Evolutionary Algorithms for One-Sided Bipartite Crossing Minimisation

Jakob Baumann
University of Passau, Germany

Ignaz Rutter
University of Passau, Germany

Dirk Sudholt
University of Passau, Germany

Abstract

Evolutionary algorithms (EAs) are universal solvers inspired by principles of natural evolution. In many applications, EAs produce astonishingly good solutions. To complement recent theoretical advances in the analysis of EAs on graph drawing [1], we contribute a fundamental empirical study.

We consider the so-called One-Sided Bipartite Crossing Minimisation (OBCM): given two layers of a bipartite graph and a fixed horizontal order of vertices on the first layer, the task is to order the vertices on the second layer to minimise the number of edge crossings. We empirically analyse the performance of simple EAs for OBCM and compare different mutation operators on the underlying permutation ordering problem: exchanging two elements (*exchange*), swapping adjacent elements (*swap*) and jumping an element to a new position (*jump*). EAs using jumps easily outperform all deterministic algorithms in terms of solution quality after a reasonable number of generations. We also design variations of the best-performing EAs to reduce the execution time for each generation. The improved EAs can obtain the same solution quality as before and run up to 100 times faster.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases Mutation Operator, Layered Graphs, Crossing Minimisation

Digital Object Identifier 10.4230/LIPIcs.GD.2024.51

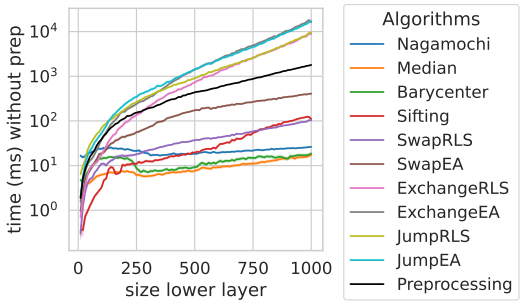
Category Poster Abstract

Related Version *ArXiv Version*: `https://arxiv.org/abs/2409.15312`

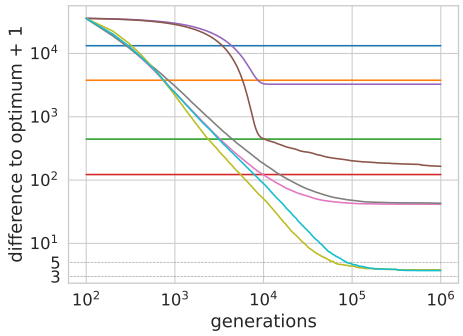
1 Empirical Performance Comparison

We study the mutations swap, exchange, and jump on a simplistic (1+1)-type EA [7], on One-Sided Bipartite Crossing Minimisation [3]. The (1 + 1)-EAs (Swap-EA, Exchange-EA, Jump-EA) start with a random permutation and apply the corresponding operator k times, following a Poisson distribution with $\lambda = 1$. We also consider the randomised local search (RLS), where we set $k = 1$ constant. We compare the EAs to four state-of-the-art algorithms: The Barycenter and Median algorithms [3], Nagamochi's algorithm [6], and a heuristic known as Sifting [5]. Nagamochi's algorithm gives the best theoretical approximation ratio, but its performance was never empirically evaluated; a gap we aim to close with this work. We note that there are other well-performing algorithms, for which evaluations are readily available. We believe that the chosen subset is sufficient for this comparison.

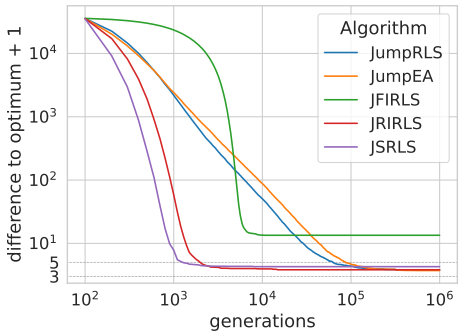
We performed tests on three different instances, similar to [2]. Due to space limitations, and as there are no significant differences, we present only the results for *random instances*, with $n = 100$ vertices on both layers, where we added each edge with a fixed probability p. Note that we also considered differently sized layers and increasing density p; the behaviour of RLS/EAs was basically the same, while the other algorithms were slightly affected, most of which was also covered in [2]. We computed the optimum solution using an ILP [4], which can solve instances of size up to $n \approx 190$. EAs and RLS perform a preprocessing step of computing the cross table [2], which takes $\Theta(nm)$ steps.



(a) Preprocessing is needed for Sifting and EAs/RLS.



(b) Classical algorithms and simple EAs.



(c) Jump variants.

Figure 1 (a) Wall-clock times averaged over the same 100 random instances with increasing n. Evolutionary algorithms were stopped when no improvement was found throughout $n^{1.5}$ subsequent generations. The costs for initialising the crossing matrix for the EAs and the Sifting algorithm are subtracted and shown separately. (b) & (c) Difference between the final evolved crossing number (for EAs) or the returned crossing number (for deterministic algorithms) and the optimal crossing number plotted over generations for classical algorithms and evolutionary algorithms. The plots show averages taken over a suite of instances.

Previous theoretical work [1] suggests that jump is the most effective mutation operator, which we confirm empirically. When given enough time, Jump-RLS/EA almost find nearly-optimal solutions, see Figure 1(b). We verify with statistical significance (using the Wilcoxon rank sum test [8]) that swaps are worse than exchange, which are in turn worse than jumps on the tested instances. The jump-operator also clearly outperforms all other state-of-the-art algorithms when given enough time.

While the jump-algorithms show the best performance, their running times are amongst the highest, see Figure 1. We improve the convergence-speed of Jump-RLS by not performing jumps at random, but by scanning for *acceptable jumps* (i.e. not increasing the crossings number), which does not increase the expected running time asymptotically. We propose three different strategies to make a choice among the acceptable moves found by the algorithm: Performing the first acceptable jump (*JFIRLS*), scanning all jumps and selecting an acceptable one uniformly at random (*JRIRLS*), and choosing the best jump (*JSRLS*). We tested the three algorithms on the same datasets. We verified with statistical significance that the JFIRLS is worse than the other two variants, which show roughly the same performance, see Figure 1(c). The JRIRLS and the JSRLS converge up to 100 times faster than a normal Jump-RLS or Jump-EA on these instances, which coincides with a factor of n.

References

1 Jakob Baumann, Ignaz Rutter, and Dirk Sudholt. Evolutionary computation meets graph drawing: Runtime analysis for crossing minimisation on layered graph drawings. In Xiaodong Li and Julia Handl, editors, *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2024, Melbourne, VIC, Australia, July 14-18, 2024*. ACM, 2024. To appear. doi:10.1145/3638529.3654105.

2 Camil Demetrescu and Irene Finocchi. Removing cycles for minimizing crossings. *ACM J. Exp. Algorithmics*, 6:2–es, 2001. doi:10.1145/945394.945396.

3 Peter Eades and Nicholas C. Wormald. Edge crossings in drawings of bipartite graphs. *Algorithmica*, 11(4):379–403, 1994. doi:10.1007/BF01187020.

4 Michael Jünger and Petra Mutzel. 2-layer straightline crossing minimization: Performance of exact and heuristic algorithms. *J. Graph Algorithms Appl.*, 1(1):1–25, 1997. doi:10.7155/jgaa.00001.

5 Christian Matuszewski, Robby Schönfeld, and Paul Molitor. Using sifting for k-layer straightline crossing minimization. In *Graph Drawing, 7th International Symposium, GD'99, Stirín Castle, Czech Republic, September 1999, Proceedings*, pages 217–224. Springer, Springer, 1999. doi:10.1007/3-540-46648-7_22.

6 Hiroshi Nagamochi. An improved bound on the one-sided minimum crossing number in two-layered drawings. *Discrete & Computational Geometry*, 33:569–591, 2005. doi:10.1007/s00454-005-1168-0.

7 Pietro S. Oliveto, Jun He, and Xin Yao. Time complexity of evolutionary algorithms for combinatorial optimization: A decade of results. *International Journal of Automation and Computing*, 4(3):281–293, 2007. doi:10.1007/S11633-007-0281-3.

8 Frank Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83, 1945.