |V_1|)$ expected time (assuming ties are broken randomly).

Given a solution, it is important to be able to calculate the number of crossings efficiently. There are numerous algorithms to compute the crossings in $O(|V_1| \log |V_1|)$ time [1, 2]. We remark that the problem is equivalent to count the number of non-dominating pairs of points in the plane by mapping each edge into the point defined by the x -coordinates of the two endpoints.

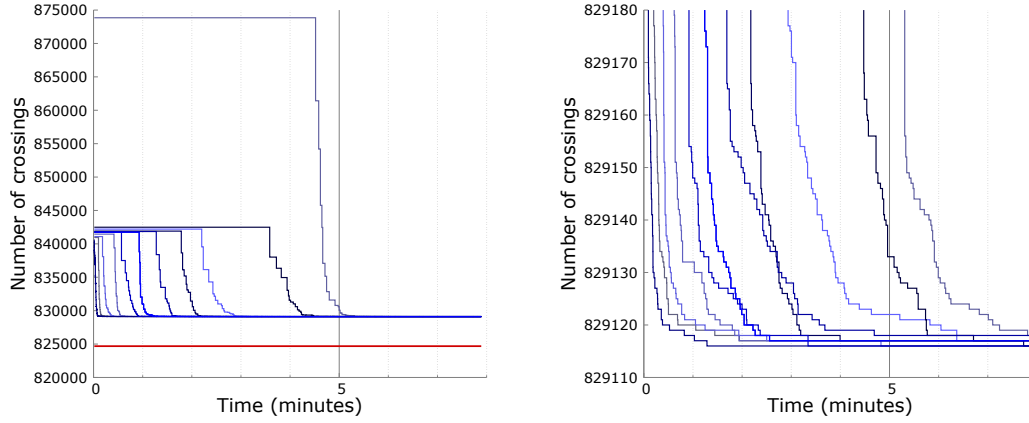
2 Our Solver

In our solver, we obtain initial solutions using the barycenter, median, and split heuristics and then improve the solutions in three different ways:

- **Jump:** We move a (randomly chosen) vertex of V_0 to the position that minimizes the number of crossings. Using the precomputed M , this takes $O(|V_0|)$ time.
- **Slice:** We randomly choose an interval of bottom vertices and compute a split solution to that interval. The new order is kept if the number of crossings does not increase.
- **Slice-and-jump:** We randomly choose an interval of bottom vertices and compute a split solution to that interval and subsequently apply several jumps to that interval. The new order is kept if the number of crossings does not increase.

The size of the interval in the slice greatly affects the performance of the heuristic. For the challenge, we set this choice as follows, with the interval defined in terms of a uniformly distributed *center* and a *radius* to assure that every element is equally likely to be in the interval. We used the radius as the *square* of a uniform random variable in the range $8, \dots, 30$ and $8, \dots, 18$ for slice and slice-and-jump, respectively. We used the square to optimize small intervals (that are faster) more often than large intervals.

Alternating only between jump and slice-and-jump gives very good results for most instances. The slice approach is however faster when it is very easy to make small improvements the solution (notably heuristic instance **44.gr**). Figure 1 shows the evolution of a solution over time, comparing the submitted version that uses slice with an $8, \dots, 30$



■ **Figure 2** Number of crossings over 8 minutes of running time of the heuristic solver for 12 independent solutions of heuristic instance 72.gr. Two different scales of the same graph are shown. The lower bound is shown in red. The instance has $|V_0| = 15818$, $|V_1| = 8772$, and $|E|/|V_1| = 3.5$.

range, the version using slice with an 8, . . . , 18 range, and the version with no slice without jump.

We run the heuristic simultaneously on a number s of solutions. In the heuristic track, we set $s = 12$, while the exact track has $s = 32$. Figure 2 shows the evolution of these $s = 12$ solutions over time. We prioritize improving the best solution so far, and proceed to the next one when we could not improve it.

The *confidence* of the best solution is the number of solutions achieving the best number of crossings divided by s . In the exact version a confidence of $3/4$ is required to output a solution. In both versions, we stop the calculation prematurely if the confidence gets to 1 (or if a trivial lower bound is reached).

References

- 1 Christian Bachmaier, Florian Fischer, and Michael Forster. Radial coordinate assignment for level graphs. In *International Computing and Combinatorics Conference*, pages 401–410, 2005.
- 2 Wilhelm Barth, Michael Jünger, and Petra Mutzel. Simple and efficient bilayer cross counting. In *Graph Drawing: 10th International Symposium, GD 2002 Irvine, CA, USA, August 26–28, 2002 Revised Papers 10*, pages 130–141. Springer, 2002.
- 3 Peter Eades and David Kelly. Heuristics for reducing crossings in 2-layered networks. *Ars Combinatoria*, 21(A):89–98, 1986.
- 4 Peter Eades and Nicholas C Wormald. Edge crossings in drawings of bipartite graphs. *Algorithmica*, 11:379–403, 1994.
- 5 Michael Jünger and Petra Mutzel. 2-layer straightline crossing minimization: Performance of exact and heuristic algorithms. In *Graph algorithms and applications i*, pages 3–27. World Scientific, 2002.
- 6 Kozo Sugiyama, Shojiro Tagawa, and Mitsuhiro Toda. Methods for visual understanding of hierarchical system structures. *IEEE Transactions on Systems, Man, and Cybernetics*, 11(2):109–125, 1981.
- 7 Roberto Tamassia. *Handbook of graph drawing and visualization*. CRC press, 2013.