

EMBEDDING METHODS INTO VNS TO SOLVE THE CROSSING MINIMIZATION PROBLEM

Bruna Cristina Braga Charytitsch

Universidade Federal de São Paulo (Unifesp)

Av. Cesare Mansueto Giulio Lattes, 1201 - Eugênio de Mello, SJC - SP, 12247014

bcbbraga@unifesp.br

Mariá Cristina Vasconcelos Nascimento

Instituto Tecnológico de Aeronáutica (ITA)

Praça Marechal Eduardo Gomes, 50 - Vila das Acacias, SJC - SP, 12228-900

mariah@ita.br

RESUMO

O problema de minimização de cruzamento (CMP, do inglês *crossing minimization problem*) é um problema de *graph drawing* (GD) para o qual foram dedicadas diversas pesquisas nos últimos trinta anos ou mais. No entanto, poucos estudos investigaram a aplicação de técnicas de aprendizado de máquina (AM) para resolver esse problema. Além disso, as estratégias existentes consideram aprendizado ponta-a-ponta e não utilizam-se de hibridizações com métodos de solução robustos, como as metaheurísticas. Em consonância com essa lacuna, este artigo apresenta uma metaheurística *variable neighborhood search* que utiliza informações fornecidas por métodos de aprendizagem por representação em grafos. Experimentos computacionais foram realizados com dois conjuntos de instâncias disponíveis em repositório. Os resultados indicaram que a incorporação de estratégias de AM ao VNS forneceu soluções de alta qualidade.

PALAVRAS CHAVE. *Graph Drawing*, Otimização Combinatória, Técnicas de Aprendizado Profundo.

Tópicos: AM - Aprendizado de Máquina, MH - Metaheurísticas, OC – Otimização Combinatória

ABSTRACT

The crossing minimization problem (CMP) is a graph drawing (GD) problem for which a substantial amount of research over the last thirty or more years has been dedicated. However, few studies have investigated the application of machine learning (ML) techniques to solve this problem. Moreover, the existing approaches consider end-to-end learning, not focusing on hybridizing ML with heuristic methods as metaheuristics. In line with this, this paper introduces the variable neighborhood search heuristic that uses information provided by graph representation learning methods. We carried out computational experiments with two publicly available sets of instances. The results indicate that incorporating ML strategies into VNS provided high-quality solutions.

KEYWORDS. Graph Drawing. Combinatorial Optimization. Deep Learning techniques.

Paper topics: ML - Machine Learning, MH - Metaheuristics, CO - Combinatorial Optimization.

1. Introduction

Crossing Minimization Problem (CMP) is a well-known GD problem [Pach e Tóth, 2000; Chen e Lee, 1998]. The CMP aims at defining a 2D graph visualization that minimizes the number of edge crossings. This problem has been investigated in the last thirty or more years [Binucci et al., 2019]. However, to the best of our knowledge, ML techniques have not been explored to help the search process of MHs.

Nevertheless, some studies focusing on end-to-end ML techniques for GD problems can be found in the literature. Vieira et al. [2015] presented a literature review of the use of ML techniques for GD. The authors observe that such techniques have not been fully investigated for these problems. Wang et al. [2019] and Safarli et al. [2020] introduced recent works involving the application of ML to GD problems. Wang et al. [2019] explored the possibility of using deep learning (DL) in solving a GD. For this, they map network structures to graph drawings from a graph-LSTM-based effective approach. Safarli et al. [2020] interpret a GD problem as a multi-agent reinforcement learning problem. Using this interpretation, graph nodes are said agents with a reward function. An aesthetically acceptable graph layout is obtained from the maximization of this reward.

The combination of Machine Learning (ML) and metaheuristics (MHs) for solving combinatorial optimization problems (COPs) has become an important object of study in different research communities in recent years [Song et al., 2019; Bengio et al., 2021]. Karimi-Mamaghan et al. [2022] reviewed the recent advances in using ML techniques to improve MHs for solving COPs. The authors presented an exhaustive list of the COPs already approached by the combination of ML and MHs, but it does not include any graph drawing (GD) problem.

This paper proposes a variable neighborhood search (VNS) metaheuristic to the CMP considering input information extracted by graph representation learning (GRL) algorithms [Hamilton et al., 2018]. The key idea behind GRL is to extract information from the latent graph structure to determine node embeddings. The goal of this paper is to investigate different GRL strategies to propose a learning-based VNS [Mladenovic e Hansen, 1997], in a low-level integration between ML and MHs. For this, three GRL algorithms, DeepWalk [Perozzi et al., 2014], Node2Vec [Grover e Leskovec, 2016], and GraphSAGE [Hamilton et al., 2017], will be used to provide the initial solution of the introduced VNS. Computational experiments were performed aiming to investigate which GRL method is more efficient for the initialization of VNS. Three data sets composed of 46, 88 and 57 instances were employed in the experiments. Besides, random coordinates were obtained to check the efficiency of the use of node embeddings as initial solutions of the CMP. Both DeepWalk and Node2Vec methods proved to be promising to generate initial solutions to the CMP.

2. Crossing minimization problem

Graph drawing is an active research topic that covers challenging problems involving the automatic layout of graphs [Kaufmann e Wagner, 2001]. Binucci et al. [2019] described ten reasons to get interested in this subject aiming to encourage readers to explore more about associated problems. The paper encompasses the basic research on GD besides outlining some application domains that include areas related to information visualization, software engineering, model-based design, automated cartography, social sciences and molecular biology.

Fundamental aesthetic criteria are symmetry maximization, area minimization, crossing minimization, edge bends minimization, angular resolution maximization, edge

length minimization, the uniform spatial distribution of the vertices, and others [Battista et al., 1998]. Depending on the aesthetic criterion adopted, different GD problems arise and could be explored. In this context, minimizing edge crossings, the so-called crossing minimization problem (CMP), has been identified as the main target in graph visualization [Ware et al., 2002]. The CMP is a well-known NP-hard COP. In the literature, substantial research over the last thirty or more years has been dedicated to this problem. Basically, the CMP aims at defining a GD that minimizes the number of edge crossings.

3. VNS to the CMP

This section presents the proposed algorithm based on VNS to solve the CMP. Three DL-based graph embedding techniques were used to obtain initial solutions for the problem. The main purpose of this study is to assess the different approaches, and identify if there is a more efficient DL method for the task of providing initial solutions for the introduced VNS to the CMP.

3.1. Embedding methods

In solving graph-based problems using GRL approaches, a graph is represented in numerical vectors, called graph embeddings. Conventionally, given a graph G , a graph embedding embeds G , nodes in G or subgraph from G , to low vector space, while preserving the structural properties of G [Hamilton et al., 2018]. To generate the embeddings in the proposed VNS, three GRL algorithms were selected: Node2Vec, DeepWalk and GraphSAGE. A brief description of each of these methods follows.

Node2Vec: Node2Vec is an algorithm composed of two stages. In the first stage, 2nd-order random walks are used to obtain different lists of node ids from a graph. Then, the set of all generated lists of node ids is employed to learn node vector embeddings.

DeepWalk: Similar to Node2Vec, DeepWalk employs short random walks each of them with a $(2t + 1)$ -length, where t is the number of nodes to be swept from a central node. Considering a specific path, the probability of observing the t -nodes before and t -nodes after the central node u_j is maximized. Formally, the algorithm conserves higher-order pair of nodes proximity by maximizing $\log Pr(u_{j-t}, \dots, u_{j-1}, u_{j+1}, \dots, u_{j+t} | Y_j)$. An autoencoder based on a dot-product learns the node embeddings that would rebuild the graph edges [Goyal e Ferrara, 2018].

GraphSAGE: It is an inductive algorithm mainly facing large graphs with substantial node feature information. Most embedding algorithms are transductive and consequently do not generalize to new nodes previously unknown. In opposition, GraphSAGE can provide node embeddings to unknown nodes. For this, a graph or set of graphs is trained and, GraphSAGE can be used to originate embeddings whether the graph and the training data have the same attribute composition.

3.2. Variable Neighborhood Search

Variable neighborhood search (VNS) is a metaheuristic employed to solve combinatorial optimization problems. It investigates the neighborhoods around a given solution. Then, if neighbor solutions have better quality than the one under investigation, they replace the current solution. The local search procedure is iterative for the method to obtain local optima.

3.3. Proposed local search movements

In the local search of the proposed VNS, where the graph arcs are described as line segments, the CMP is interpreted as a problem whose goal is to reduce the intersections

between line segments. In this context, local search movements were proposed aiming at decreasing the number of intersections between segments and, consequently, diminishing the number of crossings between arcs in a directed graph¹. Let $G = (V, A)$ be a directed graph where $(x, y) \in \mathbb{R}^2$ are the node coordinates. The CMP can be seen as the line segment intersection problem [Saalfeld, 1987]. Let S and S' be two bi-dimensional line segments (arcs). Consider their start and end points as $P_0 = (x_0, y_0)$ to $P_1 = (x_1, y_1)$ and $P'_0 = (x'_0, y'_0)$ to $P'_1 = (x'_1, y'_1)$, respectively. The intersection between the two line segments occurs when the angular coefficient values u and $v \in [0, 1]$ are such that

$$P_0 + u(P_1 - P_0) = P'_0 + v(P'_1 - P'_0) \quad (1)$$

By expanding Equation (1), u and v are defined by:

$$u = \frac{(x'_1 - x'_0)(y_0 - y'_0) - (y'_1 - y'_0)(x_0 - x'_0)}{(y'_1 - y'_0)(x_1 - x_0) - (x'_1 - x'_0)(y_1 - y_0)}, \quad v = \frac{(x_1 - x_0)(y_0 - y'_0) - (y_1 - y_0)(x_0 - x'_0)}{(y'_1 - y'_0)(x_1 - x_0) - (x'_1 - x'_0)(y_1 - y_0)},$$

where S and S' are parallel if the denominators of the equations for u and v are zero. The line segments are coincident if the denominator and numerator for the equations for u and v are zero. There is an intersection point if both u and v are between zero and one.

Furthermore, consider X and Y axes representing two layers containing the node coordinates of a graph in the Cartesian plane. The Cartesian coordinates of the starting points or the arc tails and the endpoints or arc heads are displayed in Figure 1. Once identified the line segment intersection or arc crossings, the idea is to find an ordering of the coordinates on the layers such that the arc crossings are eliminated or reduced. For this, the method performs some movements aiming to minimize the arc crossings (Figure 2). If the arc tail coordinates of a pair of arcs, (x_0, y_0) and (x'_0, y'_0) , are permuted, then the arc crossing can be eliminated. Figure 2a illustrates this case considering the initial setting in Figure 1. Initially, there is an intersection between P and P' whose the arc tails are $P_0 = (2, 1)$ and $P'_0 = (-3, 2)$ and arc heads being $P_1 = (-2, -2)$ and $P'_1(3, -3)$, respectively. The permutation between the tail coordinates of the two arcs results in a new setting where the arc tail of arc P that was $P_0 = (2, 1)$ becomes $P_0 = (-3, 2)$ and the arc tail of P' that was $P'_0 = (-3, 2)$ becomes $P'_0 = (2, 1)$. Observe that these movements eliminate the initial arc crossing.

Following similar reasoning, besides swapping the positions of the arc tail coordinates, one may swap the arc heads coordinates of two different arcs, (x_1, y_1) and (x'_1, y'_1) , as illustrated in Figure 2b. The arc head of P that was $P_1 = (-2, -2)$ becomes $P_0 = (3, -3)$ and the arc head of P' that was $P'_1(3, -3)$ becomes $P'_0 = (-2, -2)$. Another possibility would be the permutation between the arc tails and arc heads coordinates. By randomly selecting two nodes from the graph, disregarding of it being an arc tail or an arc head, the move consists of swapping only the y-ordinates of the node coordinates or only the x-ordinates of the node coordinates. Figure 2c presents the case where the arc tail P_0 and the arc head P'_1 were selected. Initially, the arc tail coordinate was $P_0 = (2, 1)$ and the arc head coordinate was $P'_1 = (3, -3)$. After permuting the y-ordinates between both nodes coordinates, the arc tail coordinate becomes $P_0 = (2, -3)$ and the arc head coordinate becomes $P'_1 = (3, 1)$. Figure 2d shows the case where two arc heads P_1 and P'_1 were selected. Initially, the arc heads coordinates were $P_1 = (-2, -2)$ and $P'_1 = (3, -3)$. After permuting the

¹If the graph is not directed, just insert directions at the edges randomly.

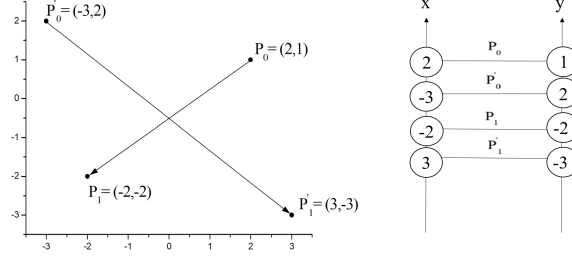


Figure 1: Arc crossing scheme in the bi-dimensional space and two layers representation.

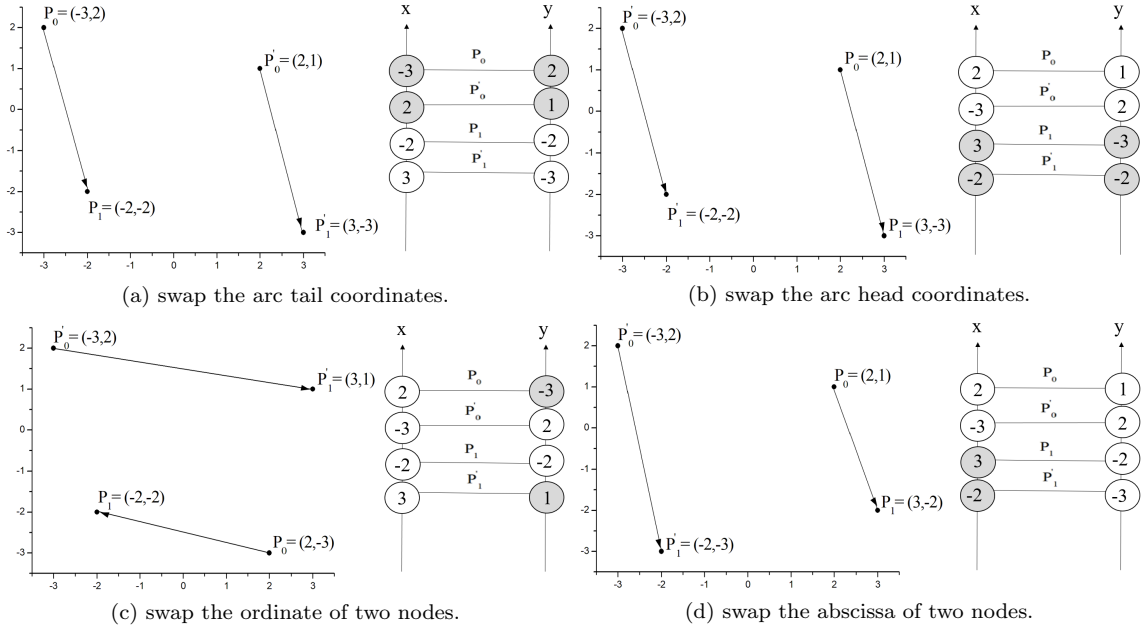


Figure 2: Exchange neighbourhood for edges crossing elimination.

x-abscissa between both node coordinates, the arc head coordinates become $P_1 = (3, -2)$ and $P_1' = (-2, -3)$. In both movements, the arc crossing was eliminated.

3.4. Proposed VNS algorithm

Algorithm 1 presents a pseudocode of the introduced VNS for the CMP. Algorithm 2 refers to the local search in the proposed VNS. The main algorithm has as input the information from the input graph and the number of neighborhoods r . The matrix $\mathbf{M}$ represents the initial solution and stores in the first column the graph nodes and in the second and third columns, their (x, y) embeddings coordinate, respectively. The output is the final solution matrix $\mathbf{S}_{best}$ (similar in structure to $\mathbf{M}$) besides the number of crossings of this solution.

First, $\mathbf{M}$ and $\mathbf{S}_{best}$ are initialized by a node embedding algorithm that extracts an $\mathbb{R}^2$ representation of the nodes. Then, on line 2, a list $\mathbf{c}_{best}$ composed of m sublists is created. Each sublist represents a graph edge and it contains the edges that cross the edge to which the sublist represents. Then, on line 3, the objective function of the initial solution produced by the node embedding algorithm is calculated. The objective function of a solution is given by function $\text{ObjFunction}()$ according to Equation (1).

Algorithm 1: VNS

Input: Graph G ; neighborhood structure number r
Output: Solution S_{best} ; minimum crossing f_{best}

```

1  $S_{best}, M \leftarrow$  Generate node embeddings;
2 Update the set  $c_{best}$  considering the coordinates produced in the previous step;
3  $f_{best} \leftarrow$  ObjFunction ( $S_{best}$ );
4 while stopping criteria have not been reached do
5    $k \leftarrow 1$ ;
6   while  $k \leq r$  do
7      $S_{nbh} \leftarrow S_{best}; f_{nbh} \leftarrow f_{best}$ ;
8     for  $j = 1 \dots k$  do
9       Randomly choose between movements  $swap_y$ ,  $swap_x$  and  $swap$  and
        apply the movement to  $S_{nbh}$ , at a pair of nodes randomly
        chosen; /* Shake movement */
10      Update the list  $c_{nbh}$  according to the new  $S_{nbh}$ ;
11       $f_{nbh} \leftarrow$  ObjFunction ( $S_{nbh}$ );
12       $(c_{nbh}, S_{nbh}, f_{nbh}) =$  LocalSearch ( $c_{nbh}, S_{nbh}, f_{nbh}$ )
13      if ( $f_{nbh} < f_{best}$ ) then
14         $S_{best} \leftarrow S_{nbh}; f_{best} \leftarrow f_{nbh}; k \leftarrow 1$ ;
15      else
16         $k \leftarrow k + 1$ 
17      end
18    end
19  end
20 end
```

Algorithm 2: Local Search

Input: Solution S ; crossing vector c ; objective function value of S , f
Output: Updated solution S ; updated c and f

```

1  $S_{copy} \leftarrow S$ ;
2 for all arc  $e$  with some crossing do
3   Random select between head and tail and assign it to variable end;
4   Obtain the end of the arc  $e$  and the end of the first arc in the subset of
     $c_{copy}$  associated with  $e$  and apply the swap to them;
5   Update  $c_{copy}$  according to  $S_{copy}$ ;
6    $f_{copy} \leftarrow$  ObjFunction ( $S_{copy}$ );
7   if ( $f_{copy} < f$ ) then
8      $c \leftarrow c_{copy}; S \leftarrow S_{copy}; f \leftarrow f_{copy}$ ;
9   else
10     $c_{copy} \leftarrow c; S_{copy} \leftarrow S; f_{copy} \leftarrow f$ ;
11  end
12 end
```

The main loop then starts. The stopping criteria of this routine are either the number of iterations defined by the user has reached or there is no improvement in the best overall objective function value in the last 1000 iterations. The iterator that controls the neighborhood to be investigated, k , is initialized on line 5. Another loop whose stopping criterion is k to be greater than a predefined number of neighborhoods, r , starts on line 6. On line 7, the solution of the neighborhood is updated with the best overall solution. Then, a move is updated if the neighbor solution has a better objective solution value. Then, one of the following neighborhoods is randomly selected for the shaking movement: **swap** - coordinates of a pair of nodes are permuted; **swap_y** - y ordinates of a pair of nodes are permuted; **swap_x** - x abscissas of a pair of nodes are permuted. A local search is applied to the shaken solution that replaces the best overall solution if its objective function value is better.

Algorithm 2 shows the first improvement-based Local Search proposed. In summary, the method evaluates if swapping positions of a pair of crossing edges would provide a better objective function.

4. Computational Experiments

We carried out computational tests to compare the results achieved by VNS considering the initial solutions provided by DeepWalk (DW), Node2Vec (N2V), and GraphSAGE (GS). The average solution of 10 independent executions is reported. The analysis of the experiment results was divided into four parts: (i) analysis of the initial crossings; (ii) analysis of the algorithm results; (iii) analysis of algorithm iterations; and (iv) random coordinates results. Details are described in the next sections. The experiments were conducted on a computer with a 3.10 GHz Intel Core i3 processor with 4GB of RAM. All procedures were implemented in Python. The next section briefly describes the collection of selected instances.

4.1. Selected Instances

Three data sets were employed in the experiments. One refers to 46 planar graphs from the reference house of graphs [Brinkmann et al., 2013]. The remain data sets (Rome and North) [Pastore et al., 2019] are available on the webpage <https://grafo.etsii.urjc.es/optsimcom/mmaccp>. The parameters tested in the experiments are presented in the next section.

4.2. Parameter Tuning

The planar graphs set was used to fine-tune the parameters of the VNS. We chose such graphs due to the prior knowledge of optimal solutions (a layout without edge crossing).

For the parameter tuning, the following set of values was tested: $r \in \{2, 3, 4, 5, 6, 7, 8\}$ and $\{1000, 2000, 3000, 4000, 5000, 6000, 7000, 8000\}$ for the number of iterations. The parameter tuning of the proposed VNS was performed considering as node embedding strategy the algorithms DW, N2V, and GS discussed in earlier sections of this paper. The combinations that provided the mode of the best solutions in 10 runs were selected: 2 and 8000 for DW, 8 and 7000 for N2V, and 8 and 4000 for GS.

4.3. Results and discussions

To generate the initial node coordinates in the proposed Algorithm, three methods were employed: DW, N2V, and GS. For simplification purposes, we will refer directly to the method instead of using the sentence “the proposed VNS with the method”. In this section, the experiment results are presented and discussed. A comparative analysis is performed considering the different graph embedding methods employed to initialize the algorithm.

4.3.1. Analysis of initial crossings

The first analysis aims to verify the quality of the initial solutions generated by the graph embedding methods. Figure 3 shows these results in both Rome and North datasets.

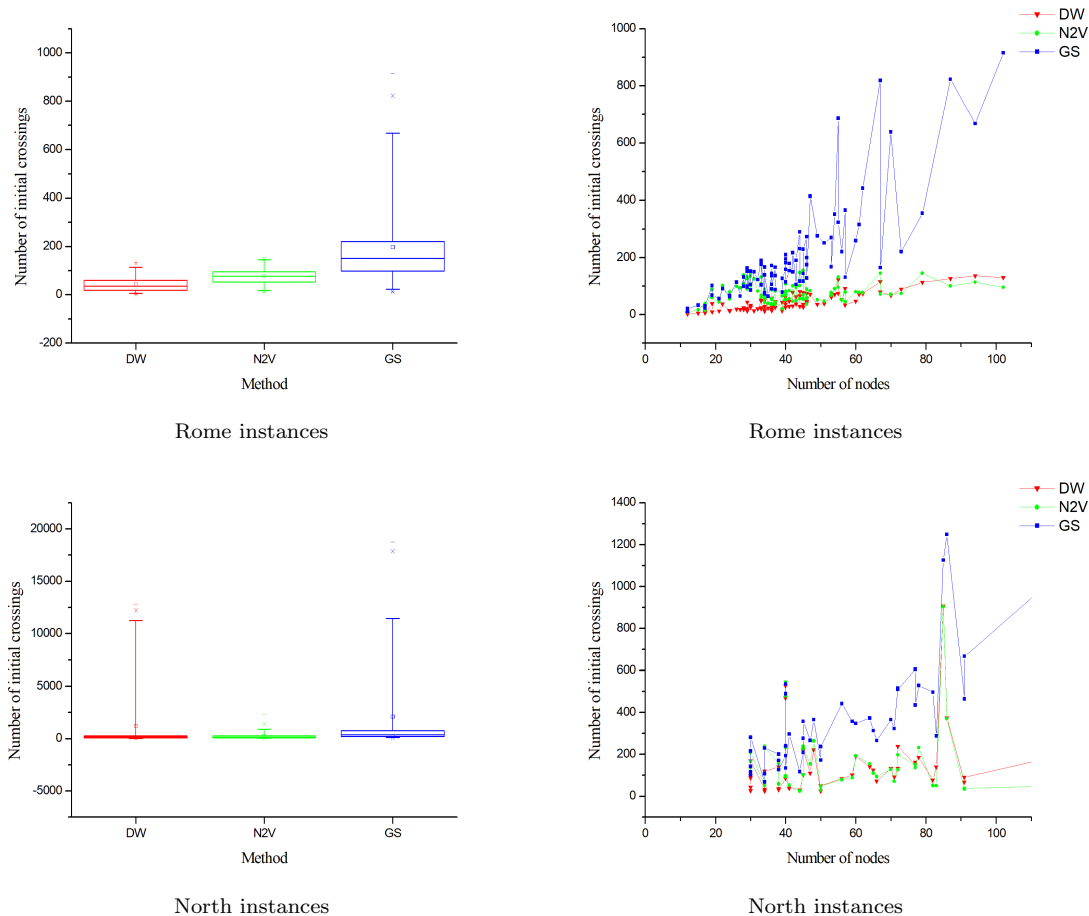


Figure 3: Box-plots with the number of initial crossings of each method and line charts showing the behavior of the number of initial crossings of each method versus the number of graph nodes.

Regarding the Rome instances, the box-plots show that DW and N2V showed a similar data dispersion. On the other hand, GS presented a greater variation when compared to the other methods. This behavior can also be seen on the line chart, where there is a greater dispersion of data as the number of nodes increases. Moreover, the results suggest that in about 91% of the instances, DW presented the best results when compared with the other methods. In approximately 9% of the instances, N2V outperformed both the DW and GS methods.

Figure 3 shows that, in North instances, the data dispersion of the results found by DW and N2V was similar. In this case, the GS method presented a slightly greater variation in comparison to the other methods. This behavior can also be observed in the line chart. The results obtained suggest that in about 51% of the instances N2V obtained the best results, whereas DW outperformed N2V and GS in about 47% of the instances. Both N2V and DW performed similarly in approximately 2% of the instances.

4.4. Analysis of final results

This section presents the mean, variance, minimum (min) and maximum (max) number of crossings of the three proposed versions of VNS. Figure 4 and Table 1 present an overview of the results obtained in Rome and North instances.

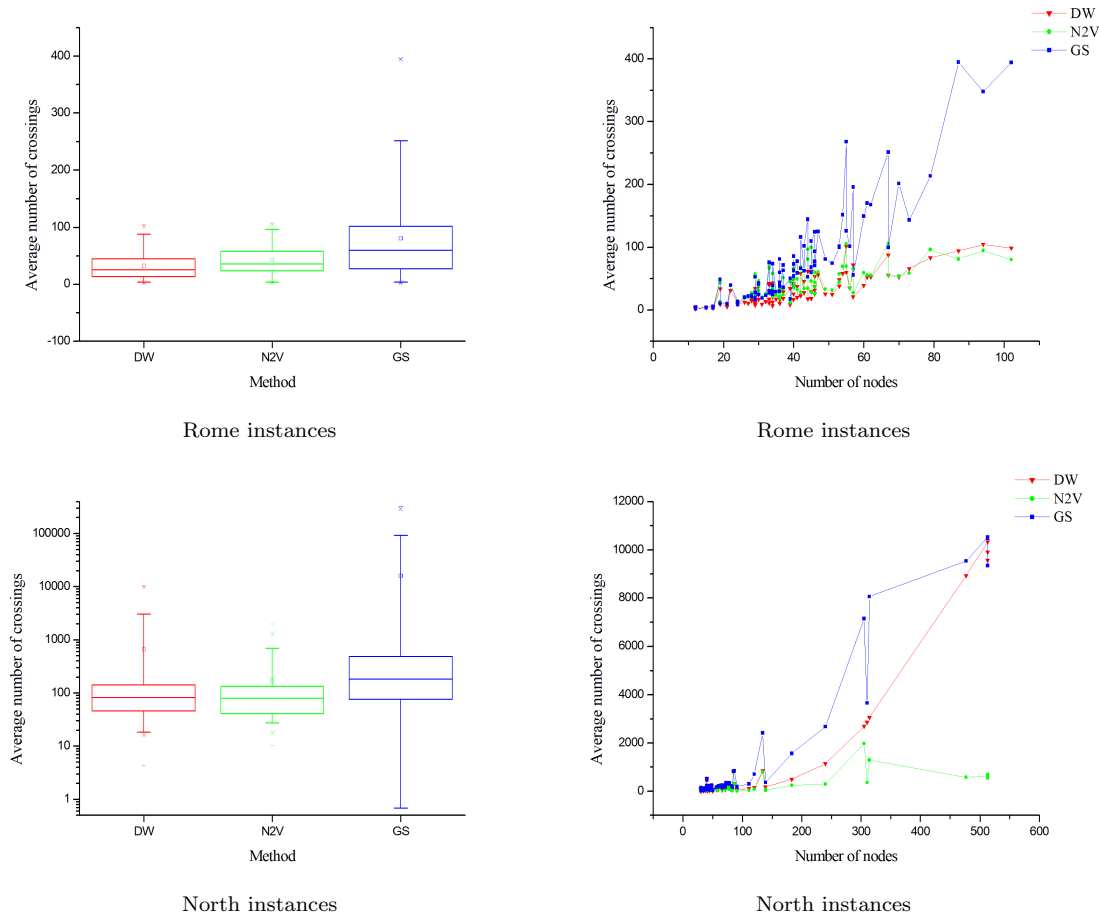


Figure 4: Box-plots of the average number of crossings of each method and a line chart showing the behavior of the same values versus the number graph nodes. Note: North's box-plot is on logarithmic scale.

Table 1: Statistics of the results in Rome and North instances.

Method	Rome				North			
	mean	variance	max	min	mean	variance	max	min
DW	32.70	5.28	35.44	29.53	984.24	13541.60	1039.93	912.21
N2V	41.93	18.24	47.84	35.56	235.16	94.46	244.81	224.56
GS	80.63	202.75	98.06	63.22	1333.64	24593.61	1451.96	1216.44

According to Table 1, in the Rome instances, DW and N2V outperformed GS. More specifically, DW presented the lowest crossing numbers in $\sim 84\%$ of the cases, while N2V and GS in $\sim 14\%$ and $\sim 1\%$ of the cases, respectively. Also, in $\sim 1\%$ of the cases, the results were the same for all methods. According to Figure 4, considering the North instances, the average number of crossings in the box-plot chart suggests that DW and N2V performed similarly. Moreover, as can be seen in the line chart, the dispersion of the results gets larger as the number of nodes of the graph increases. In general, in North instances,

VNS with the N2V method as the initial solution presented the lowest crossing numbers in $\sim 51\%$ of the cases against $\sim 39\%$ and $\sim 0\%$ of DW and GS, respectively. In $\sim 2\%$ of the cases, DW and GS achieved the same results. In $\sim 5\%$ of the cases, DW and N2V obtained the same results. Finally, in $\sim 3\%$ of the cases, all the methods achieved the same results.

4.5. Analysis of the number of iterations

This section presents an analysis of the number of iterations in which the minimum number of crossings was obtained in the search process of the three versions of VNS. The maximum number of iterations of DW, N2V and GS was 8000, 7000, and 4000, respectively. Table 2 shows some statistics with regard to the number of iterations to find the best solution by the different methods.

Table 2: Average of iterations took by DW, N2V and GS to find the best solution.

Method	Rome				North			
	mean	variance	max	min	mean	variance	max	min
DW	631	277912.97	1522	107	619	323827.02	1632	67
N2V	668	400101.48	1864	69	905	614704.79	2222	124
GS	481	276684.73	1412	39	568	304919.89	1521	50

One can observe that neither the average nor the highest number of iterations in which the best solution was obtained reached the maximum allowed number of iterations. Such results suggest that the upper bound for the number of iterations defined for each method could be reduced by more than 80% of its value.

4.6. Random coordinates results

This experiment considered the Rome instances. In addition to generating embeddings coordinates, random coordinates were obtained to check the efficiency of using node embedding as initial solutions for the VNS to solve the CMP. Again, we report the average of 10 runs of all methods. First, we compared the initial solutions found by the graph representation learning methods with random coordinates (RDM). Figure 5 shows the box-plots of the four strategies.

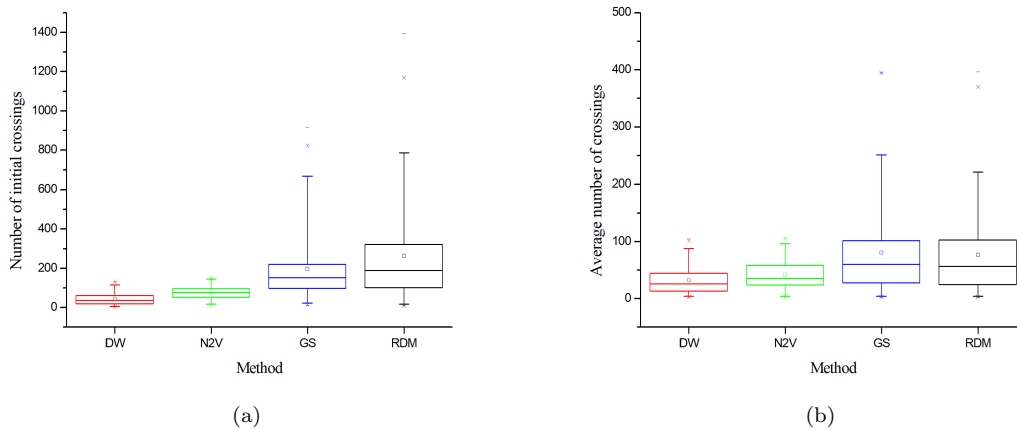


Figure 5: Box-plots with the (a) number of initial crossings of each method and, (b) the average number of crossings from each method when applied as the initial solution in VNS.

The results of this first set of box-plots (left figure) indicate that DW achieved better results than the other methods. In particular, RDM obtained the worst results. Following, the second set of box-plots in Figure 5 (right figure) shows the results obtained by VNS considering as initial solutions the coordinates obtained by DW, N2V, GS and

RDM. Note that the parameter tuning of DW, N2V and GS provided different values of r and the maximum number of iterations. Therefore, we present the results of VNS with RDM as the initialization method considering the best solutions considering the three sets of parameter values. The results indicate that DW and N2V as initialization methods of VNS performed significantly better than RDM. On the other hand, the results of GS in VNS were comparable with RDM in VNS.

4.7. Final remarks

This paper presented three forms to provide initial solutions for the crossing minimization problem (CMP), a graph drawing problem. They are based on classical graph representation learning algorithms: DeepWalk, GraphSAGE and Node2Vec. The methods generated initial solutions for a VNS metaheuristic, also proposed in this paper.

Computational experiments with a wide variety of graphs indicate that DeepWalk and Node2Vec were promising strategies to generate initial solutions to the CMP. The assessment considered the quality of the initial solutions, as well as the number of iterations required for the algorithm to converge to the local optima. In particular, DeepWalk stood out as an initialization method.

In future works, we suggest more in-depth studies to explore the use of node embeddings as initial solutions for the CMP. In line with this, the investigation of the same methods to initialize different metaheuristics might give a better insight into the potential of graph representation learning for this benchmark graph drawing problem.

Acknowledgements

The authors thank the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior/Brasil (CAPES) – Finance Code 001 – for the financial support.

References

- Battista, G., Eades, P., Tamassia, R., e Tollis, I. (1998). *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall PTR, USA.
- Bengio, Y., Lodi, A., e Prouvost, A. (2021). Machine learning for combinatorial optimization: a methodological tour d’horizon. *European Journal of Operational Research*, 290 (2):405–421.
- Binucci, C., Brandes, U., Dwyer, T., Gronemann, M., von Hanxleden, R., van Kreveld, M., Mutzel, P., Schaefer, M., Schreiber, F., e Speckmann, B. (2019). 10 reasons to get interested in graph drawing. In *Computing and Software Science: State of the Art and Perspectives*, p. 85–104, Cham. Springer International Publishing.
- Brinkmann, G., Coolsaet, K., Goedgebeur, J., e Melot, H. (2013). House of graphs: A database of interesting graphs. *Discrete Applied Mathematics*, 161(1):311–314.
- Chen, H.-F. e Lee, D. (1998). On crossing minimization problem. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 17:406–418.
- Goyal, P. e Ferrara, E. (2018). Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems*, 151:78–94. ISSN 0950-7051.
- Grover, A. e Leskovec, J. (2016). Node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD ’16*, p. 855–864, New York, NY, USA. Association for Computing Machinery. ISBN 9781450342322.

- Hamilton, W. L., Ying, R., e Leskovec, J. (2018). Representation learning on graphs: Methods and applications.
- Hamilton, W. L., Ying, Z., e Leskovec, J. (2017). Inductive representation learning on large graphs. In *NIPS*.
- Karimi-Mamaghan, M., Mohammadi, M., Meyer, P., Karimi-Mamaghan, A. M., e Talbi, E.-G. (2022). Machine learning at the service of meta-heuristics for solving combinatorial optimization problems: A state-of-the-art. *European Journal of Operational Research*, 296(2):393–422.
- Kaufmann, M. e Wagner, D. (2001). *Drawing Graphs: Methods and Models*, volume 2025. Springer, Berlin.
- Mladenovic, N. e Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research*, 24(11):1097–1100. ISSN 0305-0548.
- Pach, J. e Tóth, G. (2000). Which crossing number is it anyway? *Journal of Combinatorial Theory, Series B*, 80(2):225–246.
- Pastore, T., Martinez-Gavara, A., Napoletano, A., Festa, P., e Marti, R. (2019). Tabu search for min-max edge crossing in graphs. *Computers & Operations Research*, 114: 104830.
- Perozzi, B., Al-Rfou, R., e Skiena, S. (2014). Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, p. 701–710, New York, NY, USA. Association for Computing Machinery. ISBN 9781450329569.
- Saalfeld, A. (1987). It doesn't make me nearly as cross - some advantages of the point-vector representation of line segments in automated cartography. *Geographical Information Systems*, 1(4):379–386.
- Safarli, I., Zhou, Y., e Wang, B. (2020). Interpreting graph drawing with multi-agent reinforcement learning. *arXiv preprint arXiv:2011.00748*.
- Song, H., Triguero, I., e Özcan, E. (2019). A review on the self and dual interactions between machine learning and optimisation. *Progress in Artificial Intelligence*, 8:143–165.
- Vieira, R. S., Nascimento, R., Dantas, H. A., e da Silva, W. B. (2015). The application of machine learning to problems in graph drawing a literature review. In *Proceedings of International Conference on Information, Process, and Knowledge Management*, p. 112–118.
- Wang, Y., Jin, Z., Wang, Q., Cui, W., Ma, T., e Qu, H. (2019). Deepdrawing: A deep learning approach to graph drawing. *IEEE transactions on visualization and computer graphics*, 26(1):676–686.
- Ware, C., Purchase, H., Colpoys, L., e McGill, M. (2002). Cognitive measurements of graph aesthetics. *Information Visualization*, 1(2):103–110.