

1 PACE 2024 Team: studentgroupfuberlin

2 **Garvin Konopka** ✉

3 Freie Universität Berlin, Germany

4 **Colin Alexander Voigt** ✉

5 Freie Universität Berlin, Germany

6 **Joshua Alexander Hanheiser** ✉

7 Freie Universität Berlin, Germany

8 — Abstract —

9 In the context of the PACE 2024 challenge, this project presents a solver designed to minimize
10 crossings in bipartite graphs for the exact track. The primary goal is to reduce the number of edge
11 crossings when a bipartite graph is drawn, with one partition on the left and the other on the right.
12 We approach this problem using Integer Linear Programming (ILP), leveraging linear constraints
13 and optimization techniques to achieve minimal crossings. Additionally, we detect and eliminate
14 cycles by adding specific constraints to the ILP model. This work is a student submission.

15 **2012 ACM Subject Classification** Theory of computation Design and analysis of algorithms

16 **Keywords and phrases** PACE 2024, one-sided crossing minimization, ILP, Cycle elimination

17 **Digital Object Identifier** 10.4230/LIPIcs.CVIT.2016.23

18 **1** Problem Description and Approach

19 **1.1** Problem

20 In the one-sided crossing minimization problem, we are given a bipartite graph $G = (V, E)$,
21 consisting of a vertex set V and an edge set E . The graph is bipartite, meaning that V can
22 be partitioned into two disjoint subsets V_1 and V_2 (thus, $V = V_1 \cup V_2$, $V_1 \cap V_2 = \emptyset$, and
23 $E \subseteq V_1 \times V_2$). The nodes in V_1 are arranged in a linear order and placed in one layer, while
24 the nodes in V_2 are placed in another layer parallel to the first. Edge crossings between V_1
25 and V_2 depend on the sequence of nodes in these two partitions.

26 In the one-sided variation of this problem, the objective is to arrange the nodes in V_2
27 while keeping the order of V_1 fixed, such that the total number of edge crossings is minimized
28 [3].

29 **1.2** Approach

30 Our approach to solving the one-sided crossing minimization problem involves the use of
31 Integer Linear Programming (ILP). The goal is to minimize edge crossings by leveraging
32 linear constraints and optimization techniques. Initially, we do not impose any restrictions
33 or constraints on the y variables, which represent the ordering of nodes in V_2 . Defining all
34 constraints at once is not feasible due to the complexity and size of the problem. As a result,
35 cycles can occur in the graph. To address this, a significant aspect of our method is the
36 detection and elimination of these cycles through the addition of specific constraints to the
37 ILP model. This iterative process of adding constraints allows us to achieve a more optimal
38 arrangement of the nodes in V_2 , reducing the overall number of crossings. However, this
39 approach is still memory and runtime inefficient, especially with many crossings and initial
40 cycles.



© Garvin Konopka and Colin A. Voigt and Joshua A. Hanheiser ;
licensed under Creative Commons License CC-BY 4.0

42nd Conference on Very Important Topics (CVIT 2016).

Editors: John Q. Open and Joan R. Access; Article No. 23; pp. 23:1–23:3



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

41 **2** Algorithm Description

42 **2.1** Input Parsing

43 The input consists of graph edges and the number of nodes in each partition. The edges are
44 parsed and stored in a list, while the degree of each node in the second partition is tracked.

45 **2.2** ILP Problem Formulation

46 An ILP problem is formulated where the objective is to minimize the sum of binary crossing
47 variables. The variables represent potential crossings between edges, and constraints ensure
48 that each pair of edges either crosses or does not cross based on their relative positions.
49 This method adapts the approach from [2], who derived an ILP formulation for minimizing
50 crossings in multi-layer graphs. However, we simplify the problem by not considering crossing
51 variables directly but instead focusing on cycle elimination to achieve the desired order in V_2 .

52 **2.3** Adding Constraints for Cycle Elimination

53 Initially, we do not impose any restrictions or constraints on the y variables, which represent
54 the ordering of nodes in V_2 . Defining all constraints at once is not feasible due to the
55 complexity and size of the problem. As a result, cycles can occur in the graph. To address
56 this, a significant aspect of our method is the detection and elimination of these cycles
57 through the addition of specific constraints to the ILP model. This constraints are the same
58 that are presented i This iterative process of adding constraints allows us to achieve a more
59 optimal arrangement of the nodes in V_2 , reducing the overall number of crossings.

60 **2.4** Solving the ILP

61 The ILP is solved iteratively. After each solution, the graph is updated, and cycles are
62 detected and eliminated until an optimal solution is reached. The problem is solved using
63 the PULP_CBC_CMD solver with cycle constraints added iteratively. This approach
64 ensures that the solution remains feasible and progressively improves as more constraints are
65 introduced to eliminate cycles.

66 **2.5** Topological Sorting of Nodes

67 Once the ILP provides an optimal solution, the nodes are sorted topologically. Nodes with
68 zero in-degree are processed first, ensuring that the final order respects the dependencies
69 dictated by the ILP solution [1].

70 **3** Experimental Results

71 The proposed method was tested on several bipartite graphs provided by the PACE 2024
72 challenge. The results demonstrate significant reductions in edge crossings compared to naive
73 methods. The iterative cycle elimination and constraint addition proved effective in finding
74 optimal solutions efficiently. However, there is considerable potential for further optimization
75 in both memory usage and runtime performance. The current implementation represents the
76 results achieved within a 7-week timeframe.

77 It is important to note that only the medium test sets (excluding 9 of the 60 test
78 sets) could be solved within an acceptable time frame and memory usage with the current

79 approach. The public test instances were found to be significantly more challenging and
80 remain unsolved with the current methodology, highlighting the need for further development
81 and optimization.

82 **4 Conclusion**

83 This project presents an effective approach to minimizing edge crossings in bipartite graphs
84 using integer linear programming (ILP) and cycle elimination techniques. Combining these
85 methods provides a robust solution suitable for large graphs with complex structures. The
86 iterative process of adding constraints after detecting cycles ensures that we can handle the
87 problem's complexity without defining all constraints upfront, which is infeasible.

88 Despite achieving significant reductions in edge crossings, there is considerable potential
89 for further optimization in both memory usage and runtime performance. The current
90 implementation represents the results achieved within a 7-week timeframe. Notably, only the
91 medium test sets (excluding 9 of the 60 test sets) could be solved with the current approach,
92 while the public test instances remain unsolved, indicating the need for further development
93 and optimization.

94 Future work could explore more efficient cycle detection methods, advanced ILP-solving
95 techniques and optimization strategies to handle larger and more complex graphs. Addition-
96 ally, addressing the limitations encountered with the public test instances could provide a
97 pathway for achieving more comprehensive solutions in this domain.

98 **5 Source Code**

99 Our contribution's source code has been published under the MIT License and can be found
100 at <https://git.imp.fu-berlin.de/voic00/appalgo-rose24/>. Our login name from op-
101 til.io is „studentgroupfuberlin“. This projekt is also available under [https://zenodo.org/](https://zenodo.org/records/12282167)
102 [records/12282167](https://zenodo.org/records/12282167).

103 **References**

- 104 **1** Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to*
105 *Algorithms*. MIT press, 2009.
- 106 **2** Andrei Munteanu, Camelia-Mihaela Pintea, and D. Dumitrescu. Hierarchical optimization for
107 multiple crossings minimization. *European Journal of Operational Research*, 202(3):622–634,
108 2009.
- 109 **3** PACE Challenge. Pace challenge 2024. <https://pacechallenge.org>, 2024. Accessed: 2024-
110 06-21.