GRASP with Path Relinking for 2D-Bandwidth Minimization Problem

M.A. Rodríguez-García
Universidad Rey Juan Carlos
Móstoles, Spain
miguel.rodriguez@urjc.es

A. Duarte
Universidad Rey Juan Carlos
Móstoles, Spain
abraham.duarte@urjc.es

J. Sánchez-Oro
Universidad Rey Juan Carlos
Móstoles, Spain
jesus.sanchezoro@urjc.es

ABSTRACT

The graph bandwidth minimization problem is an interesting problem that has become relevant in a wide range of domains. Networks communication, VLSI layout designs, parallel algorithms simulations, matrix decomposition, are some of which areas where the reduction of the bandwidth is very significant. The problem consists of embedding a graph G into a line with the aim of minimizing the maximum distance between adjacent vertices. In this paper, we are focused on the 2D bandwidth minimization variant, which considers embedding the graph in a two-dimensional grid instead of in a line. Specifically, we study the problem deeply analyzing its complexity and considering a survey of different approximate algorithms for graphs. The review concludes outlining the conceptual basis of the heuristic technique that we plan to apply to this graph problem.

KEYWORDS

2D bandwidth minimization, GRASP, Path Relinking

ACM Reference Format:
M.A. Rodríguez-García, A. Duarte, and J. Sánchez-Oro. 2018. GRASP with Path Relinking for 2D-Bandwidth Minimization Problem. In *Proceedings of International Conference on Learning and Optimization Algorithms: Theory and Applications (LOPAL'18).* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

The problem of minimizing the bandwidth of a matrix (BMP) has been studied from several perspectives in the literature. It was firstly defined in the 1950s with the aim of reducing the computing time for matrix operations such as inversions and determinants [3]. The optimization consists of moving all the nonzero entries of the matrix into a band around the diagonal by reordering the rows and columns of the matrix.

The main practical application of the problem resides in solving large linear systems. For instance, for a matrix of dimension n and bandwidth b the traditional algorithm for Gaussian elimination presents a complexity of $O(n^3)$ [13]. However, it can be performed in $O(nb^2)$ using matrix bandwidth minimization when b is smaller than n. BMP presents several additional applications such as saving large hypertext media or improving data storage [1], large-scale power transmission systems, very large scale integration designs, or geophysics [17], among others.

Let $G = (V, E)$ be an undirected and connected graph, being V the set of vertices and E the set of edges, with $|V| = n$ and $|E| = m$. A labeling σ of G consists of assigning the integers in the range $[1 \dots n]$ to the vertices of G, where each vertex v is assigned a different label $\sigma(v)$. Let $N(v)$ be the set of adjacent vertices to v. If we now deploy the graph in a horizontal line where is vertex $v \in V$ is located at position $\sigma(v)$ the bandwidth of v, denoted as $B(v)$, is defined as:

$$B(v) \leftarrow \max_{u \in N(v)} |\sigma(v) - \sigma(u)|$$

The bandwidth of a graph G with respect to a given labeling σ, $B(G, \sigma)$ is defined as the maximum bandwidth among all the vertices. More formally,

$$B(G, \sigma) = \max_{v \in V} B(v)$$

The BMP then consists of finding a labeling $\sigma^\star$ with the minimum value of $B(G, \sigma^\star)$ among all possible labellings for G. The resulting permutation indicates the reordering in rows and columns that must be performed to the original matrix to minimize the bandwidth of the matrix.

Figure 1(a) shows an example graph G with 6 vertices and 9 edges and two possible solutions for the BMP. The first one, σ_1, depicted in Figure 1(b), corresponds to a lexicographical distribution of the vertices in G, while the second one, σ_2, corresponds to a different ordering of the vertices in G. The number under each vertex represents its bandwidth. For instance, for solution σ_1, $B(A) = |\Phi(A) - \sigma(E)| = |1 - 5| = 4$, and $B(D) = |\sigma(D) - \sigma(C)| = |2 - 1| = 1$. Following this evaluation, the bandwidth of solution σ_1 is $B(G, \sigma_1) = 4$, while the bandwidth of solution σ_2 is $B(G, \sigma_2) = 3$. Therefore, σ_2 is better solution than σ_1, since it presents a smaller bandwidth value.

In this paper we focus on a different variant of the problem, named two-dimensional bandwidth (2DBMP) [4, 11]. Specifically, we consider that the nodes of the graph needs to be located in a two-dimensional square grid of size $p \times p$ instead that in a horizontal line. Given a graph G, a solution φ for the 2DBMP consist of assigning, to each vertex $v \in V$, a pair of integer values (i, j), with $1 \leq i, j \leq p$, where each vertex v is assigned to a different label $\varphi(v)$. In this variant, we need to define a new distance function to evaluate the bandwidth of each vertex. As stated in previous works [12], we consider the rectilinear distance, also known as L_1-norm. Let v be the vertex located at position $\varphi(v) = (v_x, v_y)$, and vertex u the

LOPAL'18, Mayo 2018, ENSIAS, Rabat, Morocco

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00

DOI: 10.1145/3230905.3230953

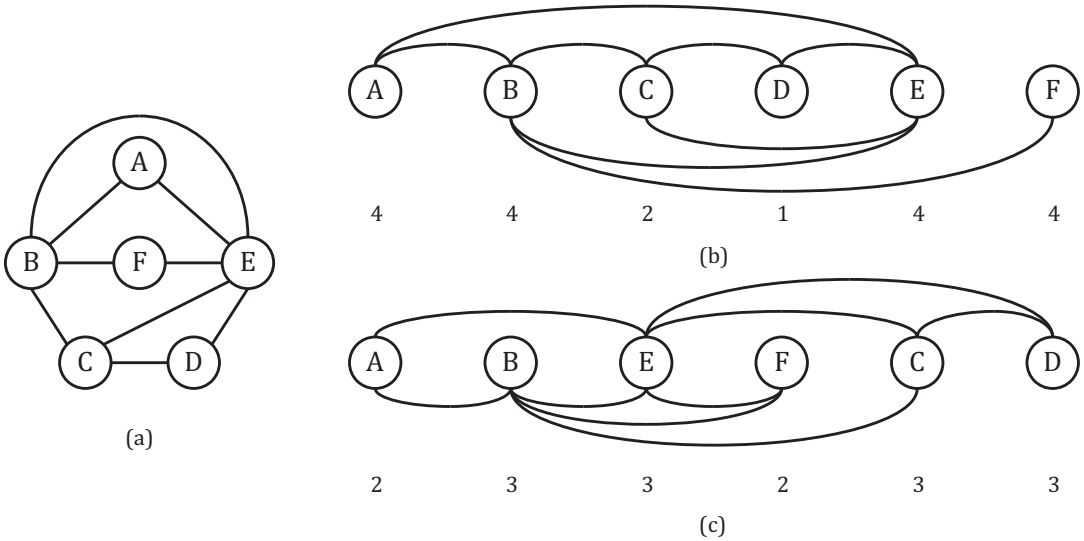

Figure 1: (a) Example graph with 6 vertices and 9 edges and (b)(c) two possible linear layouts with the bandwidth of each vertex depicted under it.

one located at position $\varphi(u) = (u_x, u_y)$, the L_1-norm distance is evaluated as:

$$d(v,u) = |v_x - u_x| + |v_y - u_y|$$

The bandwidth of a vertex is evaluated analogously to the linear variant of the problem:

$$B(v) \leftarrow \max_{u \in N(v)} d(v,u)$$

Finally, the 2DBMP consists of finding a labeling with the minimum bandwidth value, as in the linear variant.

Figure 2(a) depicts the same example graph as the one presented in Figure 1(a). Figures 2 (b) and (c) illustrates two possible solutions for the 2DBMP, φ_1 and φ_2, respectively. The bandwidth for each vertex is represented with a number close to it. For example, the bandwidth for vertex A is $B(A) = d(A,E) = |1-2| + |1-2|$, since $\varphi_1(A) = (1,1)$ and $\varphi_1(E) = (2,2)$. Then, the bandwidth of solution φ_1 is equal to 3 while the bandwidth of solution φ_2 is equal to 2, meaning that φ_2 is a better solution, since the bandwidth is smaller.

This problem has been mainly ignored from a heuristic point of view. Two different models for the problem are presented in [12], while [14] studies the square-root rule for the 2DBMP in order to improve the evaluation of the bandwidth in typical graphs. The problem has several applications in Very Large Scale Integration (VLSI) design [15] and there exist some bounds for specific types of graphs [6].

The rest of the paper is structured as follows: Section 2 presents the heuristic algorithms (constructive method and local search) proposed for obtaining high quality solutions in the 2DBMP; Section 3 describes the Greedy Randomized Adaptive Search Procedure developed in order to improve the heuristic algorithms; and finally Section 5 gathers the conclusions derived from the research and suggests some future lines of work for this problem.

2 HEURISTICS FOR THE 2DBMP

Heuristic procedures are designed for solving problems throughout an intuitive method in which the problem structure can be leveraged and exploited in order to obtain solutions of a reasonable quality, usually requiring small computing times [16]. This Section presents two heuristic methods for the 2DBMP: a constructive method and a local search.

2.1 Constructive method (C1)

Constructive methods are designed for generating a feasible solution for a given problem. The method proposed in this work starts from an empty solution where no vertex has a position assigned and iteratively assigns an available position to each vertex.

Algorithm 1 Constructive(G)

```
1:  φ = ∅
2:  CL ← V
3:  v ← SelectRandom(CL)
4:  φ ← φ ∪ {v}
5:  CL ← CL \ {v}
6:  while CL ≠ ∅ do
7:      u ← argmin_{u∈CL} g(u, φ)
8:      φ ← φ ∪ {u}
9:      CL ← CL \ {u}
10: end while
11: return φ
```

Algorithm 1 presents the pseudocode of the proposed greedy constructive method. The method starts by constructing an empty solution φ (step 1). Then, the list of candidate vertices CL is created containing all the vertices of the graph (step 2). The first vertex is selected at random among all candidates (step 3), adding it to the solution and removing it from CL (steps 4-5). It is worth mentioning

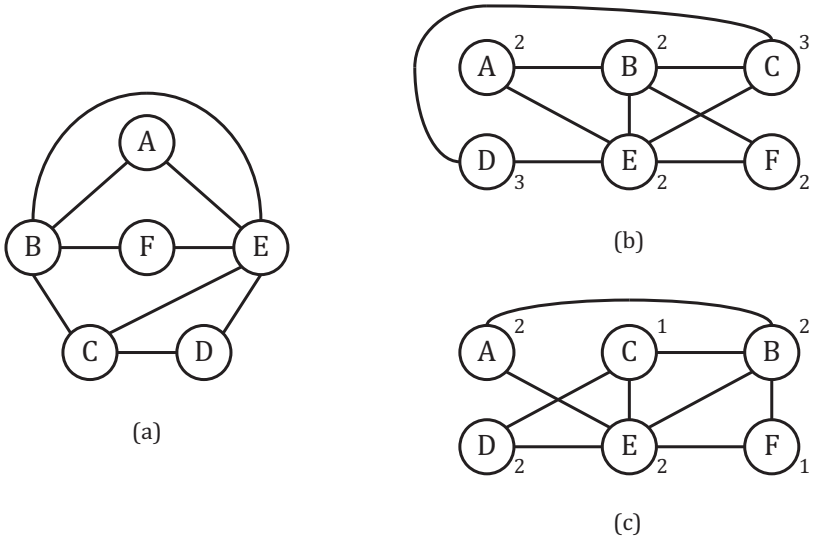

Figure 2: (a) Example graph with 6 vertices and 9 edges and (b)(c) two possible grid layouts with the bandwidth of each vertex depicted close to it.

that adding a vertex to a solution means that this vertex receives the first available location in the grid, following a left to right and top-down approach, i.e., the first one receives label (1,1), the second one label (1,2), and so on. Then, the method iterates until all vertices has been assigned to a position in the grid (steps 6-10). In each iteration, the constructive procedure selects the best candidate to be assigned following a greedy function value (step 7). The greedy function selected as the evaluation of the objective function if the vertex is added to the solution. More formally,

$$g(v, \varphi) \leftarrow B(\varphi \cup \{v\})$$

Then, the vertex with the smallest greedy function value (i.e., the one that produces the best solution) is added to the solution and removed from the candidate list (steps 8-9). The method ends when the candidate list becomes empty, returning a feasible solution for the 2DBMP (step 11).

2.2 Local search

The main aim of a local search procedure is to find a local optimum with respect to a given feasible solution that has been generated using a constructive method. The local optimum is usually found by performing basic movements over the initial solution that leads to improved solutions. In this paper, we propose two local search methods that differ in the type of movement performed over the incumbent solution.

The first one, named LS1, is based on insertion moves. The move $insert(\varphi, v_i, j)$ consists of removing vertex v_i from its current position in solution φ and inserting it at position j. More formally, if the initial solution is represented as:

$$\varphi = \{v_1, v_2, ..., v_{i-1}, v_i, v_{i+1}, ..., v_{j-1}, v_j, v_{j+1}v_n\}$$

the move $insert(\varphi, v_i, j)$ will result in the following solution φ_{ins}:

$$\varphi_{ins} = \{v_1, v_2, ..., v_{i-1}, v_{i+1}, ..., v_{j-1}, v_i, v_j, v_{j+1}v_n\}$$

The second local search procedure focuses on interchange moves. The move $interchange(\varphi, v_i, v_j)$ consists of moving the vertex v_i to the position of vertex v_j and vice versa. Specifically, starting from the same initial solution φ, the solution φ_{int} resulting from performing the move $interchange(\varphi, v_i, v_j)$ is represented as:

$$\varphi = \{v_1, v_2, ..., v_{i-1}, v_j, v_{i+1}, ..., v_{j-1}, v_i, v_{j+1}v_n\}$$

Additionally, we can define the neighborhood of a given solution for each move aforementioned. Specifically, neighborhoods $N_{ins}(\varphi)$ and $N_{int}(\varphi)$ contains all the solutions that can be reached by insertion and interchange moves, respectively. More formally,

$$\begin{aligned} N_{ins}(\varphi) &= \{\varphi' \leftarrow insert(\varphi, v_i, j) \ \forall\, v_i \in V,\ 1 \le j \le n\} \\ N_{int}(\varphi) &= \{\varphi' \leftarrow interchange(\varphi, v_i, v_j) \ \forall\, v_i, v_j \in V\} \end{aligned}$$

Both neighborhoods are explored following a first improvement approach. In particular, the search starts exploring the corresponding neighborhood randomly. The search only accepts improvement moves, so every time a better solution is found in the neighborhood, the movement is performed, restarting the search from the improved solution. Both local search procedures stops when no improvement move is found in the complete neighborhood.

3 GREEDY RANDOMIZED ADAPTIVE SEARCH PROCEDURE

Heuristic methods easily get stuck in local optima, becoming a difficult task to escape from them, since they are oriented to perform moves that lead only to better solutions. Metaheuristics are high-level methodologies with the ability to lead and modify other heuristics in order to explore a larger portion of the search space [9].

Greedy Randomized Adaptive Search Procedure (GRASP) is a population-based metaheuristic that was firstly introduced by Feo and Resende [7] and formally defined in Feo et al. [8]. GRASP has been successfully applied in several hard optimization problems in recent years [2, 5]

GRASP is a multi-start metaheuristic with two well-differenced stages: solution construction and local search. The former consist of a greedy, randomized, and adaptive construction of a solution, while the latter is based on locally improving the solution generated in order to obtain a local optima. Both stages are repeated until reaching a stopping criterion, which is usually a predefined number of iterations, returning the best solution found during the search.

The constructive stage proposed in this work is a modification of the constructive procedure proposed in Section 2.1, while the local search stage is the one previously presented in Section 2.2.

In order to escape from local optima, GRASP introduces some randomness in the constructive method. Specifically, instead of selecting the best vertex to be added to the solution (step 7 of Algorithm 1), the method selects one vertex at random among the most promising ones available. Specifically, the procedure evaluates the vertices with the minimum (g_{min}) and maximum (g_{max}) value of the greedy function as follows:

$$g_{min} \leftarrow min_{v \in V} g(v, \varphi)$$

$$g_{max} \leftarrow max_{v \in V} g(v, \varphi)$$

Then, a threshold th is evaluated to limit which vertices are the most promising ones as follows:

$$th \leftarrow g_{min} + \alpha \times (g_{max} - g_{min})$$

where α is a parameter of the algorithm that controls its greediness / randomness, which is in the range $0 \leq \alpha \leq 1$. On the one hand, if $\alpha = 0$, the method is completely greedy (as in Section 2.1) since only the most promising vertex can be selected. On the other hand, if $\alpha = 1$, then the method is totally random since any vertex can be selected.

The method then constructs a Restricted Candidate List (RCL) with the most promising nodes of the CL. The RCL contains all the vertices in CL that presents a greedy function value smaller or equal than the previously evaluated threshold. More formally,

$$RCL \leftarrow \{v \in CL : g(v) < th\}$$

Finally, the next vertex to be added to the solution is randomly selected from the RCL.

The GRASP algorithm proposed in this work performs a predefined number of iterations, which is a parameter of the algorithm. In each iteration, the algorithm construct a new solution using the aforementioned constructive method. Then, the solution is improved with the local search method in order to obtain a local optima. Finally, the algorithm returns the best solution explored among all the constructions and improvements.

4 RESULTS

As far as we know, there is no previous heuristic approaches for the 2DBMP. Therefore, in order to test the quality of the proposal, we have considered a set of instances for which the optimal value is known by construction. In particular, we have considered bidimensional grids, whose optimal objective function value is always equal to one, as stated in [12]. The benchmark instances consists of several grids whose size ranges from 3x3 from 20x20. All the algorithm have been implemented in Java 8, and the experiments have been conducted in a 3.20 GHz Intel i5-3470 CPU with 8 GB 1600 MHz DDR3 RAM.

The experiment is intended to analyze the performance of the constructive procedure coupled with each local search method proposed. Furthermore, the constructive procedure requires from an α parameter that controls the greediness / randomness of the method, so we have considered the following values for $\alpha = \{0.25, 0.50, 0.75\}$. Notice that the randomness of the method increases with the value of α parameter, being $\alpha = 1$ totally random and $\alpha = 0$ completely greedy. Table 1 summarizes the results obtained in the experiment regarding to the average objective function value of the solutions discovered, OF; the execution time measured in seconds, Time (s); the average deviation with respect to the best solution found in the experiment, Dev(%); and, finally, the number of times that a method reaches the best solution of the experiment.

The obtained results provide clear evidence about how the randomness affect directly the quality of the solutions obtained. Both local search techniques have reached worst results in the experiments when we established the α level to 0.25 that restrict the algorithm's behavior. On the contrary, higher values that provide more randomness have collected better results reducing the execution time and delivering better results. This is mainly because increasing the randomness of the search allows the algorithm to explore further regions of the search space, where the local search is able to find better solutions.

5 CONCLUSIONS AND FUTURE WORK

This paper tackles the two-dimensional bandwidth minimization problem by means of two heuristic procedures and a metaheuristic algorithm that helps them to escape from local optima. The algorithm proposed is able to obtain high-quality solutions in short computing times, which makes it suitable for the considered problem. The randomization of the constructive method inside the GRASP framework is able to outperform the results obtained by the purely greedy constructive procedure, showing the relevance of the metaheuristic.

Future lines of research are focused on evaluating the algorithm in instances in which the optimal value is known by construction in order to test the quality of the proposal. Furthermore, it would be interesting to evaluate different kind of metaheuristics (trajectorial, bioinspired, etc.) to check which one is better adapted to the problem.

ACKNOWLEDGEMENTS

This work has been partially founded by Ministerio de Economía y Competitividad with grants refs. TIN2015-65460-C2-2-P and S2013/ICE-2894, respectively.

REFERENCES

[1] M. W. Berry, B. Hendrickson, and P. Raghavan. 1996. Sparse Matrix Reordering Schemes for Browsing Hypertext. *Lectures in Applied Mathematics* 32 (1996), 99–123.

GRASP			OF	Time (s)	Dev (%)	#Best
Constructive	α	Local Search				
C1	0.25	Insert moves (LS1)	14.06	33.07	21.02	2
		Interchange moves (LS2)	13.83	0.36	16.95	7
	0.50	Insert moves (LS1)	13.22	31.15	15.69	5
		Interchange moves (LS2)	12.94	0.35	12.10	10
	0.75	Insert moves (LS1)	12.39	32.23	4.30	5
		Interchange moves (LS2)	12.78	0.34	11.15	8

Table 1: Results obtained from the performed experiments

[2] V. Campos, R. Martí, J. Sánchez-Oro, and A. Duarte. 2014. GRASP with path relinking for the orienteering problem. *Journal of the Operational Research Society* 65, 12 (2014), 1800–1813.

[3] P. Z. Chinn, J. Chvátalová, A. K. Dewdney, and N. E. Gibbs. 1982. The bandwidth problem for graphs and matrices - a survey. *Journal of Graph Theory* 6, 3 (1982), 223–254.

[4] J. Díaz, J. Petit, and M. Serna. 2002. A Survey of Graph Layout Problems. *ACM Comput. Surv.* 34, 3 (2002), 313–356.

[5] A. Duarte, J. Sánchez-Oro, M. G. C. Resende, F. Glover, and R. Martí. 2015. Greedy randomized adaptive search procedure with exterior path relinking for differential dispersion minimization. *Information Sciences* 296 (2015), 46 – 60.

[6] C. M. Duran. 2006. Two Dimensional Bandwidth REU in Mathematics at CSUSB. (2006).

[7] T. A. Feo and M. G. C. Resende. 1989. A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters* 8, 2 (1989), 67 – 71.

[8] T. A. Feo, M. G. C. Resende, and S. H. Smith. 1994. A Greedy Randomized Adaptive Search Procedure for Maximum Independent Set. *Operations Research* 42, 5 (1994), 860–878.

[9] F. Glover. 1986. Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research* 13, 5 (1986), 533 – 549.

[10] P. Hansen and N. Mladenović. 2006. First vs. best improvement: An empirical study. *Discrete Applied Mathematics* 154, 5 (2006), 802 – 817.

[11] Y.-L. Lai and K. Williams. 1999. A survey of solved problems and applications on bandwidth, edgesum, and profile of graphs. *Journal of Graph Theory* 31, 2 (1999), 75–94.

[12] L. Lan and L. Yixun. 2010. Two models of two-dimensional bandwidth problems. *Information Processing Letters* 110, 11 (2010), 469 – 473.

[13] A. Lim, B. Rodrigues, and F. Xiao. 2006. Heuristics for matrix bandwidth reduction. *European Journal of Operational Research* 174, 1 (2006), 69 – 91.

[14] L. Lin and Y. Lin. 2011. Square-root rule of two-dimensional bandwidth problem. *RAIRO-Theor. Inf. Appl.* 45, 4 (2011), 399–411.

[15] B. Monien and H. Sudborough. 1988. Comparing interconnection networks. In *Mathematical Foundations of Computer Science*, M. P. Chytil, V. Koubek, and L. Janiga (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 138–153.

[16] T. A. J Nicholson. 1971. *Optimization in industry.* London : Longman.

[17] E. Piñana, I. Plana, V. Campos, and R. Martí. 2004. GRASP and path relinking for the matrix bandwidth minimization. *European Journal of Operational Research* 153, 1 (2004), 200 – 210.