Alternative evaluation functions for the cyclic bandwidth sum problem

Eduardo Rodriguez-Tello, Frédéric Lardeux, Abraham Duarte, Valentina Narvaez-Teran

European Journal of Operational Research 273 (2019) 904–919

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Discrete Optimization

Alternative evaluation functions for the cyclic bandwidth sum problem

Eduardo Rodriguez-Tello[a,*], Frédéric Lardeux[b], Abraham Duarte[c], Valentina Narvaez-Teran[a]

[a] CINVESTAV-Tamaulipas Km. 5.5 Carretera Victoria-Soto La Marina, Victoria Tamps. 87130, Mexico
[b] LERIA, Université d'Angers., 2 Boulevard Lavoisier, Angers 49045, France
[c] Departamento de Ciencias de la Computación, Universidad Rey Juan Carlos, Calle Tulipán s/n. 28933 Móstoles, Madrid, Spain

ARTICLE INFO

Article history:
Received 14 August 2017
Accepted 22 September 2018
Available online 9 October 2018

Keywords:
Combinatorial optimization
Fitness landscape neutrality
Enhanced evaluation function
Refined discrimination capability
Search guiding efficiency

ABSTRACT

One essential element for the successful application of metaheuristics is the evaluation function. It should be able to make fine distinctions among the potential solutions in order to avoid producing wide plateaus (valleys) in the fitness landscape, on which detecting a promising search direction could be hard for certain local search strategies. In the specific case of the *cyclic bandwidth sum* (*CBS*) problem, the heuristics reported have used directly the objective function of the optimization problem to assess the quality of potential solutions. Nevertheless, such a conventional function does not allow to efficiently establish preferences among distinct potential solutions. In order to cope with this important issue, three new more refined evaluation functions for the *CBS* problem are introduced in this paper.

An in-depth comparative analysis considering the conventional and the three proposed evaluation functions is carried out and presented. It includes an assessment of their: (a) discrimination potential, (b) consistency with regard to the primary objective of the *CBS* problem, and (c) practical usefulness within two different algorithms, best improvement local search and iterated local search. A validation of the experimental results by means of a meticulous statistical significance analysis revealed that proposing more informative evaluation schemes for the *CBS* problem could be a useful means of improving the performance of metaheuristics. Indeed, our iterated local search implementation, using an alternative evaluation function, surpassed the best solutions yielded by the state-of-the-art algorithms and allow us to attain new better upper bounds for 14 out of 20 well-known benchmark instances.

1. Introduction

The *cyclic bandwidth sum* (*CBS*) is a well-studied combinatorial optimization problem. It was first studied by Yuan (1995) who demonstrated that it is a $\mathcal{NP}$-hard problem. This problem arises in some important application areas like VLSI designs (Bhatt & Leighton, 1984; Ullman, 1984), code design (Harper, 1964), simulation of network topologies for parallel computer systems (Monien & Sudborough, 1990) and scheduling in broadcasting based networks (Liberatore, 2002).

The *CBS* problem can be formally defined as follows. Let $G = (V, E)$ be a finite undirected graph (guest) of order n and C_n a cycle graph (host) with vertex set $|V_H| = n$ and edge set E_H. Given an injection $\varphi: V \rightarrow V_H$, representing an embedding of G in C_n, the cyclic bandwidth sum (the cost) for G with respect to φ is defined as:

$$\mathrm{Cbs}(G, \varphi) = \sum_{(u,v)\in E} |\varphi(u) - \varphi(v)|_n, \tag{1}$$

where $|x|_n = \min\{|x|, n - |x|\}$ (with $1 \leq |x| \leq n-1$) is called the *cyclic distance*, and the label associated to vertex u is denoted $\varphi(u)$.

Then, the *CBS* problem consists in finding the optimal embedding φ^*, such that $\mathrm{Cbs}(G, \varphi^*)$ is minimum, i.e., $\varphi^* = \arg\min_{\varphi\in\Phi}\{\mathrm{Cbs}(G, \varphi)\}$, with Φ denoting the set of all possible embeddings. It is worth noting that an embedding can also be seen as a labeling of the guest graph G using distinct vertices of the host graph C_n, thus hereafter the terms *embedding* and *labeling* are used indistinctly.

The past research on the *CBS* problem has been mainly focused on the theoretical study of its properties, with the aim of finding optimal solution values for some particular graph topologies: paths, cycles, wheels, kth power of cycles and complete bipartite graphs (Chen & Yan, 2007; Jianxiu, 2001). Jianxiu (2001) studied

* Corresponding author.
E-mail addresses: ertello@tamps.cinvestav.mx (E. Rodriguez-Tello), frederic.lardeux@univ-angers.fr (F. Lardeux), abraham.duarte@urjc.es (A. Duarte), vnarvaez@tamps.cinvestav.mx (V. Narvaez-Teran).

https://doi.org/10.1016/j.ejor.2018.09.031
0377-2217/

a special class of instances produced as the Cartesian product of two graphs and proved upper bounds for the *CBS* problem when those graphs are either a path, a cycle or a complete graph. Unfortunately, these theoretical results only permit to compute the optimal solution value (or an upper/lower bound) of the objective function, but they do not provide a procedure to construct the labeling responsible of that solution value. Therefore, heuristic or metaheuristic approaches emerge as the best strategy to find high quality embeddings in short computing times.

Satsangi, Srivastava, and Gursaran (2012) proposed the first method to solve the *CBS* problem. It is inspired by the *General Variable Neighborhood Search* methodology (GVNS) (Mladenovic & Hansen, 1997). Specifically, it starts by labeling each vertex by using a consecutive label from 1 to n (i.e., in lexicographic order), and improving this initial embedding with a *Reduced Variable Neighborhood Search* (RVNS) method. The GVNS further improves this solution by considering six different shake operators and two local search strategies. As it is shown in the computational experiences, carried out on graphs of order $n \leq 200$, GVNS achieves the optimal solution values for all the instances with known results and gives values less than the upper bound for Cartesian product of certain graphs like paths, cycles and complete graphs. For a thorough description of this method the reader is referred to Satsangi et al. (2012).

The second method specially designated for solving the *CBS* problem is a two-step algorithm, called MACH, which was proposed by Hamon, Borgnat, Flandrin, and Robardet (2016). The first step of this procedure consists in finding a collection of paths in the graph (i.e., some sequences of vertices consecutively connected). Each path is constructed by performing a depth-first search in which the next vertex is selected according to the Jaccard index (Jaccard, 1912). The second step consists in merging all obtained paths by following a greedy approach. In particular, a partial solution is augmented by inserting a new path at the position (just after or just before another already inserted path) that minimizes the cyclic bandwidth sum. As far as we know, this method currently provides the best results in the related literature. Therefore, we consider it as the current state-of-the-art algorithm.

These two algorithms have a point in common, both of them evaluate the quality of an embedding as the change in the objective function $\mathrm{Cbs}(G, \varphi)$, see Eq. (1). Nevertheless, it provides reduced information during the search process because the conventional evaluation function does not allow to establish preferences among different potential embeddings resulting in the same cyclic bandwidth sum.[1] The poor discrimination power of this function could result into large plateaus in the fitness landscape (Pitzer & Affenzeller, 2012; Stadler, 1992), on which detecting a promising search direction could be hard for certain local search strategies (Marmion, Dhaenens, Jourdan, Liefooghe, & Verel, 2011; Michiels, Aarts, & Korst, 2007).

In recent Operational Research literature, different approaches have been proposed to cope with this issue that could seriously compromise the search efficiency. They include the introduction of new alternative evaluation functions (Murovec, 2015; Smet, Bilgin, De Causmaecker, & Vanden Berghe, 2014), specialized diversification mechanisms to better traverse plateaus in the fitness landscape (Benlic, Epitropakis, & Burke, 2017), objective space decomposition (Derbel, Humeau, Liefooghe, & Verel, 2014), evaluation functions with aggregated penalty terms (Karapetyan, Mitrovic Minic, Malladi, & Punnen, 2015; Umetani, 2017) and multi-objectivization (Garza-Fabre, Toscano-Pulido, & Rodriguez-Tello, 2015; Lochtefeld & Ciarallo, 2015), just to mention some relevant examples.

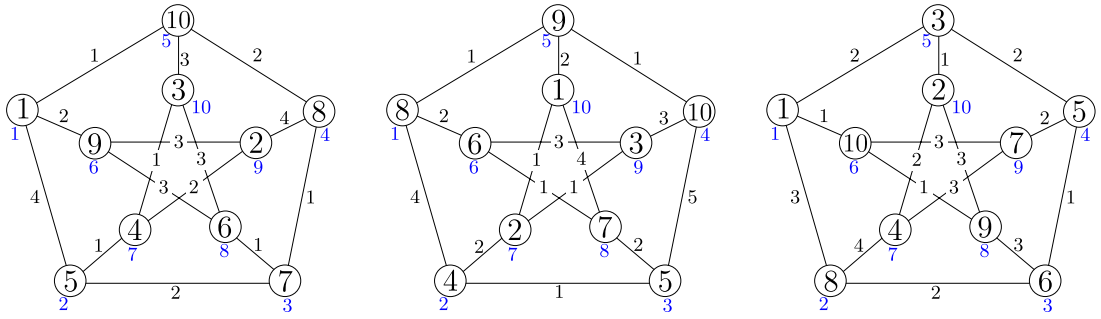
Based on our past experience designing more informative evaluation schemes (Rodriguez-Tello, Hao, & Romero-Monsivais, 2015; Rodriguez-Tello, Hao, & Torres-Jimenez, 2008a; 2008b), in this paper three new more discriminating evaluation functions for the *CBS* problem are introduced. These evaluation functions were thoroughly devised to capture even the smallest improvement that orients the searching of better solutions and permits to find embeddings in which all the cyclic distances are minimized. An in-depth comparative study considering the conventional and the three novel evaluation schemes proposed by us is carried out by following the methodology reported in Garza-Fabre, Rodriguez-Tello, and Toscano-Pulido (2013). It includes: (a) an investigation of their discrimination potential, (b) an analysis concerning the consistency of the three new evaluation functions with regard to the primary objective of the *CBS* problem, (c) an assessment of the practical usefulness of the four evaluation approaches when used within two distinct algorithms, local search with a best improvement move strategy and iterated local search, and (d) a validation of the experimental results by means of a meticulous statistical significance analysis. To further understand the extent to which the studied evaluation functions can influence the convergence process of search algorithms, this comparative study was complemented with an analysis of the evolution profiles of the average best cyclic bandwidth sum (cost) attained by the proposed iterated local search implementation. All experiments presented consider a test-suite composed of 20 standard benchmark graphs for the *CBS* problem having different topologies.

The rest of this manuscript is structured as follows: Section 2 highlights some potential drawbacks of the conventional evaluation function and presents an analysis of its main characteristics. Additionally, the three considered alternative evaluation functions for *CBS* problem are also formally described. We introduce in Section 3 the algorithmic approach to deal with the *CBS* problem, which is based on the Iterated Local Search methodology. Section 4 details the adopted benchmark instances and the performance assessment methodology. Section 5 presents computational experimentations that were carried out for examining two relevant characteristics of the studied evaluation schemes, the *capacity for discrimination* and the CBS-*compatibility*. This section continues by assessing the usefulness to guide the search process of these four evaluation schemes when they are used within two distinct algorithms, as well as their influence in the global convergence. Section 6 is devoted to compare the performance of our iterated local search implementation, equipped with the best identified evaluation function, with respect to two state-of-the-art heuristics: GVNS (Satsangi et al., 2012) and MACH (Hamon et al., 2016). Finally, Section 7 provides the overall outcomes of this work as well as some lines for future research.

2. Evaluation functions for the *CBS* problem

Metaheuristic algorithms depend on an effective evaluation approach to correctly direct the search process towards more promising zones in the solutions space (Marmion et al., 2011; Michiels et al., 2007). Nevertheless, as it was previously indicated the conventional evaluation function for the *CBS* problem, originally defined in Eq. (1), discriminates poorly among potential embeddings having the same cyclic bandwidth sum cost. For instance, consider a Petersen (guest) graph G of order $n = 10$. For this particular graph, the same cyclic bandwidth sum cost $\mathrm{Cbs}(G) = 33$ is obtained by the three different embeddings depicted in Fig. 1. Each of these embeddings is represented by the corresponding labels placed inside the vertices, which are uniquely identified with the blue numbers outside The cyclic distance of each edge $(u, v) \in E$ is

[1] Note that multiple embeddings can produce different cyclic distance combinations resulting in the same total cyclic bandwidth sum.



(a) Embedding φ. **(b)** Embedding φ'. **(c)** Embedding φ''.

Fig. 1. Three different embeddings for a Petersen (guest) graph G of order $n = 10$. All of them have the same cyclic bandwidth sum, $\text{Cbs}(G) = 33$. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

computed as it was aforementioned and represented by the number close to each edge.

More precisely, given a graph $G = (V, E)$ of order $n = |V|$ and size $m = |E|$, the conventional evaluation function $\text{Cbs}(G, \varphi)$, called hereafter Cbs for simplicity, can only take values in the range from $n - 1$ (the optimal value for a path graph) to the upper bound value for any graph given by the following expression (Jianxiu, 2001):

$$\frac{m\lfloor n/2 \rfloor \lceil n/2 \rceil}{n-1}. \tag{2}$$

These values are used for ranking a total of $|\Phi| = (n-1)!/2$ embeddings, i.e., the entire search space. Note that each possible Cbs value induces one equivalence class grouping the embeddings in Φ with the same cost. For instance, given a complete graph K_n of order $n = 200$ and size $m = 19,900$ there are only 999,802 different Cbs values ($199 \leq \text{Cbs} \leq 1,000,000$) which can be employed to rank a total of 3.9416E+372 potential embeddings. Nevertheless, some equally ranked embeddings (within the same equivalence class) could lead the search algorithm to reach better solutions than others in further iterations.

The weak capacity for discrimination furnished by the conventional evaluation function of the *CBS* problem generates the existence of wide plateaus in the search landscape. In such neutral zones, metaheuristics (principally those based on local search methods) could fail to identify a promising search direction, leading to an almost randomly guided search process.

In order to overcome the negative features of the conventional evaluation function Cbs, three alternative evaluation functions for the *CBS* problem are proposed. The aim of these alternative evaluations schemes is to provide a more fine-grained discrimination among potential solutions, which allows metaheuristic algorithms to make the most appropriate choice at each iteration of the optimization process. These functions take into consideration not only the total cyclic bandwidth sum of an embedding, but also additional semantic information related to the potential solution.

In these new evaluation functions d_k represents the number of cyclic distances with value (magnitude) k between adjacent vertices of G, i.e., $d_k = \sum_{(u,v)\in E} l_{uv}$, with l_{uv} equals 1 if $|\varphi(u) - \varphi(v)|_n = k$, and 0 otherwise. Observe that the conventional function Cbs can be expressed in terms of the number of cyclic distances d_k in the graph using the following equation:

$$\text{Cbs}(G, \varphi) = \sum_{k=1}^{\lfloor n/2 \rfloor} k \cdot d_k, \tag{3}$$

since the maximum value k that a cyclic distance can reach is $\lfloor n/2 \rfloor$.

The main idea of the following three new functions is to consider that each number of cyclic distances d_k should have a distinct level of contribution when they are used to compute the cost value of an embedding. It is accomplished by assigning a different weight value to each number of cyclic distances d_k.

$$f_1(G, \varphi) = \sum_{k=1}^{\lfloor n/2 \rfloor} \left(\sum_{i=1}^{k} i^3 \right) \cdot d_k, \tag{4}$$

$$f_2(G, \varphi) = \sum_{k=1}^{\lfloor n/2 \rfloor} n^{(k+1)} \cdot d_k, \tag{5}$$

$$f_3(G, \varphi) = \text{Cbs}(G, \varphi) + \sum_{k=1}^{\lfloor n/2 \rfloor} \left(\frac{1}{n2^k} \right) \cdot d_k. \tag{6}$$

Please observe that evaluation functions f_1 and f_2 assign small weight values to small cyclic distances. By minimizing these evaluation functions a search algorithm penalizes the cyclic distances d_k having large values of k which are closer to the cyclic bandwidth $\text{Cb}(G, \varphi)$ of the graph[2], and privileges those with small values of k. As an indirect consequence, the Cbs value of the entire graph is reduced.

On the contrary evaluation function f_3 attributes big weight values to those cyclic distances d_k having small k values. The logic behind this is that it could be easier to reduce the total cyclic bandwidth sum of an embedding if it has summands of bigger value. Besides, $\lfloor f_3 \rfloor$ equals the integer value computed with Eq. (1).

By attributing a different weight value to each cyclic distance magnitude, the three new evaluation functions have the ability to create more equivalence classes with a lower cardinality. This is an important characteristic which allows them to capture even the smallest improvement that orients the searching process toward better embeddings.

Once we have introduced the three alternative evaluations functions, we proceed to analyze their computational complexity. Recall that evaluating the quality of an embedding φ, by using the conventional evaluation function Cbs, requires analyzing all the edges in the graph $G = (V, E)$, thus $\mathcal{O}(|E|)$ instructions must be executed.

The alternative evaluation functions f_1, f_2, and f_3 can be computed more efficiently if the weight values associated to each number of cyclic distances d_k (defined in Eqs. (4), (5), and (6), respectively) are precomputed and stored in a separate array of size $\lfloor |V|/2 \rfloor$. This requires to execute $\lfloor |V|/2 \rfloor$ operations, i.e., $\mathcal{O}(|V|)$. Then, each time that the value of one of the alternative evaluation functions should be calculated, the weighted sum over all

[2] $\text{Cb}(G, \varphi) = \max_{(u,v)\in E} \{|\varphi(u) - \varphi(v)|_n\}$.

the edges $(u, v) \in E$ of the graph is computed. It implies a computational complexity similar to that required to compute Cbs, $\mathcal{O}(|V| + |E|) \approx \mathcal{O}(|E|)$.

Additionally, the alternative evaluation functions f_1, f_2, and f_3 permit an incremental evaluation of neighboring solutions using appropriate data structures.[3] Indeed, suppose that the labels of two different vertices u, v are exchanged, then we should only recompute the $|\mathcal{A}(u)| + |\mathcal{A}(v)|$ cyclic distances that change, where $|\mathcal{A}(u)|$ and $|\mathcal{A}(v)|$ represent the number of adjacent vertices to u and v, respectively. As it can be seen this is faster than the $\mathcal{O}(|E|)$ operations required originally.

3. Algorithmic approach

Local search methods might easily get trapped into local optima, which usually prevents them from finding global optimum solutions (Blum & Roli, 2003; Talbi, 2009). Therefore, there is a need to implement additional strategies for assisting the search process to find trajectories getting away from local optima. Iteratively restarting the local search from a distinct initial solution is one possible strategy, which has been implemented in the *Iterated Local Search* (ILS) algorithm (Lourenço, Martin, & Stützle, 2003; 2010; Martin, Otto, & Felten, 1991). ILS has proved to be an effective tool for approximating globally optimal solutions for distinct $\mathcal{NP}$-hard optimization problems. For this reason, in recent years, there has been a growing interest in the Operational Research community for studying this kind of metaheuristic algorithm (Avci & Topaloglu, 2017; Coelho et al., 2016; Cruz, Subramanian, Bruck, & Iori, 2017; Godim da Fonseca, Gambini Santos, Machado Toffolo, Souza Brito, & Freitas Souza, 2016; Porumbel, Goncalves, Allaoui, & Hsu, 2017; Silva, Subramanian, & Ochi, 2015; Silva Paiva & Carvalho, 2017).

To investigate into the suitability of the analyzed evaluations functions, an ILS procedure, outlined in Algorithm 1, is used. The proposed method starts with a feasible embedding generated at random, denoted as φ (line 1). This initial solution is improved with a local search method, described below, to produce a local optimum φ^*, which becomes the incumbent solution (line 2). Then, it performs the perturbation procedure, explained below (line 4), to change slightly the incumbent solution which provides the embedding φ'. This is followed by a new round of the local search procedure (line 5) to reach a new local optimum φ'' from the perturbed solution. After each local search procedure call, the new local optimum reached φ'' is accepted as the new incumbent solution if it scores a better cost value (computed with one of the proposed evaluation functions) than the current incumbent solution φ^* (lines 6–8). These three steps, considered as an iteration, are repeated until a maximum predefined CPU time MT is consumed. Finally, the best solution found φ^* is returned as the result of the ILS procedure. Note that our ILS implementation is not equipped with an explicit mechanism to prevent the search from generating cycles (i.e., revisiting previously selected solutions). The experiments discussed in Section 5.6 demonstrate that the alternative evaluation functions reduce the emergence of such cycles.

Algorithm 1: Iterated Local Search algorithm.

Input: Maximum allowed CPU time MT, perturbation strength PS

1: Choose an initial solution $\varphi \in \phi$ uniformly at random
2: $\varphi^* \leftarrow$ SteepestDescent(φ)
3: **repeat**
4: $\quad\varphi' \leftarrow$ Perturbation(φ^*, PS)
5: $\quad\varphi'' \leftarrow$ SteepestDescent(φ')
6: $\quad$**if** $f(\varphi'') < f(\varphi^*)$ **then**
7: $\quad\quad\varphi^* \leftarrow \varphi''$
8: $\quad$**end if**
9: $\quad cpuTime \leftarrow$ getElapsedCpuTime()
10: **until** $cpuTime < MT$
11: **return** a local minimum φ^*

The proposed local search method follows a typical *Steepest Descent* (SD) strategy. The neighborhood structure $\mathcal{N}(\varphi)$ implemented for our SD algorithm can be formally defined as:

$$\mathcal{N}(\varphi) = \left\{\varphi' \in \Phi : \text{swap}(\varphi, u, v) = \varphi',\ u, v \in V,\ u \neq v\right\}, \tag{7}$$

where $\text{swap}(\varphi, u, v)$ is a function permitting to exchange the labels of two vertices u and v from an embedding φ. Given a graph G of order n, the size of such a neighborhood is $|\mathcal{N}(\varphi)| = n(n-1)/2$.

The proposed local search proceeds as follows. Given a feasible solution $\varphi \in \Phi$, SD replaces it with the best solution found in its neighborhood $\mathcal{N}(\varphi)$, where ties are broken randomly. This iterative process terminates automatically when φ is locally optimal, i.e., no better embedding can be found within its neighborhood.

The motivation for using such a simple local search algorithm is the following. First, SD can be an appropriate option for measuring the effect of changing the evaluation scheme because its performance is only determined by the guiding capabilities of the analyzed evaluation functions, once a neighborhood relation has been fixed (Blum & Roli, 2003; Gendreau & Potvin, 2010). Furthermore, it is expected that the SD algorithm stops after a small number of iterations, given the poor capacity of discrimination furnished by some of the studied evaluation approaches. Second, the parameter-free nature of the SD algorithm allows to avoid the influence of a non appropriate tunning process over the behavior induced by the studied evaluation functions.

Within our ILS implementation, the perturbation operator (see Algorithm 2) is designed to jump out of the current local optimum trap by accepting some deteriorating solutions. A preliminary experiment testing different perturbation functions allowed us to identify that the perturbation procedure described next is the one that attains the best results. This perturbation mechanism helps the local search to escape from the basin of attraction of the local optima found, without an important cost deterioration of the new incumbent solution. Given an embedding, passed as a parameter, our perturbation operator executes a predefined number PS of controlled label exchange moves swap($\cdot$), where PS is called the strength of perturbation. More in detail, for each perturbation step (i.e., each iteration of the outer *for* loop, line 2), the best embedding among $|V|$ randomly sampled neighbor solutions of the current solution $\varphi^\star$ is identified (lines 3–9). Then, this best embedding φ' is used to replace the current solution $\varphi^\star$ (line 10), which will be the starting point for the next perturbation step. The new

Algorithm 2: Perturbation.

Input: Input solution φ, perturbation strength PS

1: $\varphi^\star \leftarrow \varphi$
2: **for** $L \leftarrow 1$ **to** PS **do**
3: $\quad$Choose randomly a neighbor solution $\varphi' \in N(\varphi^\star)$
4: $\quad$**for** $\ell \leftarrow 1$ **to** $|V|$ **do**
5: $\quad\quad$Choose randomly a neighbor solution $\varphi'' \in N(\varphi^\star)$
6: $\quad\quad$**if** $f(\varphi'') < f(\varphi')$ **then**
7: $\quad\quad\quad\varphi' \leftarrow \varphi''$
8: $\quad\quad$**end if**
9: $\quad$**end for**
10: $\quad\varphi^\star \leftarrow \varphi'$
11: **end for**
12: **return** a perturbed solution $\varphi^\star$

[3] Note that Cbs can also be incrementally computed.

embedding generated after these *PS* perturbation steps is returned as the incumbent solution of the next round of the local search procedure. Please note that this perturbation procedure is guided by an evaluation function f (line 6) which can be one of the four analyzed evaluation functions in this work.

4. Experimental setup

The experimentation in this research work was carried out on 20 benchmark instances[4] previously reported in the literature (Hamon et al., 2016; Satsangi et al., 2012). These instances are grouped into four different subsets which include: six Cartesian products of graphs with known upper bounds (Jianxiu, 2001), five standard graphs with known optimal solution values (Chen & Yan, 2007; Jianxiu, 2001), five graphs arising from scientific and engineering practical problems coming from the Harwell–Boeing Sparse Matrix Collection,[5] and four random graphs constructed with the Erdös–Rényi generator provided by *NetworkX 1.11* (Hagberg, Schult, & Swart, 2016; 2008).

Even though new alternative evaluation schemes for the *CBS* problem are analyzed in this work, it is worth mentioning that the target of the optimization process remains to minimize the total cyclic bandwidth sum (Cbs). Therefore, all the results furnished by this experimental comparison are assessed with respect to the conventional evaluation function of the *CBS* problem.

Two additional performance measures were considered, both computed over multiple independent executions of the implemented search algorithms. First, the *relative root mean square error* (RMSE) for a given test instance t is defined as follows:

$$\text{RMSE}(t) = 100\%\sqrt{\frac{\sum_{r=1}^{R}\left(\frac{\text{Cbs}_r(t)-\text{Cbs}^*(t)}{\text{Cbs}^*(t)}\right)^2}{R}}, \tag{8}$$

where $\text{Cbs}_r(t)$ denotes the cyclic bandwidth sum of the best embedding found during a single execution r, R is the total number of executions carried out in the experiment, and $\text{Cbs}^*(t)$ is the optimal (or best-known) solution value for instance t. Given that the range of possible cost values varies from instance to instance in the *CBS* problem, RMSE(t) is defined in a common 0% to 100% scale, making possible to evaluate together the results obtained for the different considered test instances. The preferred value for this measure is $\text{RMSE}(t) = 0\%$, corresponding to a perfect performance.

Second, the *overall relative root mean square error* (O-RMSE) measure extends RMSE in order to evaluate the global performance of the studied approaches, considering all the set of benchmark instances. Formally, O-RMSE can be stated as follows:

$$\text{O-RMSE} = \frac{1}{|\mathcal{T}|}\sum_{t\in\mathcal{T}}\text{RMSE}(t), \tag{9}$$

where $\mathcal{T}$ is the set of all benchmark instances. In this way, $\text{O-RMSE} = 0\%$ suggests the ideal situation where the optimal solution value for each instance was attained at each performed execution.

A statistical significance analysis was performed for all the experiments presented in this paper. Each analysis was conducted using the following methodology. First, the normality of data distributions was evaluated by applying the *Shapiro–Wilk* test. In the case of normally distributed data, either *ANOVA* or *Welch's t* parametric tests were used depending on the result of *Bartlett's* test, which was used to investigate if the variances across the samples were homogeneous (*homoskedasticity*) or not. On the contrary, the nonparametric *Kruskal–Wallis* test was employed for non-normal data. These tests are carried out by consistently considering a significance level of 0.05.

The C programming language was used to make the algorithmic implementations needed for this experimental study. These implementations were then compiled with *gcc* using the optimization flag *-O3* and executed sequentially into a CPU Xeon X5650 at 2.66 gigahertz, 2 gigabytes of RAM with Linux operating system.

5. Discussion and analysis of the alternative evaluation functions

In this section four different experiments are presented to evaluate and to compare the four different evaluation functions for the *CBS* problem: the conventional evaluation function Cbs, and three new alternative evaluation schemes introduced in Section 2. Through the first pair of experiments two essential characteristics of the analyzed evaluation functions are examined in detail in Sections 5.1 and 5.2. Then, the search guiding efficiency of these evaluation schemes is assessed in Sections 5.3 and 5.4.

5.1. Potential of discrimination

The potential of discrimination is an essential property of any evaluation scheme that has direct influence on the global behavior of metaheuristic algorithms. If an evaluation function is unable to establish an appropriate ranking among candidate solutions, then the optimization process could be practically guided by random decisions.

The potential of discrimination provided by the analyzed evaluation approaches is investigated next. This is carried out by studying the distribution of ranks that these evaluation methods produce for a given set of potential embeddings. A ranking establishes an order relation over the items contained in a set by considering a predefined criterion. In this research work, embeddings should be ranked and the criterion to establish such an order relation is their cyclic bandwidth sum value (cost). Given a set of potential embeddings, the first ranking position is assigned to the solution with the best (smaller) Cbs value, the next ranking position is allotted to the one with the second best cyclic bandwidth sum value, and so forth. If two or more embeddings present the same cost, then they will share the same rank.

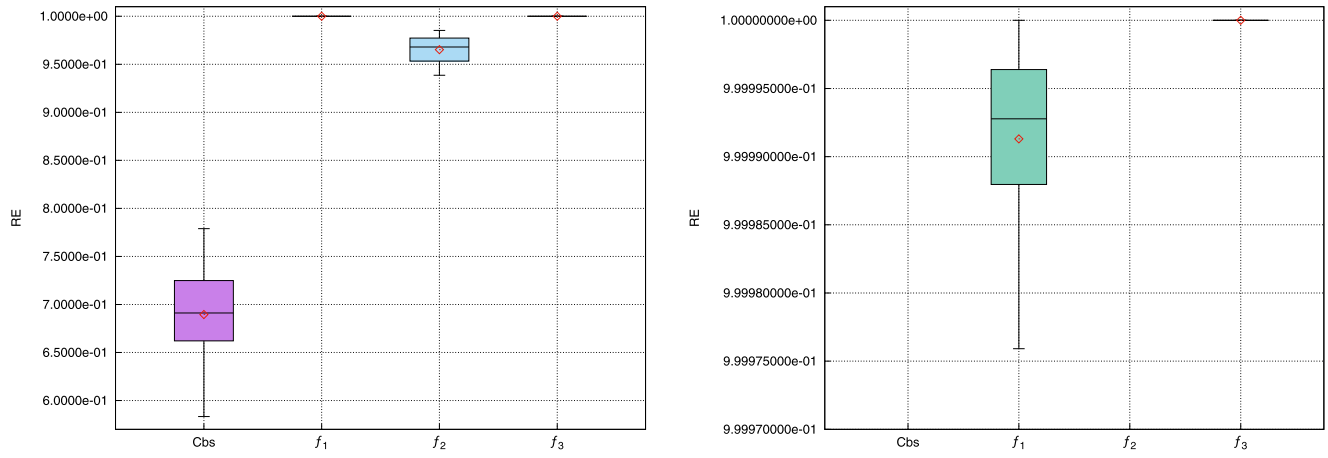
The *relative entropy* (RE) measure proposed by Corne and Knowles (2007) was adopted for this experiment. Given a set having c ranked embeddings (there are at most c ranks, and at least 1), the relative entropy RE(D) for the distribution of ranks D can be computed with the following expression:

$$\text{RE}(D) = \frac{\sum_{j=1}^{c}\frac{D_j}{c}\log\left(\frac{D_j}{c}\right)}{\log\left(\frac{1}{c}\right)}, \tag{10}$$

where D_j denotes the number of embeddings with rank j. RE(D) tends to 1 as the rank distribution D approaches to the ideal case where each embedding has a different rank (i.e., the maximum potential of discrimination). On the contrary, when all the potential solutions share the same ranking position (i.e., the weakest discrimination), RE(D) equals zero.

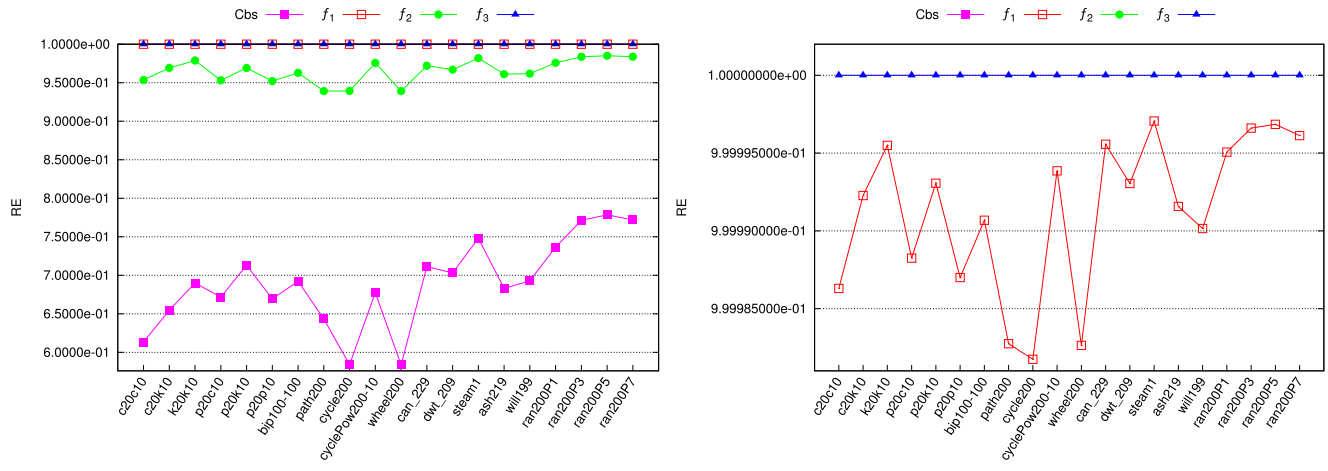
In this experiment, 100,000 different embeddings were generated at random. Using each one of the four studied evaluation functions these embeddings were evaluated and ranked in order to compute the corresponding RE measures. This experiment was performed 50 times using all the selected benchmark instances. The global results produced by this experiment are summarized with boxplots in Fig. 2, while Fig. 3 depicts for each test instance the average RE values obtained by the distinct analyzed functions using points.

[4] Available at http://www.tamps.cinvestav.mx/~ertello/cbsp.php.
[5] http://math.nist.gov/MatrixMarket/data/Harwell-Boeing.



(a) Overall view. **(b)** Zoom-in view.

Fig. 2. Boxplots representing the global relative entropy (RE) of the distribution of ranks computed for each studied evaluation function. Overall statistics for the 20 selected benchmark instances.



(a) Overall view. **(b)** Zoom-in view.

Fig. 3. Relative entropy (RE) of the distribution of ranks computed for each analyzed evaluation function. Each point represents the average of 50 independent executions over one tested instance.

By inspecting Fig. 2(a) it is possible to observe that certain of the studied evaluation functions are able to discriminate stronger than others. In all the benchmark instances tested, the conventional evaluation function for the *CBS* problem, Cbs, achieved in average the lowest RE score (6.89496912e−01). This confirms the weak capacity for discrimination furnished by this function, which has been the principal motivation for exploring the use of alternative evaluation approaches. Among the alternative functions, f_2 presented the worst performance in terms of discrimination (9.65213479e−01). The evaluation function f_1 most of the time scores high RE values. Nevertheless, it suffered slight decreases on some of the tested instances, see Fig. 3(b), leading to an average RE value of 9.99913033e−01 as can be appreciated in the zoom depicted in Fig. 2(b). Finally, it is important to remark that according to the obtained results (9.99999999e−01), f_3 is the function offering the higher capacity for discrimination among potential solutions.[6] Furthermore, this property is conserved over all the graph topologies evaluated, see Fig. 3(a) and its zoom-in view in Fig. 3(b).

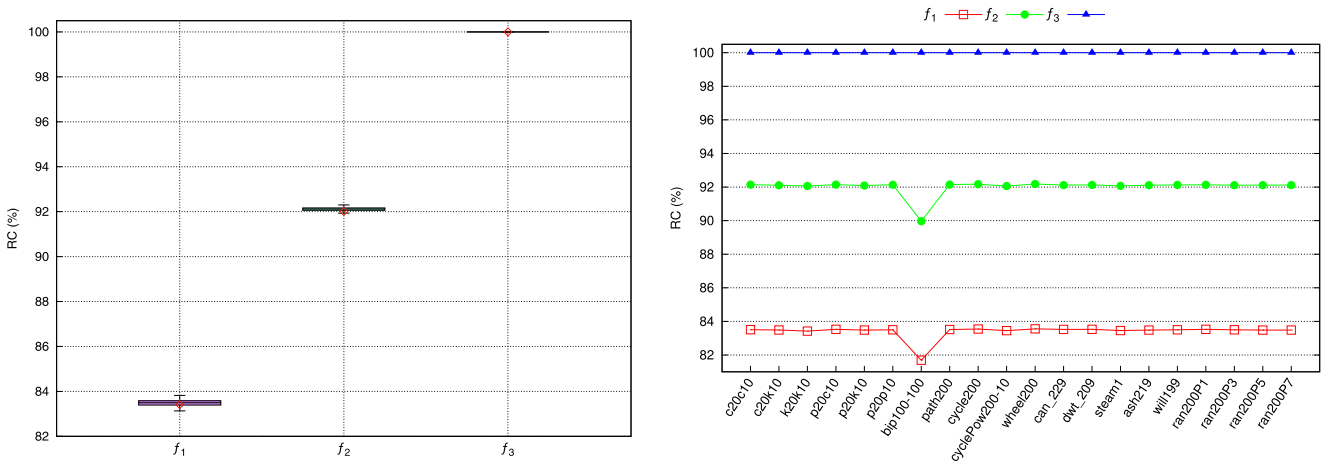
[6] Note that double precision floating point numbers were used for computing the RE values presented in this experiment.

5.2. CBS-*compatibility*

The alternative evaluation functions for the *CBS* problem introduced in Section 2 aim at performing a more effective exploration through the space of potential embeddings. Nevertheless, they should remain consistent with the original objective of *CBS* problem, which consists in minimizing the cyclic bandwidth sum function Cbs. Otherwise, false optima can potentially be introduced and the search algorithm could be oriented towards embeddings diverging from the optimal solutions of the original optimization problem. Hence, investigating whether or not the proposed evaluation schemes are consistent with the original objective is an important issue.

In this work, functions respecting this requirement (not contradicting function Cbs) are said to be CBS-*compatible*. Thus, the capacity of an alternative evaluation function to preserve the conventional rank ordering among potential embeddings can be defined as its *CBS*-compatibility. More formally:

Definition 1. An alternative evaluation function $f : \Phi \rightarrow \mathbb{R}$ is said to be *CBS*-compatible if and only if $f(\varphi) < f(\varphi') \Rightarrow \text{Cbs}(\varphi) \leq \text{Cbs}(\varphi')$ for every pair of solutions $\varphi, \varphi' \in \Phi$. Otherwise, if at least one pair of embeddings φ, φ' exists such that $\text{Cbs}(\varphi) < \text{Cbs}(\varphi')$ but $f(\varphi) > f(\varphi')$, then function f is not *CBS*-compatible.



(a) Overall statistics for the 20 selected benchmark instances.

(b) Average results for each benchmark instance.

Fig. 4. Relative compatibility (RC) values computed for each alternative evaluation scheme studied.

Note, however, that the case where $\text{Cbs}(\varphi) = \text{Cbs}(\varphi')$ but $f(\varphi) \neq f(\varphi')$ is not considered a contradiction. This is a convenient scenario, because the main objective of using the alternative function f is to enable a more fine-grained discrimination among potential embeddings.

In this section, the *CBS*-compatibility property is explored for all the alternative evaluation functions introduced in this work. An experiment was conducted where 100,000 different embeddings were randomly generated, then all pairwise comparisons among them were performed. The percentage of such comparisons where an agreement of the alternative evaluation approach with respect to the conventional one is computed and referred to as *relative compatibility* (RC).

Even though the value RC = 100% does not offer the assurance of the *CBS*-compatibility property for a given function, RC < 100% is sufficient to negate it. In other words, the gravity of the cases where the *CBS*-compatibility property is not satisfied could be assessed with the aid of the RC value. For every selected benchmark instance, 50 independent repetitions of this experiment were performed. The overall statistics produced in this experiment are depicted as boxplots in Fig. 4(a), while the average RC values obtained for each one of the tested instances are presented in Fig. 4(b).

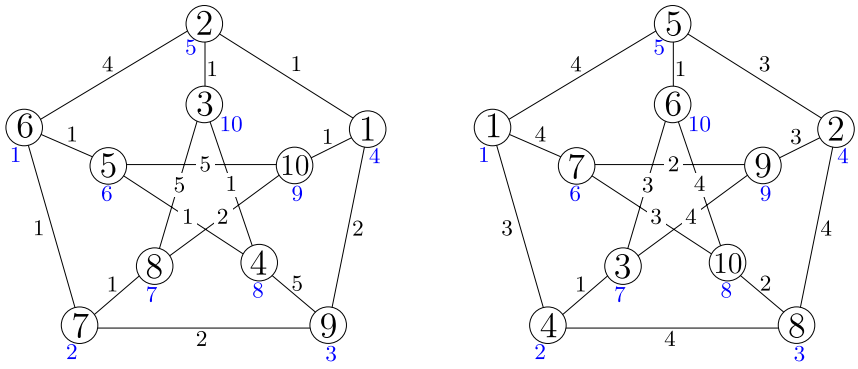
From Fig. 4(a) and (b), it is possible to note that function f_3 showed 100% of agreement with the conventional Cbs evaluation function for all the benchmark instances used in this experiment. Thus, it appears that evaluation function f_3 is *CBS*-compatible. On the contrary, the experimental results disclose that functions f_1 and f_2 do not present the *CBS*-compatibility property for any of the graph topologies evaluated, since they scored an average RC value of 83.410% and 92.012%, respectively. One can observe, from Fig. 4(b), that functions f_1 and f_2 reached slightly lower average RC values (81.692% and 89.968%) for the instance *bip100-100* (a complete bipartite graph of order $n = 200$). After analyzing this particular case, we noticed that the embeddings for this instance generated in average a distribution of the number of cyclic distances with $d_k = 100.503$ for $1 \leq k \leq 99$ and $d_{100} = 50.236$, while most of the other tested instances induced distributions with at most $d_k = 20.010$ for $1 \leq k \leq 99$ and $d_{100} = 9.984$. Furthermore, in average only 5,406 different cost values were produced by the evaluation function Cbs for assessing the 100,000 embeddings produced for the instance *bip100-100* in this experiment. In contrast, the alternative evaluation functions f_1 and f_2 delivered much more different cost values (99,995 and 72,200, respectively). These situations could explain the difficulties of the alternative evaluation functions for respecting the extremely restrictive order relation established by the conventional evaluation function Cbs for this particular instance.

Finally, it can be highlighted the poor performance exhibited by function f_1. For all the selected benchmark instances, this evaluation scheme scored the lowest RC values leading to an average of 83.410%. Fig. 5 presents an example scenario where function f_1 contradicts the conventional function Cbs. In this example, a pair of different embeddings φ and φ' for a Petersen graph of order $n = 10$ are compared against each other by applying the Cbs and f_1 evaluation functions. As a result, the conventional evaluation function Cbs prefers the solution φ (with a smaller value $\text{Cbs}(G, \varphi) = 33$) to φ' (with value $\text{Cbs}(G, \varphi) = 45$), while function f_1 delivers the inverse preference order among them.

The low RC values attained by certain analyzed evaluation schemes, particularly f_1, bring evidence of a serious issue. It is more likely that the global optimum produced by an alternative evaluation function differs largely from the global optimum of the original optimization problem when this function scores lower RC values. Therefore, it is expected that *CBS*-incompatible alternative functions could have problems to effectively guide the search process.

5.3. Search performance using a basic local search algorithm

The experimental performance of the SD algorithm was evaluated when using each of the presented evaluation functions. Through this experimentation, the following methodology was consistently followed. First, 50 embeddings were randomly generated for every one of the 20 studied benchmark instances. Then, each one of these embeddings was used as initial solution for one independent run of the four analyzed evaluation schemes over each considered graph. The main objective was to guarantee that all the studied functions start the search of better solutions from the same set of initial embeddings, and thus to reduce the bias produced by this important design element in our comparisons. The average results achieved in this experimental comparison are provided in Table 1. The first three columns in this table list the instance (graph), its order ($|V|$) and size ($|E|$). Column four reports the best-known cyclic bandwidth sum value (B) for each instance. It was either attained by MACH, when compiled and executed



(a) Embedding φ. **(b)** Embedding φ'.

Fig. 5. Two different embeddings for a Petersen graph of order $n = 10$. This is an example where function f_1 contradicts function Cbs: $\text{Cbs}(\varphi) = 33 < \text{Cbs}(\varphi') = 45$ but $f_1(\varphi) = 810 > f_1(\varphi') = 800$.

independently (50 times) in our computational platform,[7] or by our SD algorithm when value is marked with a $\star$. For each of the compared evaluation functions three columns are used to depict the average (*Avg*) and standard deviation (*Dev*) of the cyclic bandwidth sum attained with the SD algorithm equipped with the corresponding solution assessment scheme, and the average number of iterations (I) needed to obtain that solution cost. The O-RMSE value computed for each compared evaluation scheme is presented at the bottom of the table. The lowest average cyclic bandwidth sum value (*Avg*) for each of the tested instances is shown in bold.

The data presented in Table 1 allow us to observe that among the three considered alternative evaluation functions for *CBS* problem, f_1 has the most negative effect on the overall efficiency of the SD algorithm (it has the highest O-RMSE value 169.67%). This could be explained by its poor relative *CBS*-compatibility (RC = 83.410%) evidenced in the experiments presented in Section 5.2. Function f_2 has a better performance than both, f_1 and the original objective function (Cbs); and is able to reach for 4 instances the smaller average cyclic bandwidth values. For 80% of the tested benchmark instances the best performer is clearly the SD algorithm using function f_3 (attaining an O-RMSE = 76.51%). Notice that the search with f_3 last always longer time than with Cbs (contrast the values in columns 7 and 16), leading to better quality local optima. The conventional function Cbs stops early because it is unable to differentiate among neighboring embeddings having equal cyclic bandwidth sum value.

The methodology described in Section 4 was used to perform a statistical significance analysis of the results produced by this experiment. Table 2 highlights whether the performance differences between the studied evaluation approaches were statistically significant or not. Each row in this table compares two strategies, say A and B, which is denoted as A/B. If a significant performance difference exists between A and B for a particular instance, the corresponding cell is marked either + or − providing that such a difference was in favor of, or against A. Please note that a significant performance difference against A, marked as −, implies that B significantly outperformed A. Unmarked cells indicate that there was not a statistically important difference between the compared evaluation strategies. The rightmost column of the table summarizes the results of this analysis.

As shown in Table 2, Cbs significantly outperformed functions f_1 and f_2 in 15 and 10 of the instances. Function f_3 achieved a statistically significant performance increase with regard to Cbs for 9 of the adopted test graphs, while for the rest of the benchmarks a significant difference could not be concluded. The f_1 and f_2 evaluation strategies scored significantly worst results than f_3 in 18 and 14 instances, respectively. Nevertheless, f_3 was significantly surpassed by f_2 in 2 of the benchmark instances.

5.4. Search performance using an Iterated Local Search algorithm

The main objective of the first experiment presented in this section is to analyze the performance fluctuations of our ILS implementation produced when varying the selected evaluation function and the input parameter values for the perturbation strength (*PS*) and the maximum allowed CPU time in seconds (*MT*). To this end, a full factorial experimental design was carried out considering the four studied evaluation schemes, as well as three different values for each of the parameters $PS = \{5, 10, 15\}$, and $MT = \{300, 600, 900\}$ (see Algorithm 1). This leads to a total of 36 configurations for our ILS implementation. All these configurations were evaluated using 50 independent executions over each of the 20 selected benchmark instances for a total of 36,000 executions. Fig. 6 presents the overall relative root mean square error (O-RMSE) measure obtained by each of the analyzed ILS configurations, grouped by the MT parameter values. Lower O-RMSE values are preferred, see Section 4.

Fig. 6 reveals that function f_1 consistently presented the worst search performance for the different ILS parameter configurations tested. It seems that function f_1 is negatively affected by higher perturbation strength values, even when the maximum allowed computational time *MT* is augmented. This can be the consequence of the incapability of function f_1 to guide the search to better local optima after a medium or high strength perturbation, produced by its low relative compatibility with respect to the original objective of the *CBS* problem (see Section 5.2). In contrast, the conventional evaluation function Cbs has a much better performance than f_1, and is outperformed by f_2 when the maximum allowed computational time *MT* is limited to 300 seconds. Nevertheless, the results provided by Cbs are slightly better than those furnished by f_2 when *MT* is augmented. From Fig. 6 it can also be observed that f_3 was able to consistently outperform the search performance provided by the other 3 analyzed functions in the 9 ILS parameter configurations tested. Furthermore, by analyzing the behavior of this particular function one observes that the ILS algorithm equipped with f_3 is able to discover better new local optima when both the perturbation strength and the CPU time allowed are increased. This remarkable performance can be related with the fact that function f_3 scored the best discrimination potential and *CBS*-compatibility in the precedent experiments.

In order to provide a more detailed analysis, the parameter configurations which allowed each of the studied evaluation

[7] Using the input parameters suggested by their authors (Hamon et al., 2016).

Table 1
Performance of the SD algorithm when using the four studied evaluation functions.

				Cbs			f_1			f_2			f_3		
Graph	$\|V\|$	$\|E\|$	B	*Avg*	*Dev*	I	*Avg*	*Dev*	I	*Avg*	*Dev*	I	*Avg*	*Dev*	I
c20c10	200	400	2360*	4993.24	585.13	1561.46	8692.44	683.55	4766.96	6049.16	632.13	4739.20	**4197.64**	703.10	3803.36
c20k10	200	1100	5300	10976.28	1304.90	1873.26	13787.64	2949.38	4820.22	**9525.32**	923.80	3378.38	11067.44	1366.13	2607.32
k20k10	200	2800	62,300*	62436.00	755.59	2280.46	84772.16	862.00	3302.62	62304.92	2.50	3633.68	**62300.00**	0.00	3455.10
p20c10	200	390	2256	4809.28	600.77	1572.40	8342.26	738.01	4888.98	5585.40	802.18	4873.40	**4062.96**	551.73	3518.82
p20k10	200	1090	5200	10537.88	1227.61	1899.76	13101.64	3262.07	4727.20	**9056.20**	804.87	3393.70	10464.00	1104.77	2596.04
p20p10	200	370	2385	4082.90	619.52	1653.46	7315.62	926.58	4827.54	4657.02	730.89	5159.54	**3354.56**	489.50	3335.00
bip100-100	200	10,000	500,000	**500000.00**	0.00	115.02	**500000.00**	0.00	914.74	**500000.00**	0.00	870.84	**500000.00**	0.00	133.94
path200	200	199	199	1817.30	209.11	831.34	1799.12	260.51	6812.78	1366.92	219.50	5961.18	**1066.42**	175.07	4975.28
cycle200	200	200	200	1888.56	207.06	850.68	1894.00	262.91	6639.96	1445.96	242.10	6201.88	**1080.16**	200.13	5226.80
cycleP200-10	200	2000	11,000*	20534.36	4437.49	2851.96	18292.24	7242.48	7333.08	**15249.36**	6166.86	6184.46	15624.40	5275.83	5543.84
wheel200	200	398	10,200	11885.32	206.32	819.16	11854.64	261.79	6801.38	11455.80	254.86	6136.12	**11082.36**	173.86	5158.52
can_229	229	774	6301*	9281.88	1409.97	3260.88	15772.50	2090.70	6498.38	10778.56	1910.59	6577.76	**8577.52**	1566.76	4946.46
dwt_209	209	767	7119*	**8964.78**	817.21	2171.70	13774.14	1721.82	6805.96	9382.72	1410.75	6062.88	8978.40	681.48	2815.02
steam1	240	1761	24,158*	30878.12	3037.17	3422.12	43278.28	5591.47	4852.92	31692.08	4180.80	4890.12	**30305.04**	3396.44	3753.00
ash219	219	431	6705*	8269.36	542.69	1953.90	12586.34	472.12	4731.16	9480.28	652.37	5555.60	**7848.82**	645.95	4256.12
will199	199	660	14,116*	15722.04	653.69	2239.12	21010.08	606.12	3786.00	17658.28	817.34	4135.80	**15522.24**	629.90	3361.78
ran200P1	200	1991	71,394*	72802.78	705.67	2422.18	80034.24	938.28	3039.18	75927.04	654.93	3567.24	**72693.96**	619.91	3269.74
ran200P3	200	5970	256,987*	259502.00	1061.70	2770.02	269017.16	1041.76	2812.12	263671.86	1043.50	3453.40	**259250.16**	923.42	3476.50
ran200P5	200	9955	452,486*	455109.68	1167.33	2955.38	465122.66	1223.02	2646.84	459212.96	1329.09	3380.70	**454989.04**	1178.27	3542.62
ran200P7	200	13,827	651,128*	653407.40	1134.01	2816.72	661903.78	1417.15	2274.00	656706.26	1221.45	3135.66	**653187.90**	837.67	3448.70
	O-RMSE				122.86%			169.67%			105.44%			**76.51%**	

Table 2
Statistical analysis for comparing the performance of the SD algorithm when using the four analyzed evaluation approaches.

	Instance																				
Function	c20c10	c20k10	k20k10	p20c10	p20k10	p20p10	bip100-100	path200	cycle200	cycleP200-10	wheel200	can_229	dwt_209	steam1	ash219	will199	ran200P1	ran200P3	ran200P5	ran200P7	Overall
Cbs / f_1	+	+	+	+	+	+						+	+	+	+	+	+	+	+	+	15 + 0 −
Cbs / f_2	+	−	−	+	−	+		−	−	−	−	+			+	+	+	+	+	+	10 + 7 −
Cbs / f_3	−			−		−		−	−	−	−	−			−						0 + 9 −
f_1 / f_2	−	−	−	−	−	−		−	−	−	−	−	−	−	−	−	−	−	−	−	0 + 19 −
f_1 / f_3	−	−	−	−	−	−		−	−		−	−	−	−	−	−	−	−	−	−	0 + 18 −
f_2 / f_3	−	+	−	−	+	−		−	−		−	−			−	−	−	−	−	−	2 + 14 −

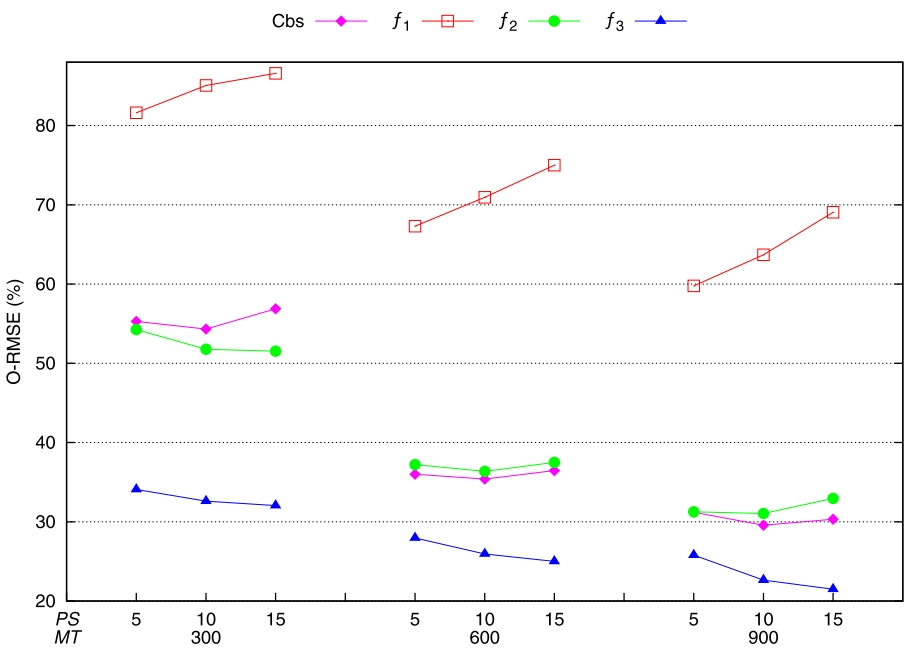


Fig. 6. Overall relative root mean square error (O-RMSE) obtained by 9 different parameter configurations of the ILS algorithm equipped with each of the four analyzed evaluation function.

Table 3
Selected parameter settings for the ILS algorithm.

Function	*MT*	*PS*
Cbs	900	10
f_1	900	5
f_2	900	10
f_3	900	15

functions to reach the lowest O-RMSE value have been selected (see Table 3). The detailed results from this experimental comparison are depicted in Table 4 by using the same column headings defined previously for Table 1. Column four reports the best-known cyclic bandwidth sum value (B), which was obtained either by MACH, when compiled and executed independently (50 times) in our computational platform, or by our ILS algorithm when value is marked with a $\star$. For each of the tested instances the lowest average cyclic bandwidth sum value (Avg) is shown in bold.

It is evident, from Table 4, that employing f_3 for assessing solution quality within the ILS algorithm permits to obtain for 12 out of 20 instances better results (smaller average cyclic bandwidth sum values, Avg) than those produced by the other three compared evaluation approaches, resulting in the smallest O-RMSE value (21.50%). In the particular case of instance *bip100-100*, function f_3 attains the same results than the best performing approaches. In fact, it seems that instance *bip100-100* is not as hard to solve as other graphs in the test suite, since all the evaluation schemes found the same average cyclic bandwidth sum with a zero standard deviation (Dev).

A statistical significance analysis was carried out on the experimental results presented above by employing the methodology described in Section 4. The details of this analysis are depicted in Table 5 using the same format and conventions presented in Section 5.3. From Table 5 one observes that even when Cbs significantly outperformed functions f_1 and f_2 in 16 and 10 benchmark instances, it scored significantly worse results than f_3 in 12 graphs. Function f_3 significantly increased the performance of the ILS algorithm in 16 and 14 of the test cases with respect to f_1 and f_2, respectively. However, f_1 was able to reach a statistically significant performance improvement with regard to the three other tested evaluation schemes for the instances *c20k10* and *p20k10*.

5.5. Influence of the evaluation scheme over the ILS convergence process

In order to further investigate the degree of influence that the different evaluation schemes have over the convergence process of the ILS algorithm, their evolution profiles of the average best cyclic bandwidth sum (Cbs) along the search process (ILS iterations) were computed using the data produced in the previous experiment and individually analyzed. To reduce the space needed for reporting the results of this analysis, they are presented using plots (see Fig. 7) for only four instances (one representing each subset of graphs described in Section 4): *c20k10*, *path200*, *dwt_209* and *ran200P1*. In these figures each line represents the average result of 50 executions, and points indicate improvements of the incumbent solution.

The plot in Fig. 7(a) permits to find out why function f_1 was able to statistically outperform the rest of the evaluation approaches when solving the instance *c20k10*. It is possible to observe that starting from iteration 6, the convergence curve for f_1 presents more frequent improvements (points) of the incumbent solution compared with the other curves in the plot. It indicates that this evaluation approach exhibits a greater capacity to detect a promising search direction for this particular instance. It is worth noting, that f_1 and f_2 avoid getting stuck in local optima, because they allow deteriorating cost moves as a consequence of their 83.49% and 92.11% of agreement with the conventional evaluation function. In contrast, function Cbs gets easily trapped in local optima due to its inability to perform a proper discrimination among potential embeddings.

A quite different scenario can be observed in Fig. 7(b). Function Cbs reached the best average solution value for the graph *path200* faster than the other evaluation approaches (similar results were obtained for the graph *cycle200*). For this particular instance the neutrality of the search landscape is high, since Cbs takes only 9801 possible values ($199 \leq \text{Cbs} \leq 10,000$) to discriminate among a total of 3.9416E+372 potential embeddings (see Section 2). How-

Table 4
Performance of the ILS algorithm when using the four studied evaluation functions.

				Cbs			f_1			f_2			f_3		
Graph	$\|V\|$	$\|E\|$	B	*Avg*	*Dev*	I	*Avg*	*Dev*	I	*Avg*	*Dev*	I	*Avg*	*Dev*	I
c20c10	200	400	2360*	2521.36	489.60	26.10	3492.92	1228.57	139.02	**2395.96**	254.28	73.30	2422.44	309.24	246.38
c20k10	200	1100	5300	10251.40	946.65	2.44	**6725.40**	1253.63	37.08	7596.00	943.50	20.88	7943.28	1502.48	39.02
k20k10	200	2800	62,300*	63755.28	2022.45	1.02	83941.12	925.83	37.42	62300.24	0.66	23.86	**62300.00**	0.00	33.08
p20c10	200	390	2238*	2455.68	399.46	25.42	3251.12	897.94	140.94	**2263.92**	1.66	71.70	2335.58	268.12	240.90
p20k10	200	1090	5200	9925.76	1040.17	2.96	**6169.62**	702.93	38.20	7016.00	831.61	21.72	7119.30	1228.09	43.64
p20p10	200	370	1991*	2092.58	168.86	47.42	2684.10	363.98	141.56	2199.94	152.18	61.78	**2015.12**	34.83	229.06
bip100-100	200	10,000	500,000*	**500000.00**	0.00	8.96	**500000.00**	0.00	4.28	**500000.00**	0.00	4.50	**500000.00**	0.00	8.06
path200	200	199	199	**352.12**	69.76	57.40	738.94	65.79	111.18	489.66	74.28	36.88	363.26	51.51	97.92
cycle200	200	200	200	**378.68**	81.20	58.76	760.92	71.55	107.72	531.28	80.94	37.84	389.08	56.41	93.16
cycleP200-10	200	2000	11,000*	16951.28	5441.33	22.70	**15153.60**	5602.16	61.02	15230.60	6110.02	64.14	16238.20	5297.60	48.62
wheel200	200	398	10,200	10439.72	75.58	27.70	10837.24	74.86	56.22	10611.28	89.39	18.68	**10421.64**	53.25	49.98
can_229	229	774	6243*	6915.46	1169.85	12.80	9402.98	1988.30	61.26	6671.48	885.43	31.30	**6264.80**	4.23	91.88
dwt_209	209	767	6355*	7369.46	508.75	9.48	9629.50	962.30	50.18	7587.88	525.68	26.92	**6677.90**	328.73	120.04
steam1	240	1761	24,158*	31853.64	3633.94	1.00	45230.10	4189.08	1.00	34145.88	3490.00	1.00	**31261.18**	2842.05	1.00
ash219	219	431	6229*	6644.94	239.91	11.32	9766.48	490.43	130.04	7357.54	352.16	40.40	**6410.88**	216.71	137.30
will199	199	660	13,699*	13976.68	218.81	18.58	17658.06	625.05	115.94	15132.26	256.08	48.82	**13801.82**	50.37	144.76
ran200P1	200	1991	69,993*	71291.26	386.19	5.84	76954.72	564.40	52.44	73234.92	456.13	26.64	**70444.86**	230.92	44.64
ran200P3	200	5970	255,001*	258634.08	931.37	1.30	265569.88	941.25	21.90	260282.90	811.22	10.24	**256094.34**	467.71	13.34
ran200P5	200	9955	451,266*	455114.62	995.99	1.00	461683.78	1167.28	12.74	456302.98	826.20	6.42	**452517.20**	592.95	6.24
ran200P7	200	13,827	648,917*	654312.52	1174.19	1.00	658789.40	1049.85	11.08	654406.30	885.77	4.34	**651048.42**	697.50	4.30
	O-RMSE				29.57%			59.76%			31.05%			**21.50%**	

Table 5
Statistical analysis for comparing the performance of the ILS algorithm when using the four analyzed evaluation approaches.

	Instance																					
Function	c20c10	c20k10	k20k10	p20c10	p20k10	p20p10	bip100-100	path200	cycle200	cycleP200-10	wheel200	can_229	dwt_209	steam1	ash219	will199	ran200P1	ran200P3	ran200P5	ran200P7	Overall	
Cbs / f_1	+	−	+	+	−	+		+	+		+	+	+	+	+	+	+	+	+	+	16 +	2 −
Cbs / f_2		−	−	−	−	+		+	+		+	−		+	+	+	+	+	+		10 +	5 −
Cbs / f_3		−	−	−	−	−							−		−	−	−	−	−	−	0 +	12 −
f_1 / f_2	−	+	−	−	+	−		−	−		−	−	−	−	−	−	−	−	−	−	2 +	16 −
f_1 / f_3	−	+	−	−	+	−		−	−		−	−	−	−	−	−	−	−	−	−	2 +	16 −
f_2 / f_3			−	+		−		−	−		−	−	−	−	−	−	−	−	−	−	1 +	14 −

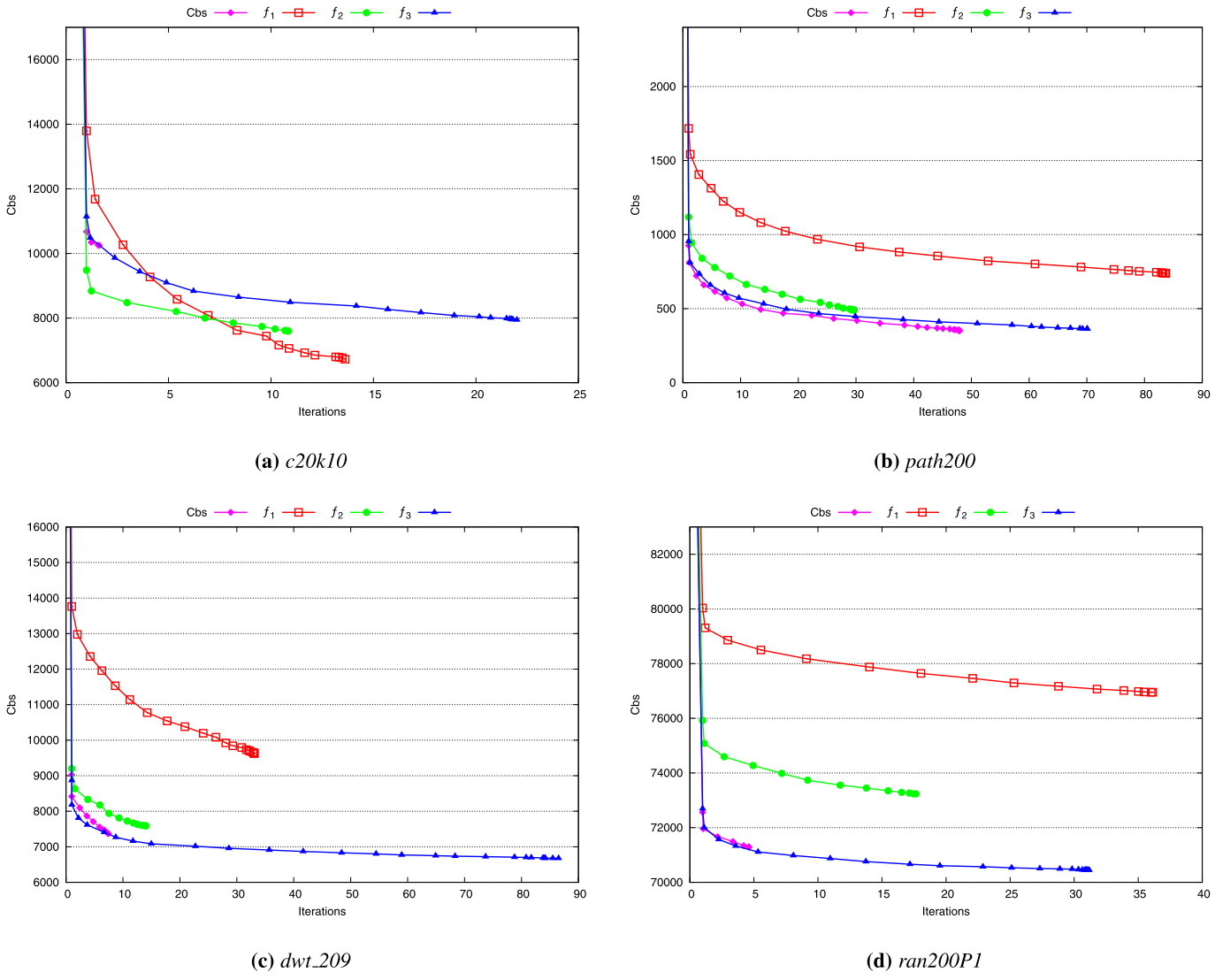


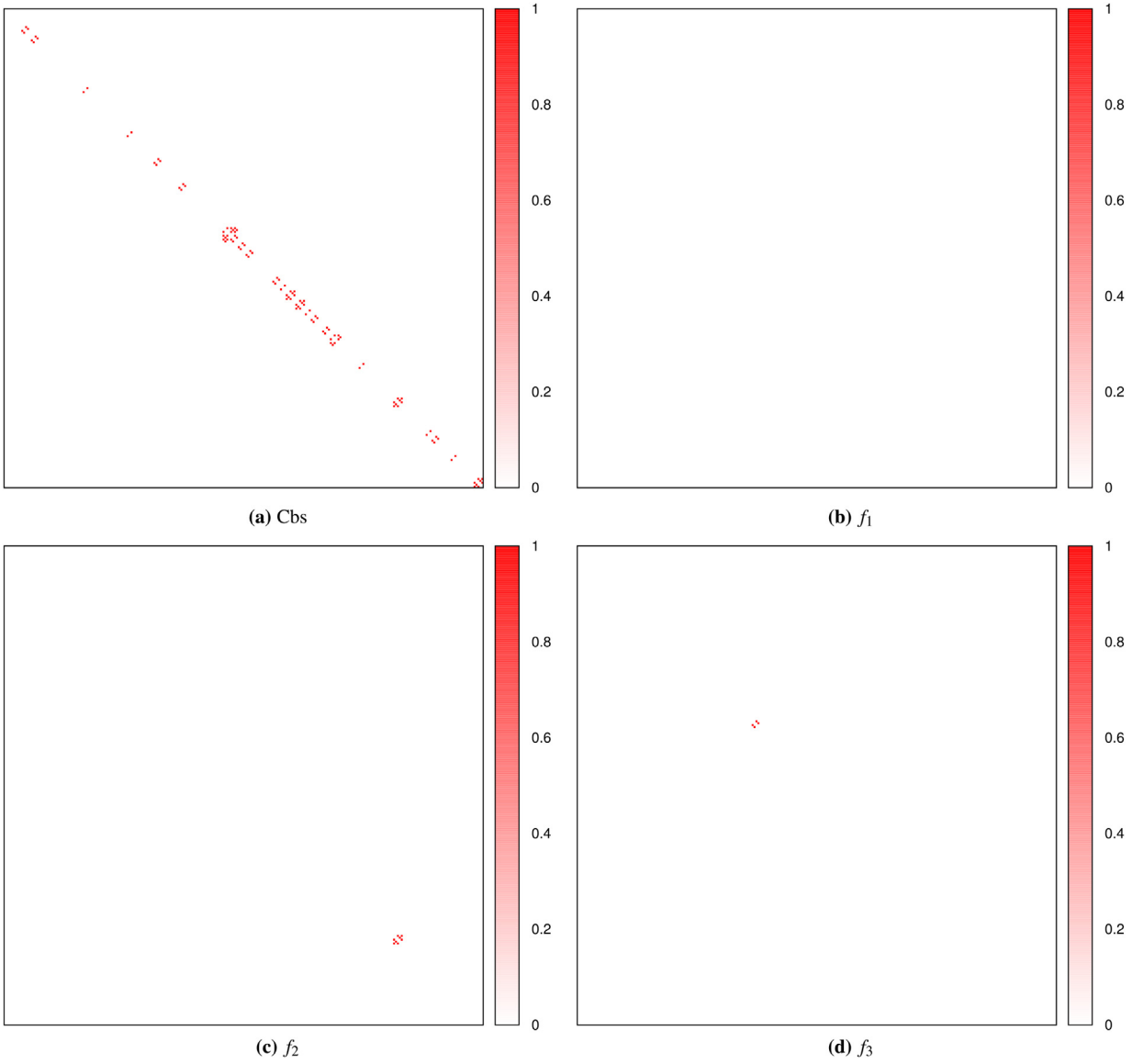
Fig. 7. Comparison of the convergence profiles of the ILS algorithm equipped with the studied evaluation functions on four representative instances.

ever, the results show that even when Cbs assigns the same cost to many neighboring solutions, breaking ties randomly within the embedded local search heuristic could be enough to guide the search out of those plateaus. Function f_3, makes a more fine-grained discrimination among potential solutions than Cbs, thus passing from one plateau (with respect to Cbs) to another one demands the execution of a most important number of local search steps to reach a similar final solution cost. This suggests that under certain circumstances it is possible that a high degree of discrimination among solutions, like that provided by f_3, may also hinder the ability of an algorithm to identify a promising search direction.

Fig. 7(c) and (d) depict the convergence results for the instances *dwt_209* and *ran200P1*. The former arises from a practical structural engineering problem, while the latter corresponds to a graph randomly generated. Both instances have non-structured topologies. These figures illustrate that using the same number of ILS iterations, f_3 reaches always a better average cyclic bandwidth sum than the other compared evaluation approaches through the search process. It is noteworthy that the performance of Cbs and f_3 is almost the same at the begin of the search. However, f_3 enables the ILS algorithm to explore the search space more efficiently, leading in general to better quality final embeddings. This is because the second term in Eq. (6) (a non-integer value) permits to better distinguish embeddings having the same cyclic bandwidth sum. Finally, the strategy used by f_3 for assigning big weight values to those cyclic distances d_k having small k values in an embedding seems to yield better final results than that used by both f_1 and f_2.

5.6. Investigating the existence of search cycles in the proposed ILS

The existence of frequent cycles during the search process can have a negative influence in the global performance of a local search algorithm. To investigate this important issue in the proposed ILS algorithm, equipped with the four studied evaluation functions, the following experiment was carried out on the four representative instances employed in the previous section: *c20k10*, *path200*, *dwt_209* and *ran200P1*. For each evaluation scheme and instance combination a total of 10,000 local search steps of the ILS algorithm (visited potential solutions) were recorded in a file. Then, the interchange distance (Cicirello & Cernera, 2013) between each pair of them was calculated to produce square distance matrices like those depicted in Fig. 8 for the instance *dwt_209*. Since very similar results were obtained for the other three tested instances, for the sake of space, only this figure is presented. In this kind of matrices a red point is depicted in coordinates (a, b) if the



(a) Cbs **(b)** f_1 **(c)** f_2 **(d)** f_3

Fig. 8. Interchange distances matrices for instance *dwt_209* using ILS and the four compared evaluation functions.

visited potential solutions at iterations a and b have an interchange distance equal to zero between them, otherwise a white point is printed at those coordinates. In this way red points in consecutive rows and columns (local search iterations) can reveal the existence of potential cycles in the search. For allowing an easy visualization of data, only the first 250 local search steps are depicted in these plots.

From Fig. 8(a) it can be observed that the conventional evaluation function Cbs produces a large number of cycles along the search. It may be explained by the fact that Cbs is unable to differentiate among distinct potential embeddings resulting in the same cost. On the contrary, Fig. 8(b)–(d) reveal that the more refined evaluation schemes proposed in this work, have the ability to strongly reduce the possibility of visiting twice the same configuration in consecutive local search iterations, thanks to their improved guiding capacities.

6. Comparing ILS with the state-of-the-art algorithms

Given the promising results obtained in the experiments reported before, we have decided to asses the performance of our ILS algorithm, equipped with the evaluation function f_3, with respect to two state-of-the-art heuristics: GVNS (Satsangi et al., 2012) and MACH (Hamon et al., 2016). The source codes of these algorithms were kindly provided by their respective authors. These codes were compiled and run independently (50 times) in our computational platform by using the input parameters suggested by their authors and setting a maximum allowed computing time of 900 seconds for each execution. The experimental results from this comparison are shown in Table 6. The benchmark instance, its order ($|V|$), size ($|E|$) and known optimal solution values (Chen & Yan, 2007; Jianxiu, 2001) are listed in the first four columns. Column 5 depicts known upper bounds. Lines 1–6 (Cartesian products of graphs) were computed as indicated by Jianxiu (2001), while for the rest of the graphs (arising from practical problems and random instances) it was done by using the expression $(m\lfloor n/2\rfloor\lceil n/2\rceil)/(n-1)$. Next, the best (*Best*), average (*Avg*) and standard deviation (*Dev*) of the cyclic bandwidth sum reached by each compared algorithm are depicted. In this table, the lowest average cyclic bandwidth sum value (*Avg*) for each of the tested instances is written in bold, and the O-RMSE value computed for each compared algorithm is presented at the bottom of the table.

Table 6
Performance comparison of the ILS algorithm, equipped with the evaluation function f_3, with respect to two state-of-the-art methods.

					GVNS			Mach			ILS		
Graph	$\|V\|$	$\|E\|$	*Opt*	*UB*	*Best*	*Avg*	*Dev*	*Best*	*Avg*	*Dev*	*Best*	*Avg*	*Dev*
c20c10	200	400		1560	6432	6467.80	24.66	2446	3754.24	1097.34	**2360**	2422.44	309.24
c20k10	200	1100		50,380	15,578	15733.00	49.57	**5300**	5300.00	0.00	**5300**	7943.28	1502.48
k20k10	200	2800		103,300	78,610	78911.00	316.92	76,614	76973.88	139.44	**62,300**	62300.00	0.00
p20c10	200	390		3790	5657	5663.48	8.30	2256	2256.00	0.00	**2238**	2335.58	268.12
p20k10	200	1090		100,190	15,092	15298.88	76.67	**5200**	5200.00	0.00	**5200**	7119.30	1228.09
p20p10	200	370		2080	5434	5441.40	5.35	4482	6271.36	703.74	**2004**	2015.12	34.83
bip100-100	200	10,000	500,000		500,014	500014.00	0.00	**500,000**	500000.00	0.00	**500,000**	500000.00	0.00
path200	200	199	199		2350	2355.20	4.16	**199**	199.00	0.00	234	363.26	51.51
cycle200	200	200	200		2036	2042.88	7.30	**200**	200.00	0.00	220	389.08	56.41
cycleP200-10	200	2000	11,000		39,210	39430.16	77.25	11,070	11217.00	72.84	**11,000**	16238.20	5297.60
wheel200	200	398	10,200		12,476	12476.00	0.00	**10,200**	10200.00	0.00	10,280	10421.64	53.25
can_229	229	774		44,505	13,842	13933.84	37.31	7764	11899.82	2358.96	**6255**	6264.80	4.23
dwt_209	209	767		40,268	13,576	13639.50	32.27	8264	10169.66	1124.26	**6371**	6677.90	328.73
steam1	240	1761		106,102	51,938	52210.00	158.50	36,713	40831.46	2373.94	**25,913**	31261.18	2842.05
ash219	219	431		23,705	10,364	10397.42	18.94	8335	8955.44	339.76	**6245**	6410.88	216.71
will199	199	660		33,000	17,613	17657.68	8.64	19,218	21205.34	788.45	**13,738**	13801.82	50.37
ran200P1	200	1991		100,050	76,487	76533.32	27.12	88,900	91052.80	1410.98	**70,037**	70444.86	230.92
ran200P3	200	5970		300,000	269,521	269526.92	29.30	294,397	294397.00	0.00	**255,192**	256094.34	467.71
ran200P5	200	9955		500,251	472,432	472466.08	22.22	502,332	504694.76	1334.77	**451,417**	452517.20	592.95
ran200P7	200	13,827		694,824	668,810	668853.12	6.22	703,730	704559.46	947.28	**649,478**	651048.42	697.50
						182.90%			34.96%			**21.01%**	
					O-RMSE								

Employing the procedure detailed in Section 4 the statistical significance of the results of this experiment was also investigated. Table 7 summarizes whether the performance differences between the studied algorithms were statistically significant or not.

Analyzing the data presented in Table 6 allowed us to make the following observations. First, our ILS algorithm is able to surpass the best solutions attained by GVNS and Mach in 14 out of 20 graphs (70.00%) and to equal the best results furnished by Mach for other 3 benchmark instances. The analysis presented in Table 7 shows that ILS achieved statistically significant better results than GVNS and Mach on 20 and 12 graphs, respectively. This highlights the suitability of the studied ILS approach which obtains the lower O-RMSE value in this experiment.

Second, Table 6 lists 15 graphs with known upper bounds (lines 1–6 and 12–20), for 14 of them our ILS algorithm equipped with evaluation function f_3 was able to improve these bounds. Our ILS algorithm is even able to furnish optimal solutions for the instances *bip100-100* and *cycleP200-10* and to supply embeddings with a small RMSE for the instances *path200*, *cycle200* and *wheel200*.

Third, one can notice that for 15 of the 20 graphs analyzed Mach found embeddings having lower cost than those provided by GVNS. We found out that graphs with a non-structured topology (i.e., random graphs and instances arising from practical problems) are difficult to solve for the Mach algorithm, given its working principle which is based on decomposing the graph in different paths. Indeed, a statistically significant performance improvement was reached by GVNS with respect to Mach on 6 benchmark instances, as can be observed in Table 7.

7. Conclusions and future work

The conventional evaluation function for the *CBS* problem provides a very poor discrimination among potential embeddings that could produce large plateaus in the fitness landscape, on which detecting a promising search direction could be hard for certain local search strategies. Three new evaluation functions for this combinatorial optimization problem have been carefully devised and introduced in this paper. All of them have the ability to create more equivalence classes with a lower cardinality, by attributing a different weight value to each cyclic distance magnitude in the graph.

Extensive comparative experiments were performed, using 20 well-known test instances, for assessing these three new evaluation approaches with respect to the conventional evaluation function for the *CBS* problem. The first experiments were aimed at analyzing the degree of discrimination that each considered evaluation function is able to provide. By means of the results produced by this analysis, it was possible to confirm the weak capacity for discrimination supplied by the conventional *CBS* evaluation function, which has been the main motivation for exploring alternative evaluation formulations. It was found that all the alternative functions are able to provide a higher capacity for discrimination among potential solutions. The functions offering the most fine-grained discrimination are f_3 and f_1, followed by f_2 and Cbs, in this order.

The *CBS*-compatibility property was defined and investigated for each of the alternative evaluation functions. This essential characteristic assesses the capacity of an alternative evaluation function to preserve the rank ordering offered by the conventional evaluation scheme among potential embeddings of the *CBS* problem. The obtained results suggest that function f_3 possesses this property. Very competitive results were also obtained by function f_2. In contrast, f_1 scored the worst *CBS*-compatibility in this experiment.

The effectiveness of the four analyzed evaluation approaches to guide the search process was assessed experimentally using a Steepest Descent (SD) algorithm. The poorest performance of the algorithm was observed when using the alternative function f_1, fol-

Table 7
Statistical analysis for comparing the performance of the ILS algorithm using evaluation function f_3 against that of the state-of-the-art methods.

Function	Instance																			Overall
	c20c10	c20k10	k20k10	p20c10	p20k10	p20p10	bip100-100	path200	cycle200	cycleP200-10	wheel200	can_229	dwt_209	steam1	ash219	will199	ran200P1	ran200P3	ran200P5	ran200P7	
GVNS / MACH	−	−	−	−	−	+	−	−	−	−	−	−	−	−	−	+	+	+	+	+	6 + 14 −
GVNS / ILS	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	0 + 20 −
MACH / ILS	−	+	−	+	+	−		+	+		+	−	−	−	−	−	−	−	−	−	6 + 12 −

lowed by that attained by the conventional evaluation approach Cbs. Function f_2 had a slightly better performance than f_1 and Cbs. In contrast, the alternative function f_3 presented a very promising behavior in most of the tested graphs.

To further explore the suitability of the studied evaluation functions, a more sophisticated metaheuristic was implemented: Iterated Local Search (ILS). The results of this experimental comparison disclosed that among the analyzed evaluation functions, f_2 exposed a promising behavior, while function f_1 presented the worst overall performance in this experiment. The results produced by the conventional evaluation scheme Cbs indicate that the neutrality of the search landscape (Pitzer & Affenzeller, 2012), induced by the low discrimination of this function, could be exploited by implementing an appropriate perturbation procedure. Finally, it was found that function f_3 helped the ILS algorithm to make a more effective search than Cbs. Considering that function f_3 was designed to work independently of other algorithmic components, it can be deployed within other advanced metaheuristics for the *CBS* problem to ameliorate their search capacity.

All the experimental evidence presented confirms the practical benefits of employing more refined evaluations schemes as a means of improving the search capacities of the implemented metaheuristic algorithms for the *CBS* problem. In particular our ILS algorithm, employing f_3 as evaluation function, was able to outperform the best solutions provided by the state-of-the-art algorithms GVNS and MACH in 14 out of 20 benchmark instances. Indeed, this algorithm was able to contribute to the state-of-the-art of the *CBS* problem by reaching optimal solutions for 2 instances in the benchmark set and the establishment of 14 new upper bounds for other graphs.

Although very promising average results were obtained by using f_3 as evaluation function within our ILS implementation, we observed that for some graph topologies certain alternative evaluation schemes provided better final embeddings (see Section 5.5). For this reason, our future work will concentrate on: (1) identifying essential properties (besides the degree of discrimination and the *CBS*-compatibility) of the studied evaluation functions that permit to better explain why one function works better than others on certain graph types, and (2) designing an adaptive mechanism for combining the best guiding properties of different evaluation functions through the search process for producing even better quality solutions at a reasonable computational effort.

Acknowledgments

The first author thankfully acknowledge a sabbatical leave granted by CINVESTAV (01/09/2016–31/08/2017), the financial aid from CONACYT Mexico through the grant *Estancias Sabáticas en el Extranjero 2016 – No. 454954*, as well as the courtesies and facilities of the LERIA at the University of Angers, France. The research of the third author has been partially supported by the Spanish Ministry of "Economía y Competitividad", grant ref. TIN2015-65460-C2-2-P.

References

Avci, M., & Topaloglu, S. (2017). A multi-start iterated local search algorithm for the generalized quadratic multiple knapsack problem. *Computers & Operations Research, 83*, 54–65. doi:10.1016/j.cor.2017.02.004.

Benlic, U., Epitropakis, M. G., & Burke, E. K. (2017). A hybrid breakout local search and reinforcement learning approach to the vertex separator problem. *European Journal of Operational Research, 261*(3), 803–818. doi:10.1016/j.ejor.2017.01.023.

Bhatt, S. N., & Leighton, F. T. (1984). A framework for solving VLSI graph layout problems. *Journal of Computer and System Sciences, 28*(2), 300–343. doi:10.1016/0022-0000(84)90071-0.

Blum, C., & Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys, 35*(3), 268–308. doi:10.1145/937503.937505.

Chen, Y., & Yan, J. (2007). A study on cyclic bandwidth sum. *Journal of Combinatorial Optimization, 4*(2–3), 295–308.

Cicirello, V. A., & Cernera, R. (2013). Profiling the distance characteristics of mutation operators for permutation-based genetic algorithms. In *Proceedings of the twenty-sixth international Florida Artificial Intelligence Research Society conference* (pp. 46–51.). St. Pete Beach, FL, USA: AAAI.

Coelho, V., Grasas, A., Ramalhinho, H., Coelho, I., Souza, M., & Cruz, R. (2016). An ILS-based algorithm to solve a large-scale real heterogeneous fleet VRP with multi-trips and docking constraints. *European Journal of Operational Research, 250*(2), 367–376. doi:10.1016/j.ejor.2015.09.047.

Corne, D. W., & Knowles, J. D. (2007). Techniques for Highly Multiobjective Optimisation: Some Nondominated Points are Better than Others. In *Proceedings of the ninth genetic and evolutionary computation conference: 1* (pp. 773–780). London, UK: ACM Press. doi:10.1145/1276958.1277115.

Cruz, F., Subramanian, A., Bruck, B. P., & Iori, M. (2017). A heuristic algorithm for a single vehicle static bike sharing rebalancing problem. *Computers & Operations Research, 79*, 19–33. doi:10.1016/j.cor.2016.09.025.

Derbel, B., Humeau, J., Liefooghe, A., & Verel, S. (2014). Distributed localized bi-objective search. *European Journal of Operational Research, 239*(3), 731–743. doi:10.1016/j.ejor.2014.05.040.

Garza-Fabre, M., Rodriguez-Tello, E., & Toscano-Pulido, G. (2013). Comparative analysis of different evaluation functions for protein structure prediction under the hp model. *Journal of Computer Science and Technology, 28*(5), 868–889. doi:10.1007/s11390-013-1384-7.

Garza-Fabre, M., Toscano-Pulido, G., & Rodriguez-Tello, E. (2015). Multi-objectivization, fitness landscape transformation and search performance: A case of study on the hp model for protein structure prediction. *European Journal of Operational Research, 243*(2), 405–422. doi:10.1016/j.ejor.2014.06.009.

Gendreau, M., & Potvin, J. Y. (Eds.). (2010). *Handbook of metaheuristics volume 146 of International Series in Operations Research & Management Science*. Berlin, Heidelberg: Springer. doi:10.1007/978-1-4419-1665-5.

Godim da Fonseca, G. H., Gambini Santos, H., Machado Toffolo, T., Souza Brito, S., & Freitas Souza, M. J. (2016). GOAL solver: a hybrid local search based solver for high school timetabling. *Annals of Operations Research, 239*(1), 77–97. doi:10.1007/s10479-014-1685-4.

Hagberg, A., Schult, D., & Swart, P. (2016). Networkx - high-productivity software for complex networks. https://networkx.github.io/.

Hagberg, A. A., Schult, D. A., & Swart, P. J. (2008). Exploring network structure, dynamics, and function using networkx. In G. Varoquaux, T. Vaught, & J. Millman (Eds.), *Proceedings of the seventh python in science conference* (pp. 11–15). Pasadena, CA USA.

Hamon, R., Borgnat, P., Flandrin, P., & Robardet, C. (2016). Relabelling vertices according to the network structure by minimizing the cyclic bandwidth sum. *Journal of Complex Networks, 4*(4), 534–560. doi:10.1093/comnet/cnw006.

Harper, L. (1964). Optimal assignment of numbers to vertices. *Journal of SIAM, 12*(1), 131–135.

Jaccard, P. (1912). The distribution of the flora in the alpine zone. *New Phytologist, 11*(2), 37–50. doi:10.1111/j.1469-8137.1912.tb05611.x.

Jianxiu, H. (2001). Cyclic bandwidth sum of graphs. *Applied Mathematics – A Journal of Chinese Universities, 16*(2), 115–121. doi:10.1007/s11766-001-0016-0.

Karapetyan, D., Mitrovic Minic, S., Malladi, K. T., & Punnen, A. P. (2015). Satellite downlink scheduling problem: a case study. *Omega, 53*, 115–123. doi:10.1016/j.omega.2015.01.001.

Liberatore, V. (2002). Multicast scheduling for list requests. In *Proceedings of the twenty-first annual joint conference of the IEEE computer and communications societies: 2* (pp. 1129–1137). IEEE. doi:10.1109/INFCOM.2002.1019361.

Lochtefeld, D. F., & Ciarallo, F. W. (2015). Multi-objectivization via decomposition: An analysis of helper-objectives and complete decomposition. *European Journal of Operational Research, 243*(2), 395–404. doi:10.1016/j.ejor.2014.11.041.

Lourenço, H. R., Martin, O., & Stützle, T. (2003). *Iterated local search*. In F. Glover, & G. A. Kochenberger (Eds.) (pp. 320–353)). Boston, MA, USA: Springer. doi:10.1007/b101874.

Lourenço, H. R., Martin, O. C., & Stützle, T. (2010). *Iterated local search: framework and applications*. In M. Gendreau, & J. Potvin (Eds.) (pp. 363–397)). Boston, MA, USA: Springer. doi:10.1007/978-1-4419-1665-5_12.

Marmion, M., Dhaenens, C., Jourdan, L., Liefooghe, A., & Verel, S. (2011). On the neutrality of flowshop scheduling fitness landscapes. In C. A. Coello Coello (Ed.), Learning and Intelligent Optimization: 5th International Conference, LION 5, Rome, Italy, January 17–21, 2011. Selected Papers (pp. 238–252). Berlin, Heidelberg: Springer Berlin Heidelberg volume 6683 of Lecture Notes in Computer Science. doi:10.1007/978-3-642-25566-3_18.

Martin, O., Otto, S., & Felten, E. (1991). Large-step Markov chains for the traveling salesman problem. *Complex Systems, 5*(3), 299–326. doi:10.1016/0167-6377(92)90028-2.

Michiels, W., Aarts, E., & Korst, J. (2007). *Theoretical aspects of local search*. Springer. doi:10.1007/978-3-540-35854-1.

Mladenovic, N., & Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research, 24*, 1097–1100. doi:10.1016/S0305-0548(97)00031-2.

Monien, B., & Sudborough, I. H. (1990). Embedding one interconnection network in another. *Computational Graph Theory, Computing Supplementum, 7*, 257–282.

Murovec, B. (2015). Job-shop local-search move evaluation without direct consideration of the criterion's value. *European Journal of Operational Research, 241*(2), 320–329. doi:10.1016/j.ejor.2014.08.044.

Pitzer, E., & Affenzeller, M. (2012). A comprehensive survey on fitness landscape analysis. In J. Fodor, R. Klempous, & C. P. Suárez-Araujo (Eds.), *Recent advances in intelligent engineering systems*. In *Studies in Computational Intelligence: 378* (pp. 161–191). Springer.

Porumbel, D., Goncalves, G., Allaoui, H., & Hsu, T. (2017). Iterated local search and column generation to solve arc-routing as a permutation set-covering problem. *European Journal of Operational Research, 256*(2), 349–367. doi:10.1016/j.ejor.2016.06.055.

Rodriguez-Tello, E., Hao, J. K., & Romero-Monsivais, H. (2015). Boosting the performance of metaheuristics for the minla problem using a more discriminating evaluation function. *Tehnicki Vjesnik – Technical Gazette, 22*(1), 11–24. doi:10.17559/TV-20130905130612.

Rodriguez-Tello, E., Hao, J. K., & Torres-Jimenez, J. (2008a). An effective two-stage simulated annealing algorithm for the minimum linear arrangement problem. *Computers & Operations Research, 35*(10), 3331–3346. doi:10.1016/j.cor.2007.03.001.

Rodriguez-Tello, E., Hao, J. K., & Torres-Jimenez, J. (2008b). An improved simulated annealing algorithm for bandwidth minimization. *European Journal of Operational Research, 185*(3), 1319–1335. doi:10.1016/j.ejor.2005.12.052.

Satsangi, D., Srivastava, K., & Gursaran (2012). General variable neighbourhood search for cyclic bandwidth sum minimization problem. In *Proceedings of the students conference on engineering and systems* (pp. 1–6). IEEE Press. doi:10.1109/SCES.2012.6199079.

Silva, M. M., Subramanian, A., & Ochi, L. S. (2015). An iterated local search heuristic for the split delivery vehicle routing problem. *Computers & Operations Research, 53*, 234–249. doi:10.1016/j.cor.2014.08.005.

Silva Paiva, G., & Carvalho, M. (2017). Improved heuristic algorithms for the job sequencing and tool switching problem. *Computers & Operations Research, 88*, 208–219. doi:10.1016/j.cor.2017.07.013.

Smet, P., Bilgin, B., De Causmaecker, P., & Vanden Berghe, G. (2014). Modelling and evaluation issues in nurse rostering. *Annals of Operations Research, 218*(1), 303–326. doi:10.1007/s10479-012-1116-3.

Stadler, P. F. (1992). Correlation in landscapes of combinatorial optimization problems. *Europhysics Letters, 20*, 479–482.

Talbi, E. (2009). *Metaheuristics: From design to implementation*. John Wiley & Sons.

Ullman, J. D. (1984). *Computational Aspects of VLSI*. Computer Science Press.

Umetani, S. (2017). Exploiting variable associations to configure efficient local search algorithms in large-scale binary integer programs. *European Journal of Operational Research, 263*(1), 72–81. doi:10.1016/j.ejor.2017.05.025.

Yuan, J. (1995). Cyclic arrangement of graphs. *Graph Theory Notes of New York, New York Academy of Sciences*, 6–10.