New Metaheuristic Algorithms to Reduce the Bandwidth of Real-World Graphs

Alexandre Augusto Alberto Moreira de Abreu[1,2,3(✉)] and Sanderson L. Gonzaga de Oliveira[1]

[1] Universidade Federal de São Paulo, São José dos Campos, SP, Brazil
{alexandre.abreu,sanderson.oliveira}@unifesp.br
[2] Instituto Tecnológico de Aeronáutica, São José dos Campos, SP, Brazil
[3] Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina, Canoinhas, SC, Brazil

Abstract. This paper evaluates new heuristics for the bandwidth reduction problem, two BRKGA-based algorithms, and NCHC+, an adaptation of the NCHC heuristic. We experimentally evaluated these heuristics against the most successful graph-theoretic algorithms and state-of-the-art metaheuristic algorithms using a benchmark set of large-scale graphs from the literature. The results confirm that FNCHC+ yields the best results for real-world graphs. Furthermore, the proposed heuristics outperformed DRSA, the previous state-of-the-art metaheuristic for small-sized graphs.

Keywords: Heuristic methods · Bandwidth reduction problem · Biased Random-Key Genetic Algorithm · Sparse graphs · Metaheuristics

1 Introduction

The bandwidth minimization problem is a prominent $\mathcal{NP}$-hard [14,43] combinatorial optimization problem with numerous scientific and engineering applications [7,33]. Garey et al. [13] demonstrated that the problem is $\mathcal{NP}$-hard for trees with maximum degree 3. Additionally, researchers have established particular classes of graphs in which the problem remains $\mathcal{NP}$-hard [44]. Harper [29] examined the problem in the context of error-correcting codes and formally defined it [30].

In an undirected graph, the goal is to minimize the maximum absolute difference between the labels of adjacent vertices. This task belongs to the broader class of graph layout problems, which optimize a specific objective function by assigning distinct natural numbers from the set $\{1, 2, \ldots, |V|\}$ to the vertices of a graph $G = (V, E)$.

To accelerate linear system solvers in sparse numerical linear algebra, practitioners typically employ low-cost graph-theoretic algorithms [21,25–28]. These methods relabel the vertices of a graph to improve the non-null patterns in

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2027
O. Gervasi et al. (Eds.): ICCSA 2026, LNCS 16767, pp. 161–178, 2027.
https://doi.org/10.1007/978-3-032-30491-9_11

the associated matrix. Specifically, the Reverse Cuthill-McKee (RCM) [15] and Gibbs-Poole-Stockmeyer (GPS) [17] methods are the most successful graph-theoretic algorithms for this purpose. In contrast, this study targets domains lacking stringent runtime constraints, making them highly suitable for metaheuristic algorithms. Examples of such applications include VLSI design, network survivability, and efficient sparse graph representation [37], compression of large hypertext media [4], small-world networks [31], visual dataset analysis [42], graph entropy rate minimization [5], mesh layout optimization [8], symbolic model checking [2], and the seriation problem [9]. Particularly in big data, low-memory storage formats are essential for processing large-scale graphs, improving data locality, and avoiding inefficient computations [39].

Consequently, this paper focuses on metaheuristic algorithms for bandwidth reduction. For large-scale graphs (with orders exceeding one million and sizes exceeding millions), FNCHC+ [21,24] is the state-of-the-art metaheuristic algorithm [22]. Conversely, DRSA [50] is the leading metaheuristic algorithm for smaller graphs (of orders up to 1,000 and sizes up to 50,000). However, a previous study [22] indicated that DRSA relies on an outdated benchmark set, and this paper evaluates the heuristic in large-scale graphs.

This paper proposes two heuristics based on the BRKGA metaheuristic employing a new constructive method. The new BRKGA-based heuristics differ chiefly in the fitness functions adopted. The first uses bandwidth as the only fitness function. The second fitness function uses each individual's bandwidth and the number of critical vertices as a tiebreaker. Additionally, the paper evaluates an adaptation of the NCHC heuristic [36], named NCHC+.

This work experimentally evaluates three new heuristics for bandwidth reduction. We compare the proposed heuristics against state-of-the-art metaheuristic algorithms and the most successful graph-theoretic algorithms (RCM [15] and GPS [17]) using a benchmark set of large-scale graphs from the literature [22].

We organized the remainder of the paper as follows. Section 2 defines the problem, and Sect. 3 reviews the most important related work. Section 4 introduces the two novel BRKGA-based heuristics. Section 5 explains the new constructive method used in the BRKGA-based heuristics. Section 6 details the experiment, including an explanation of the NCHC+ heuristic in Sect. 6.4. Subsequently, Sect. 7 reports the computational results. Finally, Sect. 8 presents the conclusions.

2 Problem Definition

Gonzaga de Oliveira [22] defined the vertex labeling of an undirected graph, the bandwidth of a vertex, and the bandwidth of an undirected graph as follows.

Definition 1 (Vertex labeling of an undirected graph). *A vertex labeling L for an undirected graph $G = (V, E)$ is a bijective mapping $L : V \rightarrow \{1, 2, \ldots, |V|\}$. It assigns a unique integer identifier $l(v) \in [1, |V|]$ to each vertex $v \in V$, resulting in the set $L = \{l(v_1), l(v_2), \ldots, l(v_{|V|})\}$.*

Definition 2 (Bandwidth of a vertex in undirected graph). *Given a vertex labeling L, the bandwidth of a vertex v in $G_L = (V, E)$ is $\beta_v(G_L) = 0$ if $\nexists u \mid u \in Adj(G_L, v) \wedge l(u) < l(v)$; otherwise, $\beta_v(G_L) = \max_{u \in Adj(G_L,v) \wedge l(u) < l(v)} [l(v) - l(u)]$, where $Adj(G_L, v) = \{u : \{v, u\} \in E\}$ denotes the adjacent vertices of v.*

Definition 3 (Bandwidth in undirected graph). *The bandwidth of the undirected graph $G_L = (V, E)$ with a vertex labeling L is $\beta(G_L) = \max_{v \in V} [\beta_v(G_L)]$.*

The bandwidth minimization problem seeks to minimize the maximum absolute difference between the labels of adjacent vertices [22]. Figure 1 illustrates this reduction process on an undirected graph. Initially, vertices $v_1, \ldots, v_5$ are labeled as $l(v_i) = i$ for $i \in \{1, \ldots, 5\}$, resulting in $\beta(G_L) = 4$. A subsequent relabeling to $l(v_1) = 5, l(v_2) = 2, l(v_3) = 1, l(v_4) = 3$, and $l(v_5) = 4$ reduces the bandwidth to $\beta(G_L) = 2$.

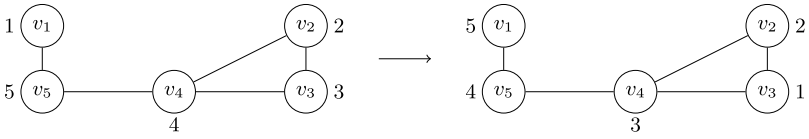

Fig. 1. Bandwidth reduction of an undirected graph from $\beta(G_L) = 4$ (left) to $\beta(G_L) = 2$ (right).

3 Related Work

Since the 1990s, researchers have developed numerous metaheuristic algorithms for bandwidth reduction [22]. Following a recent literature review [22], this section presents the most prominent and recent metaheuristic algorithms for this problem.

Dueck and Jeffs [11] proposed a Simulated Annealing heuristic. Esposito and Tarricone [12] introduced a heuristic based on Tabu Search. Martí et al. [40] proposed another Tabu Search (TS) algorithm that incorporates a frequency-based restart strategy for diversification alongside the tabu list. For vertex label swapping, the heuristic selects from a set of near-critical vertices defined as $C(L) = \{v : \beta_v(G_L) \geq \lambda \cdot \beta(G_L)\}$, where $\lambda \in (0, 1)$. Specifically, the heuristic considers v a critical vertex if its bandwidth is the maximum in the graph G_L, i.e., $\beta_v(G_L) = \beta(G_L)$. In 2001, the TS algorithm became the state-of-the-art metaheuristic algorithm for bandwidth reduction. However, the authors evaluated TS using small-sized graphs with up to 1,000 vertices and 50,000 edges. A recent review [22] classified these graphs as outdated and unrepresentative of modern problem scales and applications. Furthermore, modern computational capabilities make execution times on such small graphs indistinguishable. Consequently,

researchers must evaluate the heuristics on larger-scale graphs to verify whether a heuristic maintains its performance advantage as the problem size increases [32].

Piñana et al. [45] introduced a heuristic based on the Greedy Randomized Adaptive Search Procedures with Path Relinking (GRASP-PR) metaheuristic. The authors compared the bandwidth solutions on 31 graphs with up to 199 vertices and 80 graphs with up to approximately 1,000 vertices. A recent review [22] provided a detailed description of these graphs. The GRASP-PR heuristic emerged as the state-of-the-art method in 2004.

Lim et al. [37] proposed a Genetic Algorithm (GA) and the Node Shift (NS) heuristic, which incorporates a Hill Climbing procedure. After evaluating these heuristics on the benchmark set of Martí et al. [40,45] and 14 additional graphs, the NS algorithm yielded the best bandwidth solutions, becoming the state-of-the-art algorithm. Lim et al. [35] proposed a heuristic based on PSO.

The Node Centroid Hill Climbing (NCHC) and Fast Node Centroid Hill Climbing (FNCHC) heuristics were published in 2007 by Lim et al. [36]. The NCHC heuristic iteratively generates an initial solution via a Breadth-First Search starting from an arbitrary vertex. Subsequently, the heuristic improves the initial solution using a global optimization operator called Node Centroid (NC) and a Hill Climbing (HC) local search. The algorithm applies both the NC and HC procedures NC_{max} times, where NC_{max} is an algorithm parameter.

The NC procedure relabels all vertices toward a central position (centroid) relative to their neighborhood. For this purpose, it computes the metric $\omega(v) = \frac{\sum_{u \in B_\lambda(v)} l(u)}{|B_\lambda(v)|}$ for each vertex $v \in V$, where $B_\lambda(v) = (Adj(v) \cap \{u \in V : |l(u) - l(v)| \geq \lambda \cdot \beta(G_L)\}) \cup \{v\}$. Afterward, it relabels the vertices in non-decreasing order of their $\omega(v)$ values. The HC algorithm swaps labels of near-critical vertices using the candidate list of moves proposed by Martí et al. [40] to reach a local optimum. Similar to the GRASP-PR heuristic [45], NCHC considers the reduction in the number of critical vertices as the acceptance criterion for label swaps. Due to its high computational cost, the algorithm executes the HC procedure only during odd iterations. FNCHC is a faster variant of NCHC, optimized by fine-tuning the parameter λ that defines the near-critical vertex set. In computational tests on the benchmark set of Martí et al. [40,45], NS achieved the best bandwidth results, whereas FNCHC demonstrated lower running times.

Rodriguez-Tello et al. [46] introduced another Simulated Annealing heuristic (SA) in 2008, called SA–δ. Campos et al. [6] introduced a heuristic based on Scatter Search and Tabu Search.

Mladenović et al. [41] introduced the Variable Neighbourhood Search for bandwidth reduction (VNS-band). The heuristic became the state-of-the-art metaheuristic algorithm, achieving superior bandwidth reduction for graphs with orders up to 1,000 and sizes up to 50,000. The authors evaluated the algorithm using the benchmark set of Martí et al. [40,45]. In 2011, Koohestani and Poli [34] employed genetic programming to evolve the RCM algorithm. Although analysts

published several heuristics in subsequent years (e.g., [23]), VNS-band achieved superior results.

In 2015, Torres-Jimenez et al. [50] proposed the Dual Representation Simulated Annealing (DRSA) algorithm. When evaluated on the benchmark set composed of very small graphs from Martí et al. [40,45], DRSA outperformed VNS-band. Its superior results established it as the state-of-the-art metaheuristic algorithm on graphs with orders up to 1,000 and sizes up to 50,000 edges. While subsequent studies introduced other metaheuristic algorithms, such as a hybrid Iterated Local Search-Tabu Search approach [1], DRSA continued to yield superior results.

Silva et al. [47] introduced the first heuristic based on the Biased Random-Key Genetic Algorithm (BRKGA) framework [19] for bandwidth reduction. The authors used randomly generated individuals and the parameterized uniform crossover [48]. In each iteration, new solutions are randomly generated and introduced into the population to increase diversity. DRSA outperformed BRKGA [47] on 43 of the small-sized graphs from the benchmark set of Martí et al. [40,45]. The two novel BRKGA-based heuristics proposed in this paper, named BRKGA* and BRKGA+, differ from the BRKGA-based heuristic introduced by Silva et al. [47] mainly in four aspects: the constructive phase, the use of an encoder, the creation of new solutions, and the fitness function. Specifically, BRKGA* and BRKGA+ employ a novel randomized semi-greedy constructive algorithm (Sect. 5) to generate high-quality solutions at a low computational cost for both the initial population and the newly generated individuals. Afterward, a proposed encoder generates the random keys for these high-quality individuals. In contrast, Silva et al. [47] used randomly generated individuals for the initial population and new solutions, a process that does not require an encoder. Furthermore, while the BRKGA [47] and BRKGA+ heuristics both use the bandwidth value as their fitness function, BRKGA* employs a two-tiered fitness function that prioritizes the bandwidth value but uses the number of critical vertices as a tie-breaking criterion.

In 2022, Gonzaga de Oliveira and Carvalho [24] proposed the ILS-based heuristic for bandwidth reduction (ILS-band), the first metaheuristic algorithm to solve large-scale graphs (exceeding one million vertices) within an acceptable execution time of 1,200 s. Although ILS-band achieved bandwidth reductions comparable to GPS, the former state-of-the-art for large-scale graphs, it demanded longer execution times. In the same study, the authors introduced FNCHC+, a faster refinement of FNCHC [36] that restricts the invocation of the computationally expensive Hill Climbing local search to promising solutions. Computational tests on graphs ranging from 5,940 to 112,985 vertices demonstrated that FNCHC+ [24], GPS [18], FNCHC [36], and ILS-band [24] surpassed VNS-band [41] and DRSA [50]. Furthermore, comprehensive tests on the other 76 large-scale graphs (with orders up to 9,845,725 and sizes up to 57,156,537) established FNCHC+ as the state-of-the-art metaheuristic algorithm for large-scale graphs, despite GPS initially exhibiting shorter running times. Lastly, a subsequent study [21] used 32 large-scale graphs (with orders up to 23,947,347

and sizes up to 130,641,318) and confirmed that FNCHC+ consistently outperforms GPS in both solution quality and computational cost.

In short, DRSA [50] is the state-of-the-art metaheuristic algorithm for reducing the bandwidth of small graphs (i.e., with orders of up to 1,000 and sizes of up to 50,000). FNCHC+ heuristic [21,24] is the state-of-the-art metaheuristic algorithm for large-scale graphs.

4 Two Novel Heuristics Based on the Biased Random-Key Genetic Algorithm Metaheuristic

This section introduces two new bandwidth reduction heuristics based on the Biased Random-Key Genetic Algorithm (BRKGA) metaheuristic [19]. BRKGA employs arrays of real numbers, named random-key arrays. These arrays represent possible solutions to a given problem. A deterministic decoding algorithm associates a random-key array with a feasible solution. It allows the algorithm to calculate the objective function (fitness) for the problem. The method is biased because one of the parents in BRKGA is always selected from the elite group and has a higher probability of passing on its genes.

Algorithm 1 receives the graph $G(V, E)$, the parameters $|P| > 0$, $e > 0$, $r > 0$, $init \in \{0, 1\}$, $\alpha \in [0, 1]$, $p > 0.5$, and the maximum execution time t_{max}. The parameters $|P|$, e, and r define, respectively, the population size, the size of the elite solution set, and the size of the new solution set. The parameters $init$ and α determine the initial vertex and the level of randomness considered in the constructive method (Algorithm 2). The parameter p represents the probability that a child receives genes from a parent selected from the elite set.

Algorithm 1: The pseudocode of two heuristics based on the Biased Random-Key Genetic Algorithm metaheuristic (BRKGA+ and BRKGA*).

Input: graph $G(V, E)$, parameters $|P|$, e, r, α, $init$, p, and maximum runtime t_{max}.
Output: the bandwidth of the best solution found.
Generate an initial population P (Algorithm 2), with parameters $init$ and α;
Encode the individuals of the population P;
Sort the population P;
while (execution time is less than t_{max}) **do**
    Copy the best e individuals from P into P';
    for ($i = 1$ **to** ($|P| - e - r$)) **do**
        Select a random parent from the elite set of P;
        Select a random parent from the population P;
        Generate a single new individual by parameterized uniform crossover with probability p;
        Decode the new individual;
        Insert the new individual into the population P';
    end for
    Insert r new individuals into P' (Algorithm 2), with parameters $init$ and α;
    Encode the r new individuals inserted into the population P';
    Sort the population P';
    $P \leftarrow P'$; `// the new population` P' `becomes population` P
end while
return the bandwidth of the best individual in P;

We implemented two variations of Algorithm 1. We called the first variation BRKGA+. It uses only each individual's bandwidth as a fitness function. The fitness function of the second variation, called BRKGA*, is the bandwidth value of each individual. However, for individuals with the same bandwidth, BRKGA* uses the number of critical vertices as a tiebreaker. This approach is the first difference between Algorithm 1 and the BRKGA heuristic proposed by Silva et al. [47].

In line 1 of Algorithm 1, an initial population P of size $|P|$ is generated by Algorithm 2, presented in Sect. 5, with parameters *init* and α. This new initial population is the second difference between this approach and the approach proposed by Silva et al. [47]. Specifically, Silva et al. [47] employed a set of random solutions as the initial population.

Line 2 encodes the generated individuals (solutions). An individual is a labeling L in an array of random keys. The encoder assigns the value $l(i)/|V|$, where $i = 1, 2, \cdots, |V|$, to each entry i of the random-key array. This approach is the third difference between Algorithm 1 and the BRKGA heuristic [47]. In the BRKGA heuristic [47], individuals' random keys in the initial population are generated randomly in the interval $[0, 1)$. Therefore, BRKGA [47] does not use an encoder that translates a solution (labeling) to the domain of random keys. Line 3 sorts the generated individuals in ascending order of their fitness values.

Algorithm 1 executes lines from 5 to 16 while the execution time is less than t_{max}. Line 5 copies the best e individuals from population P to the next population P'. This approach ensures the survival of the best e individuals.

Algorithm 1 executes lines 7–11 $|P| - e - r$ times. Line 7 randomly selects a parent from the elite set. Line 8 randomly selects another parent from the population P. Line 9 generates a single offspring from the selected parents. The parameterized uniform crossover [48] performs the crossover operation. For each entry in the array of random keys (gene), the algorithm randomly decides from which parent to copy the key value to the offspring. However, it is a biased decision because the algorithm defines a higher probability $p > 0.5$ of selection for the parent from the elite group. This bias encourages the propagation of characteristics from the most promising solutions throughout the evolutionary process.

In line 10, the decoder decodes the new individual. Line 11 inserts it into the population P'. In this study, the decoder sorts the random keys in ascending order and assigns the index corresponding to the original entry of the i-th key before sorting to the label of vertex i, where $i = 1, 2, \ldots, |V|$. Thus, the resulting array is a permutation array corresponding to a labeling for the graph's vertices. This decoder is the simple decoder proposed by Bean [3] and widely adopted in combinatorial optimization problems.

In line 13, the constructive method presented in Algorithm 2 with parameters *init* and α generates a set of r new solutions. This step aims to bring diversity to the population. This approach is the fourth difference with respect to the BRKGA proposed by Silva et al. [47], which generates the new solutions randomly. The encoder encodes the generated r new individuals in line 14. Line

15 sorts the population P' by the value of the fitness function, and it becomes the current population P in line 16. Finally, in line 18, the bandwidth of the best individual of the population P is returned.

5 Constructive Method

Algorithm 2 details the semi-greedy randomized constructive algorithm that the new BRKGA-based heuristics use to create high-quality individuals at low computational cost. Algorithm 1 employs this procedure in lines 1 and 13. The proposed constructive algorithm is a variant of the RCM [15] and the George-Liu [16] algorithms. In addition to the graph $G = (V, E)$, the algorithm requires the parameters *init* and α, which determine the initial vertex and the degree of randomness, respectively.

Algorithm 2: Randomized Semi-Greedy Constructive Method.

```
   Input: graph G = (V, E), parameters init and α.
   Output: labeling L.
 1 Q ← ∅;   visited ← [false, ..., false];
 2 if (init = 0) then  v ← G.RandomVertex();
 3 else  v ← G.GL(G.RandomVertex());
 4 Q.add(v);   visited[v] ← true;
 5 l(v) ← 1;   i ← 2;
 6 while (Q ≠ ∅) do
 7 |   u ← Q.remove();
 8 |   foreach (vertex w ∈ Adj(G, u), sorted by degree in a randomized semi-greedy order) do
 9 |   |   if (¬visited[w]) then
10 |   |   |   Q.add(w);   visited[w] ← true;
11 |   |   |   l(w) ← i;   i ← i + 1;
12 |   |   end if
13 |   end foreach
14 end while
15 return L;
```

In line 1, the constructive algorithm initializes a vertex queue Q and a boolean array *visited* used in its breadth-first search strategy. Depending on the parameter *init*, the procedure either randomly selects a starting vertex $v \in V$ (line 2) or determines a pseudoperipheral vertex v via the George-Liu algorithm [16] (line 3). Subsequently, line 4 enqueues v into Q and marks it as visited. Line 5 then assigns the initial label 1 to v and sets the next available label counter i to 2.

Algorithm 2 iterates through lines 7–13 until the queue Q becomes empty. In line 7, the procedure dequeues the first element and assigns it to u. Line 8 sorts the neighborhood of u in a semi-greedy randomized order based on vertex degrees. Specifically, the algorithm first sorts the adjacent vertices by degree. Then it applies a random shuffle to the initial $\lfloor |Adj(G, u)| \cdot \alpha \rfloor$ elements of the resulting ordered list. Consequently, setting $\alpha = 1$ results in a completely random traversal of the adjacent vertices. If $\alpha = 0.5$, the algorithm randomly shuffles

the first half of the sorted neighbors. To prevent greedy ordering when $\alpha = 0$, the algorithm assigns a random value in the range $(0, 1)$ to α at each iteration.

Following this partial sorting process, line 9 evaluates whether each adjacent vertex w remains unvisited. For every unvisited vertex w, line 10 enqueues it into Q and marks it as visited. Then, line 11 assigns the current label i to w and increments i by one. Finally, line 15 returns the labeling L.

6 Description of the Tests

This study conducted the experiments on a machine equipped with an Intel® Xeon® Gold 6330 processor and 32 GiB of RAM (DIMM DDR4 3200 MT/s). The operating system was Ubuntu 24.04 LTS (Linux kernel 6.8.0-45-generic).

The BRKGA, BRKGA+, BRKGA*, RCM, and DRSA heuristics employed the Compressed Sparse Row (CSR) format [49]. This study implemented all eight algorithms in C++ and compiled using g++ 13.2.0 with the -O3 optimization flag.

This study evaluates the algorithms in a benchmark set comprising 23 graphs from the literature [22]. These graphs originate from real-world applications of metaheuristic algorithms for bandwidth reduction. Moreover, these graphs are publicly available in the SuiteSparse Matrix Collection [10].

We employed Irace [38] to configure the parameters automatically of all evaluated metaheuristic algorithms. This tool efficiently explores the parameter space to identify satisfactory parameter configurations. Irace executes iterated runs, discarding inferior configurations based on statistical tests. To avoid premature convergence, the tool incorporates a restart mechanism.

We kept most Irace settings at their default values, modifying only the statistical elimination test (testType) and the experiment budget (maxExperiments). We selected the Friedman test and fixed the maximum number of experiments at 200. This value represents the minimum budget recommended by Irace to parameterize the DRSA heuristic. We executed Irace independently for each of the 23 benchmark graphs.

Defining an acceptable timeout for metaheuristics is critical. Extended execution times are often impractical. Following an earlier work [24], we established a maximum execution time of 1,200 s as the only stopping criterion for all algorithms evaluated in this study.

We evaluated the best bandwidth and the average of four solutions per graph. This count corresponds to the minimum number of executions performed by Irace under the used configuration. The subsequent sections detail the parameter tuning for each metaheuristic. Section 6.1 addresses the BRKGA heuristic [47], while Sect. 6.2 introduces the proposed BRKGA* and BRKGA+ heuristics. Section 6.3 presents the parameters configured by Irace for DRSA [50], and Sect. 6.4 details the parameterization of NCHC+ (firstly used in the present study) and FNCHC+ [24].

6.1 BRKGA [47]

The parameters configured by Irace for the BRKGA heuristic [47] were $|P| \in [50, 1000]$ (integer), $e \in [10, 400]$ (integer), $r \in [5, 200]$ (integer), and $p \in [0.55, 0.99]$ (real). Furthermore, we configured Irace to prevent parameters that violate the following rules: $e > 0.4 \cdot |P|$ and $r > 0.2 \cdot |P|$.

We defined the range of the parameter $|P|$ based on the population size reported by Silva et al. [47], and after preliminary tests. Silva et al. [47] reported the parameters $|P| = 1000$, $e = 500$, and $r = 500$. Thus, the population incorporates only individuals belonging to the elite set and newly generated individuals in each new generation in the BRKGA heuristic [47]. Therefore, we defined bounds for the parameters e and r to ensure the existence of descendants produced by crossover operations in which at least one parent belongs to neither the elite set nor the newly generated individuals from the constructive method. Furthermore, we defined the ranges for the parameters e, r, and p based on values recommended by Gonçalves and Resende [20], as well as on preliminary tests. Table 1 shows the best parameter configurations found by Irace for the BRKGA heuristic [47]. We have abbreviated the names of graphs Fashion_MNIST_norm_10NN, kmnist_norm_10NN, MISKnowledgeMap, k49_norm_10NN, preferentialAttachment, vsp_barth5_1Ksep_50in_5Kout, vsp_befref_fxm_2_4_air02, vsp_bump2_e18_aa01_model1_crew1, vsp_c-30_data_data, vsp_data_and_seymourl, vsp_finan512_scagr7-2c_rlfddd, vsp_model1_crew1_cr42_south31, vsp_msc10848_300sep_100in_1Kout, vsp_p0291_seymourl_iiasa, vsp_sctap1-2b_and_seymourl, vsp_south31_slptsk, and vsp_vibrobox_scagr7-2c_rlfddd in the subsequent tables.

6.2 BRKGA+ and BRKGA*

The parameters configured by Irace for the BRKGA+ and BRKGA* heuristics were $\alpha \in [0.0, 1.0]$ (real), $init \in \{0, 1\}$ (categorical), $|P| \in [10, 100]$ (integer), $e \in [2, 40]$ (integer), $r \in [1, 10]$ (integer), and $p \in [0.55, 0.99]$ (real). Preliminary tests showed that populations of size $|P| > 100$ yielded unpromising results due to the computational cost associated with generating and maintaining a larger population.

We defined the ranges for the parameters e, r, and p based on values recommended by Gonçalves and Resende [20], as well as on preliminary tests. Table 1 shows the best parameter settings found by Irace for the BRKGA+ and BRKGA* heuristics.

6.3 DRSA [50]

We implemented DRSA in C++ strictly following the original description by Torres-Jimenez et al. [50]. Because this algorithm is a direct application of the Simulated Annealing metaheuristic, its implementation is very straightforward. We configured the Irace to tune the DRSA parameters within the following ranges: $T_0 \in [500, 10000]$ (integer), $T_f \in [1 \times 10^{-8}, 0.1]$ (real), $\alpha \in [0.9, 0.99]$

Table 1. The best parameter settings found by Irace for the BRKGA [47], BRKGA+, and BRKGA* heuristics in 23 graphs.

Graph	BRKGA [47]				BRKGA+						BRKGA*					
	$\|P\|$	e	r	p	$\|P\|$	e	r	α	$init$	p	$\|P\|$	e	r	α	$init$	p
boyd1	793	271	54	0.60	70	20	5	0.94	1	0.77	75	17	6	0.96	1	0.70
Fashion_MNIST	930	45	152	0.75	66	3	8	0.90	0	0.57	66	5	3	0.85	0	0.70
har_10NN	647	71	107	0.82	64	17	4	0.81	1	0.75	52	15	10	0.91	1	0.98
k49_norm	927	18	140	0.66	86	28	7	0.61	1	0.87	67	20	9	0.48	1	0.74
kmnist_norm	887	97	80	0.80	77	6	9	0.85	0	0.59	94	13	8	0.82	0	0.91
lp1	706	71	43	0.96	33	7	7	0.28	1	0.63	97	38	9	0.85	0	0.96
MISKnowledge	959	32	91	0.55	71	23	6	0.93	0	0.73	71	25	4	0.31	0	0.64
power	849	91	146	0.94	62	10	6	0.85	0	0.75	56	22	5	0.87	0	0.70
preferentialAtt	476	46	69	0.57	74	8	10	0.99	1	0.79	20	3	4	0.69	0	0.70
smallworld	713	72	109	0.86	57	19	9	0.66	0	0.89	76	21	8	0.59	1	0.78
vsp_barth5	761	50	99	0.84	76	24	7	0.36	0	0.69	44	5	4	0.60	0	0.74
vsp_befref	904	141	137	0.95	45	15	5	0.65	0	0.74	60	22	7	0.70	0	0.64
vsp_bump2	661	33	79	0.80	59	18	10	0.34	0	0.92	87	14	10	0.16	0	0.70
vsp_c-30	973	74	167	0.73	100	30	8	0.87	0	0.58	79	19	3	0.89	0	0.70
vsp_data	743	166	139	0.96	76	7	8	0.53	0	0.66	83	18	8	0.58	0	0.97
vsp_finan512	662	262	34	0.72	32	5	7	0.55	0	0.69	69	19	10	0.45	0	0.70
vsp_model1	808	21	93	0.63	75	15	6	0.26	0	0.62	46	2	7	0.37	0	0.64
vsp_msc10848	438	27	67	0.92	60	12	7	0.31	0	0.86	68	24	7	0.09	0	0.57
vsp_p0291	888	199	96	0.87	87	6	5	0.27	0	0.97	87	9	7	0.29	0	0.98
vsp_sctap1-2b	831	26	112	0.80	42	2	7	0.75	0	0.84	74	20	8	0.31	0	0.79
vsp_south31	923	133	166	0.80	63	18	6	0.34	0	0.71	45	3	5	0.37	0	0.75
vsp_vibrobox	635	39	35	0.84	92	14	10	0.01	0	0.99	43	9	6	0.14	0	0.66
worms20_10NN	982	313	104	0.57	61	8	10	0.87	0	0.61	87	14	7	0.47	0	0.91

(real), $L \in [30, 50]$ (integer), $L_f \in [1, 20]$ (integer), $P_{rex} \in [0.0, 0.7]$ (real), and $P_{nex} \in [P_{rex}, 0.9]$ (real). While the original authors [50] fixed $L_f = 10 \cdot |V| \cdot |E|$, our study scales the L_f parameter by multiplying it by $|V| \cdot |E|$ before executing the heuristic. Consequently, providing the interval $[1, 20]$ allows Irace to explore L_f values both substantially lower and higher than the original baseline. We applied this same logic to establish the intervals for all other DRSA parameters. Furthermore, Sect. 6.4 details how we adopted an identical approach to configure the parameters for the NCHC+ and FNCHC+ algorithms. Finally, Table 2 presents the best parameter configuration found by Irace for the DRSA heuristic [50].

Table 2. The best parameter settings found by Irace for the DRSA heuristic [50] in 23 graphs.

Graph	α	L	L_f	P_{nex}	P_{rex}	T_0	T_f
boyd1	0.95	44	4	0.52	0.23	1680	3×10^{-2}
Fashion_MNIST	0.91	34	11	0.11	0.69	1412	4×10^{-2}
har_10NN	0.95	33	20	0.17	0.26	6692	1×10^{-8}
k49_norm	0.98	33	7	0.35	0.51	3424	1×10^{-8}
kmnist_norm	0.95	38	16	0.29	0.60	1083	2×10^{-2}
lp1	0.92	30	2	0.13	0.63	3819	5×10^{-2}
MISKnowledge	0.92	30	11	0.19	0.40	4018	2×10^{-2}
power	0.94	34	20	0.49	0.16	1409	1×10^{-8}
preferentialAtt	0.93	34	15	0.60	0.25	3130	1×10^{-2}
smallworld	0.98	44	2	0.01	0.63	3695	6×10^{-2}
vsp_barth5	0.90	47	1	0.04	0.46	825	2×10^{-2}
vsp_befref	0.95	36	10	0.12	0.57	6412	1×10^{-2}
vsp_bump2	0.96	31	5	0.65	0.14	670	1×10^{-8}
vsp_c-30	0.96	44	13	0.40	0.45	3175	1×10^{-8}
vsp_data	0.94	35	14	0.29	0.28	7125	1×10^{-8}
vsp_finan512	0.93	30	12	0.16	0.58	6636	1×10^{-8}
vsp_model1	0.94	46	16	0.15	0.53	7660	1×10^{-2}
vsp_msc10848	0.98	32	5	0.34	0.41	2683	1×10^{-8}
vsp_p0291	0.93	32	8	0.52	0.03	6133	1×10^{-8}
vsp_sctap1-2b	0.93	30	8	0.17	0.36	5594	5×10^{-2}
vsp_south31	0.92	32	7	0.16	0.54	4015	1×10^{-8}
vsp_vibrobox	0.95	39	3	0.08	0.56	2386	1×10^{-8}
worms20_10NN	0.92	50	9	0.11	0.58	612	7×10^{-2}

6.4 NCHC+ and FNCHC+ [21,24]

The NCHC+ algorithm is a variation of the NCHC heuristic [36]. NCHC used a single value for λ. Unlike NCHC, NCHC+ varies the parameter λ, such as in the FNCHC+ heuristic [21,24]. NCHC+ also uses the condition with κ [21,24] to restrict calls to the Hill Climbing (HC) local search, which has a high execution cost. Thus, NCHC+ is feasible for large-scale graphs in a reasonable runtime [21,24]. The present study applies the NCHC+ heuristic for the first time.

Irace configured the following parameters of the NCHC+ and FNCHC+ heuristics: $\kappa \in [1, 200]$ (integer) and $NC_{max} \in [1, 200]$ (integer). Additionally, Irace configured $\lambda \in [0.85, 1.0]$ (real) for the NCHC+ heuristic. We defined the ranges used for the parameters κ and NC_{max} (NCHC+ and FNCHC+) based on the parameters reported by Gonzaga de Oliveira and Carvalho [24], and on

preliminary tests. We defined the range used for the parameter λ based on the results reported by Lim et al. [36] and after preliminary tests.

Table 3 shows the best parameter settings found by Irace for the NCHC+ and FNCHC+ heuristics. The present study employed the FNCHC+ implementation of Gonzaga de Oliveira and Carvalho [24]. This study modified this implementation to use the NCHC+ heuristic. Table 3 shows that the 23 graphs used in this study have dimensions ranging from 2,427 to 534,388 rows and columns and 13,188 to 2,442,056 non-zero elements.

Table 3. The best parameter settings found by Irace for the NCHC+ heuristic and FNCHC+ [24] in 23 graphs.

Graph	$\|V\|$	$\|E\|$	NCHC+			FNCHC+	
			λ	NC_{max}	κ	κ	NC_{max}
boyd1	93,279	1,211,231	0.88	137	110	66	80
Fashion_MNIST	10,000	158,304	0.88	15	124	131	93
har_10NN	10,299	151,736	0.99	63	176	130	51
k49_norm	38,547	618,158	1.00	30	42	151	153
kmnist_norm	10,000	156,932	0.99	19	181	123	184
lp1	534,388	11,643,420	0.89	138	111	92	17
MISKnowledge	2,427	57,022	0.89	78	113	73	185
power	4,941	13,188	0.89	185	178	75	198
preferentialAtt	100,000	999,970	1.00	92	149	90	196
smallworld	100,000	999,996	1.00	161	110	113	194
vsp_barth5	32,212	203,610	1.00	43	121	108	85
vsp_befref	14,109	196,448	1.00	91	165	156	188
vsp_bump2	56,438	601,602	1.00	31	190	192	66
vsp_c-30	11,023	124,368	1.00	177	90	181	193
vsp_data	9,167	111,732	1.00	60	96	178	137
vsp_finan512	139,752	1,104,040	1.00	62	108	50	164
vsp_model1	45,101	379,952	1.00	131	198	88	200
vsp_msc10848	21,996	2,442,056	1.00	41	172	146	21
vsp_p0291	10,498	107,736	0.99	178	99	41	99
vsp_sctap1-2b	40,174	281,662	1.00	64	59	130	114
vsp_south31	39,668	379,828	1.00	175	196	188	110
vsp_vibrobox	77,328	871,172	0.97	11	38	169	191
worms20_10NN	20,055	240,826	0.89	4	171	120	181

7 Results and Analysis

This section evaluates the bandwidth results of the BRKGA+, BRKGA*, and NCHC+ heuristics against the GPS [17], RCM [15], DRSA [50], BRKGA [47], and FNCHC+ [21,24] heuristics. Table 4 shows the bandwidth results of the eight evaluated heuristics.

Table 4 displays the number of best results and the Sum of Relative Deviations (SRD) over all graphs. The Relative Deviation (RD) for a given graph, calculated as $\frac{\beta(G_L)-\beta(G_{L^*})}{\beta(G_{L^*})}$, represents the extent to which the bandwidth of a solution L deviates from the bandwidth of the best solution L^* found. The GPS algorithm failed to solve within the 1,200 s time limit when evaluated on the k49_norm_10NN, lp1, and vsp_vibrobox graphs. Consequently, we used the initial bandwidth value of these graphs to calculate the RD.

Table 4. The bandwidth results from 8 heuristics applied to 23 graphs [22].

Graph	FNCHC+		NCHC+		BRKGA*		BRKGA+		RCM	GPS	DRSA		BRKGA	
	best	avg.	best	avg	best	avg.	best	avg.			best	avg.	best	avg.
boyd1	**46634**	**46634**	46636	46636	48227	77045	87936	88855	89508	93276	46901	46919	48072	48443
Fashion_MNIST	**1422**	**1422**	1541	1541	1578	1602	1571	1591	1890	5138	9698	9717	9787	9794
har_10NN	**942**	**942**	1101	1101	1122	1126	1123	1128	1804	3434	8811	9014	10076	10105
k49_norm	**11668**	**11668**	11913	11913	12827	12945	13040	13080	16783	†	37935	37949	38170	38196
kmnist_norm	**2271**	**2271**	2323	2323	2988	2992	2988	2993	3660	7561	9825	9841	9799	9805
lp1	**284546**	**284546**	284641	284641	284550	284550	284645	284645	284744	†	532668	532766	530265	530499
MISKnowledge	**684**	**684**	934	934	1031	1031	1016	1018	1346	1994	2339	2351	2333	2336
power	230	230	**196**	**196**	272	272	272	272	294	1088	513	563	4490	4531
preferentialAtt	**50849**	**50849**	61877	61877	73427	73439	73339	73448	73802	99962	99165	99414	99378	99427
smallworld	**44502**	**44502**	45521	45521	53691	53702	53691	53719	54556	96435	99623	99642	99281	99303
vsp_barth5	**8984**	**8984**	**8984**	**8984**	11778	11783	11779	11789	12365	30232	31582	31621	31708	31732
vsp_befref	**3431**	**3431**	3546	3546	5647	5655	5647	5647	9924	13736	13730	13755	13613	13690
vsp_bump2	**17316**	**17316**	20453	20453	23916	23966	23898	23942	35336	53706	55516	55569	55839	55878
vsp_c-30	**2284**	**2284**	2300	2300	3676	3679	3687	3693	4207	9770	8365	8465	10676	10700
vsp_data	**3149**	**3149**	**3149**	**3149**	4449	4455	4459	4480	5367	9022	6390	6525	8892	8917
vsp_finan512	**39676**	**39676**	40239	40239	43712	44292	44302	44385	68124	128618	139027	139129	138159	138159
vsp_model1	**14315**	**14315**	14509	14509	17631	17631	17631	17631	21267	39133	44469	44540	44266	44316
vsp_msc10848	**6706**	**6706**	8173	8173	8221	8275	8221	8277	14817	21396	21813	21843	21887	21893
vsp_p0291	3987	3987	**3796**	**3796**	4420	4421	4299	4392	6620	9741	8312	8432	10166	10185
vsp_sctap1-2b	**12935**	**12935**	**12935**	**12935**	17426	17734	17418	17461	18991	38849	39663	39706	39588	39616
vsp_south31	**15749**	**15749**	**15749**	**15749**	17631	17632	17631	17632	20321	38106	38082	38306	38946	38972
vsp_vibrobox	**31685**	**31685**	45908	45908	43725	44096	43727	44146	49677	†	76530	76621	76686	76723
worms20_10NN	**699**	**699**	748	748	744	744	741	743	1024	2939	19751	19769	18501	18501
SRD	**0.2**	**0.2**	1.9	1.9	6.3	7.0	7.1	7.2	15.2	48.8	78.5	79.2	100.7	101.0
No. of bests results	**21**	**21**	6	6	0	0	0	0	0	0	0	0	0	0

Table 4 shows that FNCHC+ yielded 21 of the best overall bandwidth results. The new heuristic NCHC+ returned six of the best results. FNCHC+ (NCHC+) achieved a Sum of Relative Deviations (SRD) of 0.2 (1.9) for the best and average results across all 23 graphs. In contrast, the other six heuristics failed to yield competitive solutions.

BRKGA* and BRKGA+ outperformed GPS [17], RCM [15], DRSA [50], and BRKGA [47]. BRKGA* produced slightly better results than BRKGA+ because the former employs the number of critical vertices as a tiebreaker criterion (see Sect. 4). The SRD values achieved by BRKGA* (BRKGA+) for the best and average solutions were 6.3 (7.1) and 7.0 (7.2), respectively. The RCM, GPS, DRSA, and BRKGA heuristics yielded considerably worse SRD values.

Table 4 shows that BRKGA [47] achieved a considerably worse SRD than DRSA [50]. Additionally, DRSA yielded 14 better results than BRKGA, while BRKGA returned 9 better results than DRSA. Conversely, FNCHC+ [21,24], NCHC+, BRKGA*, BRKGA+, and RCM [15] outperformed DRSA.

8 Summary and Conclusions

This work introduced novel heuristics for the Bandwidth Reduction Problem: two based on the Biased Random-Key Genetic Algorithm, BRKGA* and BRKGA+, and an adaptation of the NCHC algorithm [36], NCHC+. We assessed the performance of these heuristics against classical and state-of-the-art algorithms on a benchmark of real-world applications composed of large-scale graphs from the literature [22].

Experimental results demonstrated that BRKGA* and BRKGA+ effectively outperformed DRSA [50], the former state-of-the-art metaheuristic for small-sized graphs. However, FNCHC+ [21,24] and NCHC+ outperformed BRKGA* and BRKGA+. Therefore, this study corroborates FNCHC+ as the current state-of-the-art metaheuristic algorithm for the bandwidth reduction of real-world graphs. In future research, we intend to explore the application of new heuristics based on other metaheuristics to this optimization problem.

Acknowledgments. The first author acknowledges IFSC (Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina). The second author acknowledges the Brazilian funding agency CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico) for financial support with grant 402599/2023-3. This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001. The authors also thank the anonymous referees for their careful reading, insightful comments, and suggestions.

Disclosure of Interests. The authors declare that they have no competing interests.

References

1. Abreu, A.A.A.M., Gonzaga de Oliveira, S.L.: Iterated local search with Tabu search for the bandwidth reduction problem in graphs. In: et al., O.G. (ed.) ICCSA 2024. LNCS, vol. 14813, pp. 125–136. Springer, Cham (2024). https://doi.org/10.1007/978-3-031-64605-8_9
2. Amparore, E.G., Beccuti, M., Donatelli, S.: Gradient-based variable ordering of decision diagrams for systems with structural units. In: D'Souza, D., Kumar, K.N. (eds.) ATVA 2017. LNCS, vol. 10482, pp. 184–200. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-68167-2_13

3. Bean, J.C.: Genetic algorithms and random keys for sequencing and optimization. ORSA J. Comput. **6**(2), 154–160 (1994). https://doi.org/10.1287/ijoc.6.2.154
4. Berry, M.W., Hendrickson, B., Raghavan, P.: Sparse matrix reordering schemes for browsing hypertext. In: Renegar, J., Shub, M., Smale, S. (eds.) Lectures in Applied Mathematics, vol. 32, pp. 99–123. American Mathematical Society Press, Park City, Utah, USA (1996), the Mathematics of Numerical Analysis (1996)
5. Bolanos, M.E., Aviyente, S., Radha, H.: Graph entropy rate minimization and the compressibility of undirected binary graphs. In: Proceedings of the IEEE Statistical Signal Processing Workshop (SSP), Ann Arbor, MI, pp. 109–112. IEEE (2012). https://doi.org/10.1109/SSP.2012.6319634
6. Campos, V., Piñana, E., Martí, R.: Adaptive memory programming for matrix bandwidth minimization. Ann. Oper. Res. **183**, 7–23 (2011)
7. Cavero, S., Pardo, E.G., Duarte, A., Martí, R.: Graph Layout Problems. Springer, Cham (2025). https://doi.org/10.1007/978-3-319-07153-4_45-2
8. Chen, H.K.: Evaluation of triangular mesh layout techniques using large mesh simplification. Multimedia Tools Appl. **76**(23), 25391–25419 (2017). https://doi.org/10.1007/s11042-017-4607-z
9. Concas, A., Fenu, C., Rodriguez, G.: PQser: a Matlab package for spectral seriation. Numerical Algorithms **80**(3), 879–902 (2019). https://doi.org/10.1007/s11075-018-0510-6
10. Davis, T.A., Hu, Y.: The University of Florida sparse matrix collection. ACM Trans. Math. Software **38**(1), 1–25 (2011). https://doi.org/10.1145/2049662.2049663
11. Dueck, G.W., Jeffs, J.: A heuristic bandwidth reduction algorithm. J. Comb. Math. Comb. Comput. 97–108 (1995)
12. Esposito, A., Tarricone, L.: Matrix bandwidth reduction with Tabu search: parallel implementation on Cray T3D and applications to microwave circuit design. In: Voli, E. (ed.) Science and Supercomputing at CINECA: Report, Bologna, Italy, pp. 440–450. CINECA (1996)
13. Garey, M.R., Graham, R.L., Johnson, D.S., Knuth, D.: Complexity results for bandwidth minimization. SIAM J. Appl. Math. **34**, 477–495 (1978)
14. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. Freeman, New York (1979)
15. George, A., Liu, J.W.H.: Computer Solution of Large Sparse Positive Definite Systems. Prentice-Hall, Englewood Cliffs (1981)
16. George, A., Liu, J.W.H.: An implementation of a pseudoperipheral node finder. ACM Trans. Math. Software **5**(3), 284–295 (1979). https://doi.org/10.1145/355841.355845
17. Gibbs, N.E., Poole, W.G., Stockmeyer, P.K.: An algorithm for reducing the bandwidth and profile of a sparse matrix. SIAM J. Numer. Anal. **13**(2), 236–250 (1976). https://doi.org/10.1137/0713023
18. Gibbs, N.E., Poole, W.G., Stockmeyer, P.K.: Comparison of several bandwidth and profile reduction algorithms. ACM Trans. Math. Software **4**(1), 322–330 (1976)
19. Gonçalves, J.F., Resende, M.G.C.: Biased random-key genetic algorithms for combinatorial optimization. J. Heuristics **17**(5), 487–525 (2011). https://doi.org/10.1007/s10732-010-9143-1
20. Gonçalves, J.F., Resende, M.G.C.: Random-key genetic algorithms. In: Martí, R., Pardalos, P.M., Resende, M.G.C. (eds.) Handbook of Heuristics, chap. 13, pp. 703–715. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-07124-4_30

21. Gonzaga de Oliveira, S.L.: An evaluation of heuristic methods for the bandwidth reduction of large-scale graphs. Revista Pesquisa Operacional **43**(e268255), 1–22 (2023). https://doi.org/10.1590/0101-7438.2023.043.00268255
22. Gonzaga de Oliveira, S.L.: A survey of heuristics for matrix bandwidth reduction. Comput. Sci. Rev. **56**, 100724 (2025). https://doi.org/10.1016/j.cosrev.2025.100724
23. Gonzaga de Oliveira, S.L., de Abreu, A.A.A.M., Robaina, D., Kischinhevsky, M.: A new heuristic for bandwidth and profile reductions of matrices using a self-organizing map. In: Gervasi, O., et al. (eds.) ICCSA 2016. LNCS, vol. 9786, pp. 54–70. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-42085-1_5
24. Gonzaga de Oliveira, S.L., Carvalho, C.: Metaheuristic algorithms for the bandwidth reduction of large-scale matrices. J. Comb. Optim. **43**, 727–784 (2022). https://doi.org/10.1007/s10878-021-00801-6
25. Gonzaga de Oliveira, S.L., Silva, L.M.: An ant colony hyperheuristic approach for matrix bandwidth reduction. Appl. Soft Comput. **94**, 106434 (2020). https://doi.org/10.1016/j.asoc.2020.106434
26. Gonzaga de Oliveira, S.L., Silva, L.M.: Evolving reordering algorithms using an ant colony hyperheuristic approach for accelerating the convergence of the ICCG method. Eng. Comput. **36**(4), 1857–1873 (2019). https://doi.org/10.1007/s00366-019-00801-5
27. Gonzaga de Oliveira, S.L., Bernardes, J.A.B., Chagas, G.O.: An evaluation of reordering algorithms to reduce the computational cost of the incomplete Cholesky-conjugate gradient method. Comput. Appl. Math. **37**(3), 2965–3004 (2017). https://doi.org/10.1007/s40314-017-0490-5
28. Gonzaga de Oliveira, S.L., Bernardes, J.A.B., Chagas, G.O.: An evaluation of low-cost heuristics for matrix bandwidth and profile reductions. Comput. Appl. Math. **37**(2), 1412–1471 (2016). https://doi.org/10.1007/s40314-016-0394-9
29. Harper, L.H.: Optimal assignments of numbers to vertices. J. Soc. Ind. Appl. Math. **12**, 131–135 (1964)
30. Harper, L.H.: Optimal numberings and isoperimetric problems on graphs. J. Comb. Theory **1**(3), 385–393 (1966)
31. Higham, D.J.: Unravelling small world networks. J. Comput. Appl. Math. **158**(1), 61–74 (2003). https://doi.org/10.1016/S0377-0427(03)00471-0
32. Johnson, D.S.: A theoretician's guide to the experimental analysis of algorithms. In: Data Structures, Near Neighbor Searches, and Methodology: Fifth and Sixth DIMACS Implementation Challenges, vol. 59, Providence, RI, pp. 215–250. American Mathematical Society (AMS) (2002)
33. Koohestani, B.: On the solution of the graph bandwidth problem by means of search methods. Appl. Intell. **53**, 7988–8004 (2023). https://doi.org/10.1007/s10489-022-03802-y
34. Koohestani, B., Poli, R.: A hyper-heuristic approach to evolving algorithms for bandwidth reduction based on genetic programming. In: Research and Development in Intelligent Systems XXVIII, pp. 93–106. Springer, London (2011). https://doi.org/10.1007/978-1-4471-2318-7_7
35. Lim, A., Lin, J., Xiao, F.: Particle Swarm optimization and Hill Climbing for the bandwidth minimization problem. Appl. Intell. **3**(26), 175–182 (2007)
36. Lim, A., Rodrigues, B., Xiao, F.: A fast algorithm for bandwidth minimization. Int. J. Artif. Intell. Tools **16**(3), 537–544 (2007). https://doi.org/10.1142/s0218213007003394
37. Lim, A., Rodrigues, B., Xiao, F.: Heuristics for matrix bandwidth reduction. Eur. J. Oper. Res. 69–91 (2006). https://doi.org/10.1016/j.ejor.2005.02.066

38. López-Ibáñez, M., Dubois-Lacoste, J., Pérez Cáceres, L., Birattari, M., Stützle, T.: The Irace package: Iterated racing for automatic algorithm configuration. Oper. Res. Perspect. **3**, 43–58 (2016). https://doi.org/10.1016/j.orp.2016.09.002
39. Marichal, R., Dufrechou, E., Ezzatti, P.: Optimizing sparse matrix storage for the big data era. In: Naiouf, M., Rucci, E., Chichizola, F., De Giusti, L. (eds.) Communications in Computer and Information Science (JCC-BD&ET 2021: Cloud Computing, Big Data & Emerging Topics), vol. 1444, pp. 121–135. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-84825-5_9
40. Martí, R., Laguna, M., Glover, F., Campos, V.: Reducing the bandwidth of a sparse matrix with Tabu search. Eur. J. Oper. Res. **135**(2), 450–459 (2001). https://doi.org/10.1016/S0377-2217(00)00325-8
41. Mladenovic, N., Urosevic, D., Pérez-Brito, D., García-González, C.G.: Variable neighbourhood search for bandwidth reduction. Eur. J. Oper. Res. **200**(1), 14–27 (2010). https://doi.org/10.1016/j.ejor.2008.12.015
42. Mueller, C., Martin, B., Lumsdaine, A.: A comparison of vertex ordering algorithms for large graph visualization. In: Proceedings of the 6th International Asia-Pacific Symposium on Visualization (APVIS'07), Sydney, Australia, pp. 141–148 (2007). https://doi.org/10.1109/APVIS.2007.329289
43. Papadimitriou, C.: The NP-completeness of bandwidth minimization problem. Comput. J. **16**, 177–192 (1976). https://doi.org/10.1007/BF02280884
44. Petit, J.: Addenda to the survey of layout problems. Bull. EATCS **105**, 177–201 (2011)
45. Piñana, E., Plana, I., Campos, V., Martí, R.: GRASP and path relinking for the matrix bandwidth minimization. Eur. J. Oper. Res. **153**(1), 200–210 (2004). https://doi.org/10.1016/S0377-2217(02)00715-4
46. Rodriguez-Tello, E., Kao, H.J., Torres-Jimenez, J.: An improved simulated annealing algorithm for bandwidth minimization. Eur. J. Oper. Res. **185**, 1319–1335 (2008)
47. Silva, P.H.G., Brandão, D.N., Morais, I.S., Gonzaga de Oliveira, S.L.: A biased random-key genetic algorithm for bandwidth reduction. In: Gervasi, O., et al. (eds.) ICCSA 2020, Part VI. LNCS, vol. 12249, pp. 312–321. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58799-4_23
48. Spears, W., De Jong, K.: On the virtues of parametrized uniform crossover. In: Proceedings of the 4th International Conference on Genetic Algorithms, San Diego, CA, USA, July 1991. pp. 230–236 (1991). https://doi.org/10.21236/ADA293985
49. Tinney, W., Walker, J.: Direct solutions of sparse network equations by optimally ordered triangular factorization. Proc. IEEE **55**(11), 1801–1809 (1967). https://doi.org/10.1109/PROC.1967.6011
50. Torres-Jimenez, J., Izquierdo-Marquez, I., Garcia-Robledo, A., Gonzalez-Gomez, A., Bernal, J., Kacker, R.N.: A dual representation simulated annealing algorithm for the bandwidth minimization problem on graphs. Inf. Sci. **303**, 33–49 (2015). https://doi.org/10.1016/j.ins.2014.12.041