IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received September 5, 2021, accepted October 30, 2021, date of publication November 8, 2021,
date of current version November 15, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3126015

Search Trajectory Networks Applied to the Cyclic Bandwidth Sum Problem

VALENTINA NARVAEZ-TERAN[1], GABRIELA OCHOA[2], AND EDUARDO RODRIGUEZ-TELLO[1], (Member, IEEE)

[1]Cinvestav Tamaulipas, Victoria, Tamps 87130, Mexico
[2]Computing Science and Mathematics Department, University of Stirling, Stirling FK9 4LA, U.K.

Corresponding author: Eduardo Rodriguez-Tello (ertello@cinvestav.mx)

This work was supported in part by the Mexican Secretariat of Public Education with Grant SEP-Cinvestav No. 00114 (2019–2021).

ABSTRACT Search trajectory networks (STNs) were proposed as a tool to analyze the behavior of metaheuristics in relation to their exploration ability and the search space regions they traverse. The technique derives from the study of fitness landscapes using local optima networks (LONs). STNs are related to LONs in that both are built as graphs, modelling the transitions among solutions or group of solutions in the search space. The key difference is that STN nodes can represent solutions or groups of solutions that are not necessarily locally optimal. This work presents an STN-based study for a particular combinatorial optimization problem, the cyclic bandwidth sum minimization. STNs were employed to analyze the two leading algorithms for this problem: a memetic algorithm and a hyperheuristic memetic algorithm. We also propose a novel grouping method for STNs that can be generally applied to both continuous and combinatorial spaces.

INDEX TERMS Search trajectory networks, cyclic bandwidth sum problem, hyperheuristics, memetic algorithms, hybridization.

I. INTRODUCTION

Observing the inner dynamics of metaheuristics is crucial for a better understanding of how these algorithms differentiate from each other on the way they explore the search space, and how those differences relate to their performance under specific scenarios. This is increasing in relevance, as in recent years numerous novel metaheuristics have been proposed. The metaphorical formulation of these algorithms often takes inspiration from processes found in natural systems, physics [1]–[3], social behaviors of species like ants [4], bees [5], fireflies [6], chaos and game theory [7]–[9], etc. Furthermore, there is a tendency for metaheuristic hybridization [10]–[12] and hyperheuristics [13], [14]. Very promising results have been obtained by such proposals, causing a growing interest in extending their applications and also increasing the need for analysis tools to effectively characterize their behavior.

On this matter, search trajectory networks (STNs) [15], [16] are a relatively novel analysis tool for gaining a better perspective on the inner search dynamics of metaheuristics in relation to how they explore the search space of a particular problem instance. STNs allow to identify how the success of a metaheuristic relates to the search process being conducted through specific search regions. Therefore, STNs constitute a metaphor-free attempt at profiling metaheuristics.

Both local optima networks (LONs) [15], [17] and STNs represent, in the form of directed graphs, key structural features of the search space and how the algorithms navigate them. Both models are built as networks and analyzed using network metrics to identify the paths conducting towards the best search space regions, which areas are hard to escape from, and how they are related. The key difference is that a search trajectory network is created from trajectories between solutions that are not necessarily locally optimal. In that way, STNs overcome some limitations of LONs, opening their applicability to contrast the behavior of a wider variety of metaheuristics, such as population metaheuristics that do not necessarily perform local search.

STNs were initially proposed to characterize differential evolution and particle swarm optimization [18] for several classical continuous optimization benchmark functions [19]–[21]. In this scenario the continuous search space was partitioned into regular size hypercubes. STN analysis was

The associate editor coordinating the review of this manuscript and approving it for publication was Donghyun Kim.

later extended to cover not only population-based algorithms but also stochastic local search methods, and both continuous and combinatorial optimization problems [16]. However, representative standard metaheuristics, as opposed to state-of-the-art methods, were considered. As the combinatorial case study, a classic facility location problem [22] (p-median) encoded with binary strings was used. In the continuous domain, search can be partitioned into discrete hypercubes of a predefined length, thus defining the STN locations. This cannot be done for discrete spaces. For discrete spaces, locations can be associated with individual solutions. However, it is still desirable to coarsen the search space in order to appreciate the trajectories at different levels of granularity. Therefore, the authors proposed a partitioning scheme based on the sampled solutions [16]. The idea was to group as a single location all solutions sharing the same value in some variables while ignoring (removing) other variables. This method worked well for a binary representation, but it is not suitable for a permutation encoding.

In this work we report a STN-based analysis of a combinatorial optimization problem, the cyclic bandwidth sum problem (CBSP) [23]. Our goal is to gain a new perspective on how the two state-of-the-art metaheuristics proposed for the CBSP behave, why are they successful, why some type of instances seem more challenging, as well as obtaining new insights of the underlying fitness landscapes. These algorithms are interesting subjects for STN analysis because their design features differ from the classical memetic algorithm they were originally based on. The first algorithm is a memetic algorithm (MA) that employs a non-intensive local search approach [24], the second one is a hyper-heuristic approach [25] using a dynamic multi-armed bandit (DMAB) [26] to automate the selection of the genetic operators and the fitness function [27] within a MA.

Another contribution of this work is the proposal of a search space partitioning method based on reducing the solutions dimensions while preserving their distances using multidimensional scaling [28]–[30]. This approach to search space partitioning is more general than those previously used for STN analysis [15], [16], as it can be applied to any form of solution encoding (discrete and continuous) as soon as a suitable distance metric between solutions can be devised.

The remaining sections are organized as follows. Section II provides a brief introduction to graph embedding problems and in particular to the CBSP. Section III describes the methodology for the creation and analysis of STNs. Section IV introduces the two studied algorithms. The experimental results are presented and discussed in Section V. Finally, Section VI presents our conclusions.

II. THE CYCLIC BANDWIDTH SUM PROBLEM

Graph embedding problems (GEPs) [31] are a family of combinatorial optimization problems focused on the rearrangement of graphs to fit a certain new layout. These type of problems are commonly defined in terms of a guest graph and a host graph. The guest graph describes the original relationships among whichever objects the graph represents, for example, a virtual computer network or a set of related facilities. Meanwhile, the host graph describes the new layout to rearrange the guest graph, for example, a network infrastructure architecture, or a set of connected physical locations for facilities. Therefore, graph embeddings are equivalent to mappings that assign guest nodes to host nodes, and guest edges to paths in the host. The embeddings are often represented as labelings [32], assigning to each guest node a host node represented by using its label.

GEPs arise in areas such as VLSI design [33], automatic graph drawing [34], [35], codes for error detection and correction [36], network virtualization [37], parallel virtual computing [38], scheduling [39], [40] or facility allocation [39].

Commonly, a GEP's objective is to find an embedding that optimizes a certain measure, defined with respect to the topological structure of the host graph. The generalized classification of GEPs [32] includes three groups characterized by the nature of what are they trying to optimize: a) the distance among guest nodes when embedded in host nodes, b) the sum of distances among guest nodes when embedded in host nodes, and c) the cutting of host edges embedded in host paths.

One of the best-known and widely studied GEP is the bandwidth problem [41], consisting in rearranging a graph into a linear layout, i.e, embedding a guest graph of order n into the path graph P_n in such a way that the linear distance among adjacent guest nodes, i.e., the bandwidth, is minimized. The bandwidth problem belongs to the first group mentioned above. The bandwidth sum problem [42]–[44] is then the GEP consisting on minimizing the sum of the linear distances. Now, consider how a cut on a particular edge of the host graph will disrupt the embedded guest edges that cross through it. The cutwidth problem [32], [45], [46] consists in minimizing the number of disrupted guest edges affected by a cut on any host edge.

Particular GEPs are also defined by the topology of the host graph. The examples mentioned above involve only linear topologies in the form of path graphs as hosts. When the guest graph topology is instead circular, denoted by the cycle graph, the arising problems are the cyclic bandwidth [47], the cyclic bandwidth sum [48], and the cyclic cutwidth [49], respectively.

This paper deals with the cyclic bandwidth sum problem (CBSP) [48], where the objective is to find the optimal way to embed a graph into a circular layout while minimizing the sum of cyclic distances between adjacent guest nodes.

Formally, the CBSP is defined as follows. Let $G = (V, E)$ be a finite undirected graph (the guest) of order $n = |V|$ and C_n a cycle graph (the 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 into 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 $\varphi(u)$. The CBSP consists in finding the optimal embedding φ^*, such that $\text{Cbs}(G, \varphi^*)$ is minimum, i.e., $\varphi^* = \arg\, min_{\varphi \in \Phi}\{\text{Cbs}(G, \varphi)\}$ with Φ denoting the set of all the potential embeddings.

The value of the optimum can be calculated in an exact way for the following graph topologies: *path*, *cycle*, *complete bipartite graph*, *wheel*, and *power of cycles* [23]. Approximations of the optimal value were reported for Cartesian products of two graphs (*paths, cycles,* and *complete graphs*) [50]. Regarding heuristic solution proposals, the problem has been approached by general variable neighborhood search [51], a constructive greedy heuristic [52], as well as the MA [24], and DMAB+MA [25] algorithms analyzed in this work.

III. BUILDING SEARCH TRAJECTORY NETWORKS

Given a metaheuristic algorithm A, an STN is built as a weighted, directed graph $STN_A(V_A, E_A)$ where:

- V_A is the set of search space **locations** visited by the algorithm.
- E_A is the set of directed edges, such that two nodes $a, b \in V_A$ are adjacent if the algorithm A performed a transition between solutions in the respective locations. The weight of the edge represents the number of times the transition from a to b occurred during the search.

A **location** is a search space region that contains at least one **representative solution**. Each representative solution was the current best solution during an iteration of the algorithm, corresponding to the best solution in the population for multi-trajectory metaheuristics or the incumbent solution for single-trajectory metaheuristics.

In order to compare the search dynamics of two or more metaheuristics, the STN models for two or more metaheuristics, under the same problem instance can be merged. This idea of merging the trajectories of two algorithms in a single network model was first proposed and implemented for local optima networks in [53], and later adapted to STNs [15], [16].

More formally, the merged STN for two metaheuristic algorithms A_i and A_j is the union of the STN graphs STN_{A_i} and STN_{A_j} as $STN_{A_i,A_j} = (V_{A_i} \cup V_{A_j}, E_{A_i} \cup E_{A_j})$. The weight of each edge in the merged STN is equal to the sum of the number of times algorithms A_i and A_j performed the transition represented by the edge.

A. SAMPLING, MODELING AND PARTITIONING

The first step for creating an STN is to record an aggregated sample of the transitions between representative solutions that occurred during independent executions of the studied metaheuristic for a problem instance. A parameter is employed to control the frequency of recording those transitions. This process is quite simple to implement, it has no effect on the metaheuristic search process, and it does not represent a significant overhead.

The STN modeling is the process of turning the sampled trajectories into a network of locations, under the previous definition of nodes, edges and weights. An important part of this process is to establishing a mapping between single solutions and locations. By definition, a location can be a subregion as small as containing only one single solution. However, the STN can result more informative if locations represent instead subregions of the search space. The definition of which solutions are mapped to the same location can vary. In general, it should be linked to a notion of closeness among solutions. For example, for continuous optimization problems, a partitioning of the search space into hypercubes of regular size has been induced by adjusting the precision of the solution encoding [15].

Since the quality of each location is represented by the fitness of the best solution that belongs to it, it is implied that, if the metaheuristic can reach a solution within the corresponding subregion, it is likely that it could as well reach the best solution from it. Grouping solutions into STN locations also provides a globalized perspective on the overall search dynamics. For example, by making it possible to identify if an algorithm is recurrently visiting some specific regions, and to detect regions that act like hubs, even if the algorithm does not revisit the exact same solutions.

The partitioning method employed in this work is based on the classical multidimensional scaling [28], [29]. The idea is to reduce a set of solutions from a representation in n dimensions to the Euclidean space, while minimizing the distortion of the pair-wise distances among them. The solutions we dealt with in both algorithms were encoded as permutations of n elements. Under the problem definition, they are circular permutations. Therefore, we employed the interchange distance [54] to evaluate the pair-wise distances between unique solutions in the sample. The interchange distance measures the number of cycles between two permutations, therefore it is suitable to reflect the swap-based edition distance for CBSP potential solutions. Then, each solution was mapped to a 3-tuple (three dimensional coordinates) by using classical multidimensional scaling. At that moment, the obtained coordinate values are rounded up to integers for grouping solutions sharing the same resulting integer 3-tuples. Multidimensional scaling has been previously employed in the creation of visual representations of the distribution of solutions to analyze the cartography of search spaces [30], [55].

B. ANALYZING STNs

The analysis of STNs has two complementary components: a set of network metrics and a visual representation. The metrics capture relevant features of the merged networks and their structure, while the visual representation provides a better idea of how the trajectories intersect and allows to observe relevant locations.

The set of network metrics evaluated in this work are the following:

- **nodes**: Number of nodes
- **edges**: Number of edges
- **end nodes**: The number of unique nodes without any edge going out from them. Each end node corresponds to the end of a trajectory.
- **best nodes**: The number of unique nodes with a fitness equal to the best found during the trajectory.
- **in-strength**: Evaluated for the best nodes and end nodes, it is the ratio between the sum of their weighted incoming degree and the sum of weighed incoming degrees for all end nodes.
- **shared nodes**: Number of nodes visited by more than one algorithm.
- **visited nodes (v. nodes)**: Number of nodes visited by algorithm A_i.
- **shortest path length (s. path)**: Length of the shortest path to best node visited by algorithm A_i.
- **number of shortest paths (n. paths)**: Number of shortest paths to the closest best node visited by algorithm A_i.

The STNs were plotted in R by employing the *igraph* package [56] and the force-directed layout algorithms Kamada-Kawai [57] and Fruchterman-Reingold [58]. Both layout algorithms try to create displays of the network that have:

- a roughly even distribution of vertices
- minimized crossings of edges
- approximately uniform length edges
- preservation of inherent symmetries, in such a way that similar subnetworks are depicted similarly as well.

IV. THE HYBRID ALGORITHMS

The algorithms considered in this study are a memetic algorithm (MA) [24], and DMAB+MA [25], a technique that employs a dynamic multi-armed bandit [26] as a hyperheuristic approach to adaptive operator selection (AOS) [59] within a MA. DMAB+MA can be considered also as an adaptive memetic algorithm. Both are hybrid population-based algorithms, and to the best of our knowledge, they are the top state-of-the-art methods regarding solution quality for the CBSP. These algorithms are the result of prior research on the interplay of different configurations of genetic operators within a MA, extending to the selection, crossover, fitness function, mutation and survival strategy, and how the strength of individual configurations can be combined in an AOS approach.

A. MEMETIC ALGORITHM

The MA structure is described in Algorithm 1. The MA begins with a population P of μ permutation encoded randomly generated parents. Throughout the generations, pairs of parents are selected using binary tournament. Each pair of parents produces one offspring individual o with probability p_c, by using cyclic crossover. Alternatively, with probability $1 - p_c$, individual o is instead a copy of the fittest parent in the pair. Individual o is mutated with probability p_m. The mutation operator is a fitness oriented reduced 3-swap that works as follows. Three random distinct genes are chosen, then the impact over the fitness of the six different ways on which those genes can be swapped in pairs are calculated. The mutated individual o' results from performing the swap that result on the best fitness improvement, or the one that deteriorate the fitness the least (if there is not an improving swap). After this, an inversion operator is applied with an independent probability p_i, producing a further mutated individual o''.

Algorithm 1 Memetic Algorithm

```
input Guest graph G
output Best-found solution g
P ← initializePopulation(P)
O ← ∅
g ← P_best
repeat
    for j ← 1 to μ do
        P_a, P_b ← selection(P)
        o ← crossover(P_a, P_b, p_c)
        o' ← mutation(o, p_m)
        o'' ← inversion(o', p_i)
        O ← O ∪ o''
        g ← fittest individual among g, o, o', and o''
    end for
    P ← survival(P, O)
    O ← ∅
    P_best ← localsearch(P_best, ls)
    g ← fitter individual among current g and P_best
until stop criterion is met
return g
```

Together, the reduced 3-swap and insertion operators can be seen as a two phase mutation in which each phase is independent. The first phase is oriented to seek good fitness mutations, while the second has a random nature. While only individual o'' joins the offspring population O, possible improvements on the best-so-far solution achieved by individuals o and o' are still recorded. This MA uses the (μ, λ) type survival, directly replacing the parent population P by the offspring population O.

The local search phase occurs after the survival. Under this MA approach, the local search is non-intensive, in the sense that a) it is not applied to each individual in the population, but rather only to the fittest individual in it, and b) it is not carried on until a local optimum is reached, but instead only for a reduced number ls of iterations. These particularities are intended to prevent the population from becoming rapidly overtaken by locally optimal individuals, whose genes could proliferate excessively and cause premature convergence. Notice that the local search can resume its process where it was left if the best individual in the population remains the same for more than one generation (i.e., $P_{best,t} = P_{best,t+1}$), but also it can be restarted from similar fitness value individuals (i.e., $f(P_{best,t}) = f(P_{best,t+1})$ and $P_{best,t} \neq P_{best,t+1}$).

B. DMAB+MA

The multi-armed bandit (MAB) [60]–[62], is an effective way to approach the problem of choosing *the best* among k alternatives. The MAB is traditionally modeled as k casino bandit machines, each with an underlying unknown reward probability of yielding a reward, where the objective is to pick whose machine's arm to play in order to maximize the accumulated reward over time. The dynamic multi-armed bandit (DMAB) [26] is the version of the MAB for dynamic scenarios, where the arm's reward probabilities can drift.

In the DMAB+MA, described in Algorithm 2, the DMAB was implemented as an adaptive operator selection approach to determine the best configuration of operators for a MA, that otherwise, follows the structure previously described in Section IV-A. The pool of MA operators [25] consists of four selection operators (stochastic, roulette, random and binary tournament), two operators for crossover (cyclic and order-based), three mutation operators (insertion, reduced 3-swap and cumulative swap) and two survival strategies ((μ, λ) and ($\mu + \lambda$)). Additionally, there are two fitness evaluation functions: the conventional function depicted in (1) and an alternative fitness function f', designed to deal with the intrinsic high neutrality of the CBSP which was previously reported in [27]. Each operator configuration is equivalent to a DMAB arm a_i.

The DMAB+MA has been proven to be very effective, when compared with the MA and other existing reference algorithms from the CBSP literature [25]. The DMAB begins by playing all arms and assigning them initial reward values. It employs the upper confidence bound (UCB1) [62] to assign to each arm a_i a confidence estimation based on its historical rewards and the number of times they have been used. This allows to balance the choice between arms that produced good results and those that have been underused. At each iteration the arm with the highest confidence is played and a reward on function of the fitness improvement it was able to achieve is assigned to it. The Page-Hinkley test (PH-Test) [63] is implemented to help the DMAB algorithms to adjust to scenarios were the most suitable arm to play can change over the time. When this occurs, the PH-Test triggers and all the existing arms are restarted.

Algorithm 2 DMAB Algorithm

```
input Guest graph G
output Best-found solution g
P ← initializePopulation(P)
P_best ← best individual in P
g ← P_best
Set confidence and number of times arms have been played to zero
for i ← 1 to k do
    P' ← playArm(a_i, P)
    Assign initial reward to a_i
end for
repeat
    Compute confidence for all arms
    a_s ← selectArm()
    P ← playArm(a_s, P)
    Update a_s reward and increase the number of times it has been played
    if PH-test is triggered then
        Set confidence and number of times arms have been played to zero
        for i ← 1 to k do
            P' ← playArm(a_i, P)
            Assign initial reward to a_i
        end for
    end if
    P_best ← best individual in P
    g ← fitter individual among current g and P_best
until stop criterion is met
return g
```

V. EXPERIMENTAL RESULTS

A. EXPERIMENTAL SETTINGS

A set of 23 well-studied CBSP instances, listed in Table 2, was selected. The set contains topologically diverse, and representative graphs of order $24 \leq n \leq 200$, with a number of edges ranging from 68 to 2000. The topologies included in this set are: 2 *paths*, 2 *cycles*, 2 *wheels*, 6 *Cartesian products*, 4 *power of cycles*, and 7 *Harwell-Boeing* graphs.

For constructing the STNs, we record the transitions between representative solutions that occur when the MA and DMAB+MA algorithms, described in Section IV, are used for solving the selected CBSP instances. To this end, MA and DMAB+MA were coded in C language and compiled in gcc 4.4.7 using an Intel Xeon CPU X5550 at 2.67 GHz with 16 GB in RAM. Ten independent executions, over every instance, were sampled at a $5 \cdot |V|$ frequency to produce the corresponding STNs. The two analyzed algorithms stop either after completing 50,000 generations or when the instance optimal/best-known solution value reported in the literature is attained [25]. Table 1 depicts the rest of the algorithms configuration parameter values, which were established by using the *irace* R package for automatic tuning [64].

TABLE 1. Parameter settings for the MA and DMAB+MA algorithms.

Parameter	MA	DMAB+MA
Population size μ	20	20
Crossover probability p_c	0.788	0.812
Mutation probability p_m	0.543	0.761
Inversion probability p_i	0.240	0.012
Local search iterations ls	10	10
Scaling factor C	–	7.138
Length of the reward register W	–	1
PH-test tolerance δ	–	0.299
PH-test triggering threshold λ	–	33.472

TABLE 2. Size and order of graphs in the instance set.

Problem instance	$\|V\|$	$\|E\|$
path100	100	99
path200	200	199
cycle100	100	100
cycle200	200	200
wheel100	100	198
wheel200	200	398
cyclePow100-2	100	200
cyclePow100-10	100	1000
cyclePow200-2	200	400
cyclePow200-10	200	2000
p9p9	81	144
c9c9	81	162
k9k9	81	648
p9c9	81	153
p9k9	81	396
c9k9	81	405
can24	24	68
ibm32	32	90
curtis54	54	124
will57	57	127
ash85	85	219
nos4	100	247
can144	144	576

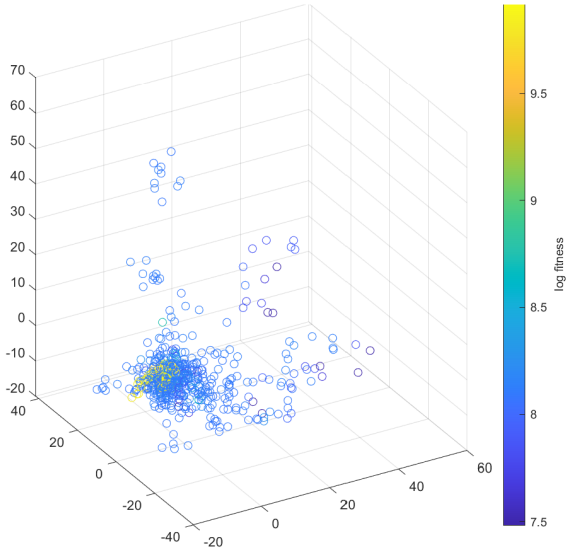


FIGURE 1. Sampled solutions obtained by executing the MA and DMAB+MA algorithms over the instance *can144*, and mapped to the Euclidean 3D space by applying multidimensional scaling. Solutions with small (better) fitness values are represented with dark violet points in the plot.

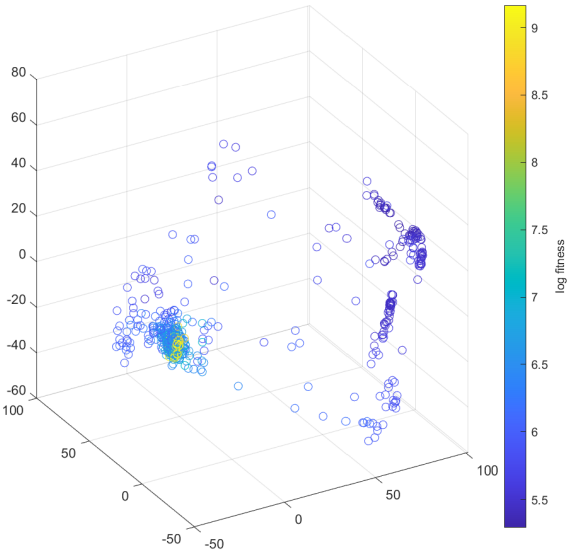


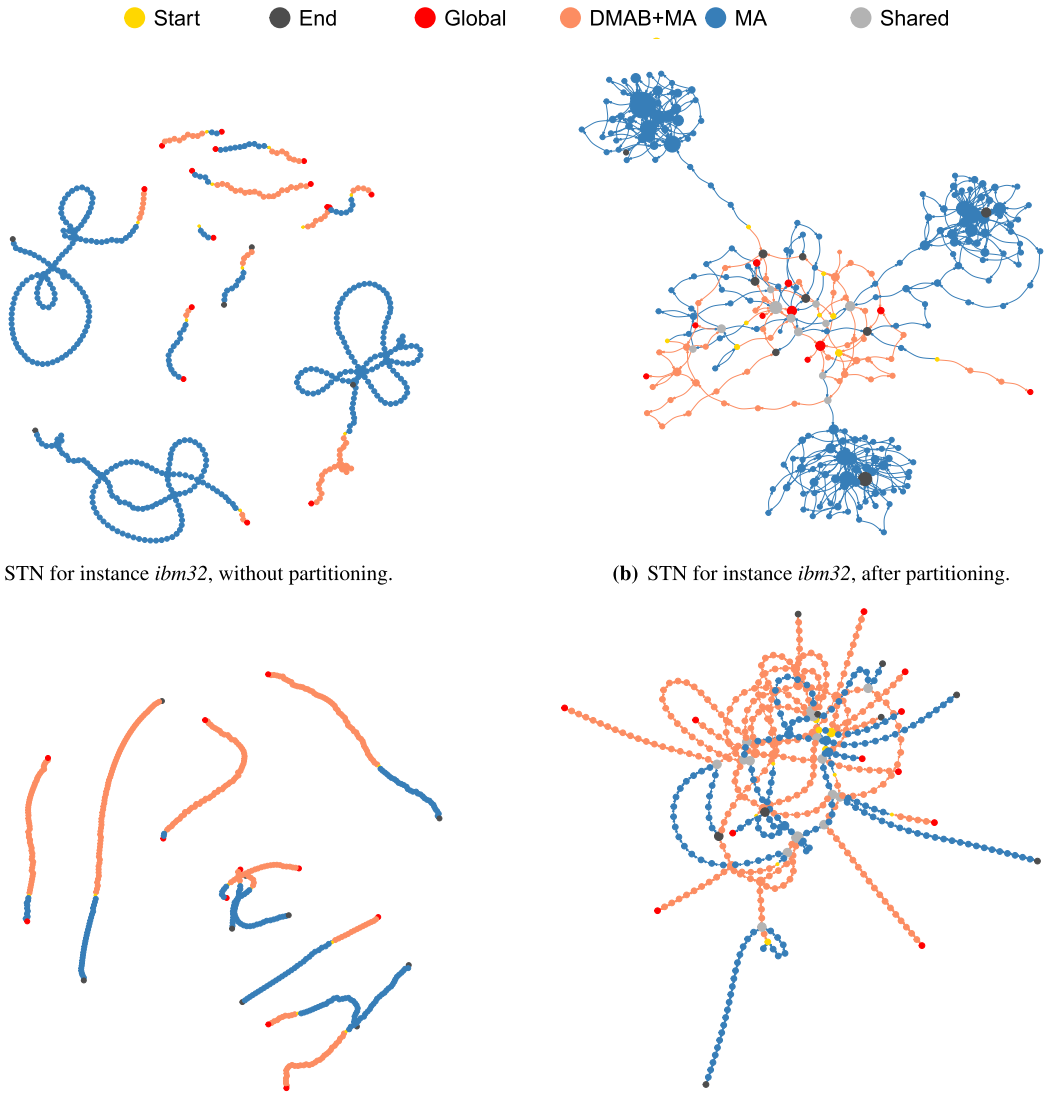
FIGURE 2. Sampled solutions obtained by executing the MA and DMAB+MA algorithms over the instance *path200*, and mapped to the Euclidean 3D space by applying multidimensional scaling. Solutions with small (better) fitness values are represented with dark violet points in the plot.

B. MULTIDIMENSIONAL SCALING VISUALIZATION

After obtaining the samples from the MA and DMAB+MA algorithms, the pair-wise interchange distances [54] between unique solutions are computed and recorded in a distance matrix $\mathcal{D}$. Then, our proposed partitioning method, based on multidimensional scaling [28], [29], maps the sampled solutions (in n dimensions) to 3-tuples in the Euclidean 3D space represented with an $n \times 3$ matrix, called $\mathcal{X}$. This is done by employing double centering and eigenvalue decomposition over the matrix $\mathcal{D}$. The resulting coordinate matrix configuration $\mathcal{X}$ minimizes a loss function, called strain, which warranties to find a mapping respecting as much as possible (i.e., with the smallest error rate) the inter-location distances stored in $\mathcal{D}$. All along this procedure, we keep track of the algorithm that produced each solution.

Figures 1 and 2 present the results obtained by our modeling and partitioning steps, based on multidimensional scaling, when applied to two representative instances: *can144* and *path200* (similar results are obtained with the rest of the tested graphs). The fitness of each point in these plots corresponds to the average of the fitness values for all the solutions that share the same Euclidean position. A color map is used to represent with dark violet, solutions with small (better) fitness values, and with light yellow tones those having large fitness costs. This kind of visualization, previously used in [30], [55], is useful to reveal the fitness variations around specific known solutions, such as the best-found. However, they lack the trajectory component to reveal how the transitions among solutions occurred through the search process.

For instance, in Figure 2 we clearly observe at right hand different zones of the search space which concentrate reduced groups of solutions with small fitness values (dark violet points), while at left hand a cluster of yellow points (i.e., large fitness cost points) is surrounded by medium fitness cost points. The former represent a zone of the search space less frequently visited by the algorithm, while the latter one seems to be a more accessible region for the algorithms. From these plots, nonetheless, it is not possible to identify if the analyzed algorithm has followed a particular search trajectory connecting these two distant zones of the space. As we will



(a) STN for instance *ibm32*, without partitioning.

(b) STN for instance *ibm32*, after partitioning.

(c) STN for instance *can144*, without partitioning.

(d) STN for instance *can144*, after partitioning.

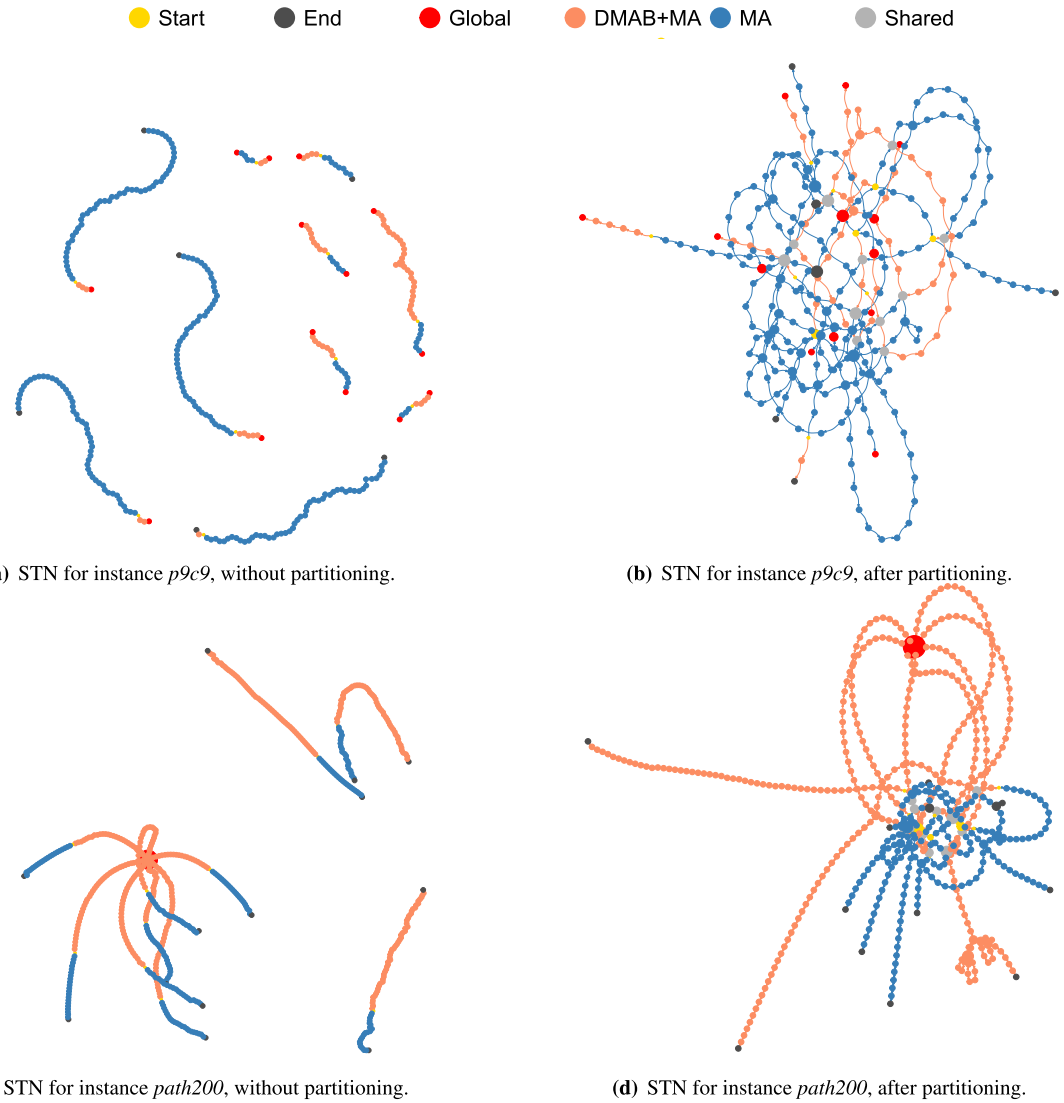
FIGURE 3. Merged STNs constructed for comparing the search dynamics of the two state-of-the-art algorithms for the CBSP over the instances *ibm32* and *can144*.

see the STNs, presented next, represent a better alternative visualization that overcomes this drawback.

C. STN ANALYSIS

The following step in constructing the STNs consist in recovering the set of search space sites visited by each algorithm (i.e., the nodes of the STN) from the coordinate matrix $\mathcal{X}$. Next, the obtained coordinate values are rounded up to integers, and those sites having exactly the same resulting 3D coordinates are grouped to become a location of the STN under construction. At that point, these locations are used to produce two single STNs: $STN_i(V_i, E_i)$ and $STN_j(V_j, E_j)$, one for each analyzed algorithm. Both single STNs are combined to produce a merged STN $STN_{A_i,A_j} = (V_{A_i} \cup V_{A_j}, E_{A_i} \cup E_{A_j})$, which allow us to compare the search dynamics of the two best-known algorithms for solving the CBSP.

The set of steps described previously, to produce merged STNs, were applied to each one of the 23 selected CBSP instances in this experiment. Figures 3 and 4 present the resulting STNs for four representative host graphs: *ibm32*, *can144*, *p9c9*, and *path200*. In these figures we can observe at right hand merged STNs produced by employing our proposed partitioning method. In contrast, those STNs depicted at left hand were built up without mapping single solutions to locations in the search space, i.e., without applying our multidimensional scaling based partitioning method. The subnetworks colored in blue correspond to the MA algorithm, while those in orange belong to the DMAB+MA algorithm. The first nodes in the search trajectories are marked in yellow,



(a) STN for instance *p9c9*, without partitioning.

(b) STN for instance *p9c9*, after partitioning.

(c) STN for instance *path200*, without partitioning.

(d) STN for instance *path200*, after partitioning.

FIGURE 4. Merged STNs constructed for comparing the search dynamics of the two state-of-the-art algorithms for the CBSP over the instances *p9c9* and *path200*.

the best ones in red, and those visited by both analyzed algorithms (shared nodes) in light gray. The size of the nodes is proportional to the number of times it was visited. Additionally, the end nodes with worse fitness value than the one of optimal/best-known solutions are depicted in dark gray color to identify them.

Table 3 presents the set of test instances we employed in this experimentation, including the name, order, size, and value of the optimum/best-known solution [24], [25] for each one of them. It continues by presenting network metrics evaluated for the single STNs of each algorithm analyzed (MA, and DMAB+MA), regarding their number of nodes, length of the shortest path to the closest end node/best node, and the number of paths of such length. Shortest paths of length zero correspond to cases where the initial solution and the optimal/best-know solution were close enough to be mapped into the same network location by our search space partitioning method. The structural metrics for the merged STNs are introduced in Table 4. It displays the number of nodes and edges, the number of shared nodes, best nodes, and end nodes. The in-strength for end nodes is marked in bold when it is equal or larger than the in-strength of the best nodes. Problem instances where this occurs can be considered quite hard to solve, since it indicates that the search trajectories can be deviated to locations of inferior fitness quality.

One of the first noticeable aspects in the STNs produced is that the trajectories of the DMAB+MA algorithm are shorter and less likely to visit previously encountered solutions (to produce search cycles). They finish more frequently in best nodes and the DMAB+MA subnetworks, in the merged STNs, have fewer nodes as well as shorter paths. This supports previous claims for DMAB+MA to be the best from

TABLE 3. STN metrics produced individually by the MA [24] and DMAB+MA [25] algorithms over the complete set of selected instances.

Problem instance		DMAB+MA			MA		
name	opt/b-k	v. nodes	s. path	n. paths	v. nodes	s. path	n. paths
path100	99	304	11	9	376	6	9
path200	199	378	19	10	214	11	8
cycle100	100	343	3	10	363	15	10
cycle200	200	260	8	10	218	8	8
wheel100	2600	60	2	31	383	1	24
wheel200	10200	101	5	21	223	2	11
cyclePow100-2	300	266	2	10	303	2	10
cyclePow100-10	5500	32	2	9	103	0	8
cyclePow200-2	600	239	1	10	226	1	9
cyclePow200-10	11000	34	2	9	114	1	9
p9p9	516	89	0	88	263	1	88
c9c9	873	231	1	110	291	1	110
k9k9	8280	135	0	104	179	2	82
p9c9	745	82	1	112	223	0	80
p9k9	1728	561	3	101	141	2	91
c9k9	1809	252	1	126	117	0	109
can24	182	37	3	18	57	0	12
ibm32	405	78	1	82	217	0	73
curtis54	411	535	2	80	256	1	80
will57	335	393	0	90	222	2	90
ash85	913	280	2	54	304	1	54
nos4	1031	56	0	30	69	0	24
can144	1776	332	3	92	214	4	42

the two studied metaheuristics, as it clearly demonstrates that the algorithm can obtain better results, in terms of solution quality, than MA while offering more efficient search space exploring capabilities.

Our STN analysis also shows that the trajectories for both analyzed algorithms tend to finish in very different regions of the search space. For example, in the STN constructed for instance *can144*, shown in Figure 3(c), there are eleven unique locations corresponding to optimal/best-known solutions (red nodes), and nine other that are end nodes of worse quality (dark gray nodes). In general, the end and best nodes tend to not be shared and to belong to distinct regions of the search space, even after applying the partitioning process as it can be seen in Figure 3(d). This is an indicator of a multimodal fitness landscape with high quality solutions scattered across it.

The shared nodes in the STNs provide interesting information, since they represent regions explored by both algorithms and the ways on which they diverted from there. A high number of shared nodes could be associated to the existence of narrow funnel structures in the underlying fitness landscape. In Figures 3(b), 3(d), 4(b), and 4(d) it was observed that shared nodes tend to be close to the best and end locations. In several cases, only the DMAB+MA algorithm is able to reach the best nodes after passing from a shared node. Furthermore, DMAB+MA is able to accomplish this task employing fewer transitions.

The problem instances constructed as the Cartesian product of two graphs (*p9p9*, *c9c9*, *k9k9*, *p9c9*, *p9k9*, and *c9k9*) usually have a higher number of shared nodes in proportion to the total number of nodes in the corresponding STN (compare columns 2 and 4 in Table 4). This could signal that the fitness landscapes of these problem instances have structures that conduct to different high quality areas.

The results of the STN analysis using graph metrics, reported in Tables 3 and 4, reveal interesting information related to the structure of the networks and the differences in difficulty of the graph topologies tested. The instances of the *path*, *cycle*, and *power of cycle* topologies had a single best node regardless of their size (see column 5 in Table 4). Since the value of the optimum is known for these topologies, we know that the network locations corresponding to those best nodes contain in fact optimal solutions.

In the *path* and *cycle* topologies, the in-strength of the best node becomes smaller than that of the end nodes for the larger instances (see columns 7 and 8 in Table 4). This suggests that the search space for the *path* and *cycle* graphs contains very few optimal solutions. Finding the optimum then becomes harder because it seems to be also larger areas of suboptimal solutions that result difficult to escape from, specially for the MA. In this sense, these two topologies could be considered harder to solve than the *power of cycle* one, since the in-strength for the best nodes was always higher than for the end nodes. In fact, their in-strength of the best node increased for those instances with power $k = 10$, which have a larger number of edges. The *wheel* graphs had smaller in-strength for the best nodes when the instance size increases as well, however their number of different best nodes is larger. They present also more paths leading to them, which would imply that this topology is easier to tackle, because the optimal solutions are not as isolated as in the case of the *cycle* and *path* graphs.

TABLE 4. Structural metrics for merged STNs produced for the complete set of selected instances by two state-of-the-art metaheuristics for the cyclic bandwidth sum minimization problem: MA [24] and DMAB+MA [25].

Instance	nodes	edges	shared nodes (%)		best nodes	end nodes	in-strength best	in-strength end
path100	654	706	26	(3.98)	1	9	0.524	0.476
path200	571	606	21	(3.68)	1	13	0.318	0.682
cycle100	683	739	23	(3.37)	1	9	0.550	0.450
cycle200	456	483	22	(4.82)	1	13	0.350	0.650
wheel100	429	432	14	(3.26)	11	9	0.550	0.450
wheel200	308	314	16	(5.19)	7	11	0.429	0.571
cyclePow100-2	551	578	18	(3.27)	1	10	0.476	0.524
cyclePow100-10	124	135	11	(8.87)	1	3	0.850	0.150
cyclePow200-2	448	461	17	(3.79)	1	12	0.400	0.600
cyclePow200-10	137	142	11	(8.03)	1	6	0.700	0.300
p9p9	303	450	49	(16.17)	11	9	0.588	0.412
c9c9	466	589	56	(12.02)	11	9	0.556	0.444
k9k9	295	315	19	(6.44)	14	6	0.704	0.296
p9c9	280	322	25	(8.93)	13	6	0.679	0.321
p9k9	653	851	49	(7.50)	11	9	0.351	0.649
c9k9	338	392	31	(9.17)	15	3	0.828	0.172
can24	83	80	11	(13.25)	15	2	0.900	0.100
ibm32	269	496	26	(9.67)	10	9	0.404	0.596
curtis54	646	1708	145	(22.45)	8	12	0.159	0.841
will57	529	1049	86	(16.26)	9	10	0.294	0.706
ash85	493	974	91	(18.46)	6	13	0.395	0.605
nos4	114	109	11	(9.65)	18	2	0.905	0.095
can144	520	551	26	(5.00)	11	9	0.500	0.500
average	406.52	542.70	34.96	(8.60)	7.74	8.43	0.54	0.46

The group of the Cartesian products (*p9p9*, *c9c9*, *k9k9*, etc.) presented some of the highest numbers of best nodes, generally in combination with numerous shortest paths towards them and a small number of end nodes. While this was also the case for some of the Harwell-Boeing graphs, like *can24* and *nos4*, this subset of instances is not as homogeneous as the other ones, since it contains problems from diverse engineering areas. Therefore, we observed more variation in the structures of their STNs, for example, *curtis54* and *will57* both had very small in-strength for the best nodes and for *can144* it was equal to that of the end nodes.

VI. CONCLUSION

This work presented an analysis, based on search trajectory networks (STNs), of two state-of-the-art metaheuristics for the cyclic bandwidth sum minimization problem (CBSP): MA [24] and DMAB+MA [25].

Search space partitioning is an essential step during the construction of STNs models, which consists in mapping the solution sampled by the studied algorithms to locations in the search space. We introduced a novel search space partitioning method based on the classical multidimensional scaling [28], [29] for reducing solutions (in n dimensions) to 3-tuples in the Euclidean 3D space, while preserving their distances. This partitioning method is more generally applicable than the previously proposed methods [15], [16], as it can be applied to any solution representation, either discrete or continuous, as soon as a suitable distance function between solutions is available.

The STN analysis carried out helped us to infer different important characteristics of the fitness landscape associated to the problem of study. For some of the CBSP instances we considered (*wheel*, *Cartesian product*, and *Harwell-Boeing* graphs), their search space contains multiple optimal/best-known solutions that are sparsely distributed across the fitness landscape. Meanwhile, for other graph topologies (*path*, *cycle*, *powers of cycle*), there seem to be specific regions containing the optimum. It also allowed us to identify that certain regions traversed by both studied algorithms are close to high quality areas, but it is usual that only the DMAB+MA algorithm has the ability to reach them, while the MA instead gets to worse quality end nodes.

The evidence from this STN-based study helps to demonstrate that DMAB+MA is more efficient for conducting the search process, as observed in its shorter trajectories, avoidance of areas where the MA gets trapped, shorter paths to the optimal/best-known, and higher success rates.

Further work would include: a) refining and further testing the partitioning method for other combinatorial problems, b) using STNs to analyze how the change in the fitness function [27] component alone helps to shape the search trajectories, and c) researching the correlation (if any) between the STN measurements and the actual size of the attraction basins.

REFERENCES

[1] Y. Zhang, S. Wang, and G. Ji, "A comprehensive survey on particle swarm optimization algorithm and its applications," *Math. Problems Eng.*, vol. 2015, p. 38, Feb. 2015.

[2] K. S. Lee and Z. W. Geem, "A new structural optimization method based on the harmony search algorithm," *Comput. Struct.*, vol. 82, nos. 9–10, pp. 781–798, 2004.
[3] O. K. Erol and I. Eksin, "A new optimization method: Big bang–big crunch," *Adv. Eng. Softw.*, vol. 37, no. 2, pp. 106–111, 2006.
[4] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE Comput. Intell. Mag.*, vol. 1, no. 4, pp. 28–39, Nov. 2006.
[5] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," Dept. Comput. Eng., Fac. Eng., Erciyes Univ., Kayseri, Turkey, Tech. Rep. TR06, 2005.
[6] X.-S. Yang and X. He, "Firefly algorithm: Recent advances and applications," *Int. J. Swarm Intell.*, vol. 1, no. 1, pp. 36–50, 2013.
[7] S. Talatahari and M. Azizi, "Chaos game optimization: A novel metaheuristic algorithm," *Artif. Intell. Rev.*, vol. 54, pp. 917–1004, 2021.
[8] M. Mohammadi, R. Tavakkoli-Moghaddam, A. Siadat, and Y. Rahimi, "A game-based meta-heuristic for a fuzzy bi-objective reliable hub location problem," *Eng. Appl. Artif. Intell.*, vol. 50, pp. 1–19, Apr. 2016.
[9] D. Greiner, J. Periaux, J. M. Emperador, B. Galván, and G. Winter, "Game theory based evolutionary algorithms: A review with Nash applications in structural engineering optimization problems," *Arch. Comput. Methods Eng.*, vol. 24, no. 4, pp. 703–750, Nov. 2016.
[10] H. Moayedi, M. Gör, M. Khari, L. K. Foong, M. Bahiraei, and D. T. Bui, "Hybridizing four wise neural-metaheuristic paradigms in predicting soil shear strength," *Measurement*, vol. 156, May 2020, Art. no. 107576.
[11] E.-G. Talbi, "Combining metaheuristics with mathematical programming, constraint programming and machine learning," *Ann. Oper. Res.*, vol. 240, no. 1, pp. 171–215, May 2016.
[12] E.-G. Talbi, "A unified taxonomy of hybrid metaheuristics with mathematical programming, constraint programming and machine learning," in *Hybrid of Metaheuristics* (Studies in Computational Intelligence), E.-G. Talbi, Ed. Berlin, Germany: Springer, 2013, pp. 3–76.
[13] E. K. Burke, M. R. Hyde, G. Kendall, G. Ochoa, E. Özcan, and J. R. Woodward, "A classification of hyper-heuristic approaches: Revisited," in *Handbook of Metaheuristics* (International Series in Operations Research & Management Science), M. Gendreau and J. Y. Potvin, Eds. Cham, Switzerland: Springer, 2019, pp. 453–477.
[14] E. Özcan, B. Bilgin, and E. E. Korkmaz, "A comprehensive analysis of hyper-heuristics," *Intell. Data Anal.*, vol. 12, no. 1, pp. 3–23, 2008.
[15] G. Ochoa, K. M. Malan, and C. Blum, "Search trajectory networks of population-based algorithms in continuous spaces," in *Proc. Int. Conf. Appl. Evol. Comput.*, in Lecture Notes in Computer Science, vol. 12104, P. A. Castillo, J. L. J. Laredo and F. F. de Vega, Eds. Cham, Switzerland: Springer, 2020, pp. 70–85.
[16] G. Ochoa, K. M. Malan, and C. Blum, "Search trajectory networks: A tool for analysing and visualising the behaviour of metaheuristics," *Appl. Soft Comput.*, vol. 109, Sep. 2021, Art. no. 107492, doi: 10.1016/j.asoc.2021.107492.
[17] S. Verel, G. Ochoa, and M. Tomassini, "Local optima networks of NK landscapes with neutrality," *IEEE Trans. Evol. Comput.*, vol. 15, no. 6, pp. 783–797, Dec. 2011.
[18] G. Ochoa, M. Tomassini, S. Vérel, and C. Darabos, "A study of NK landscapes' basins and local optima networks," in *Proc. 10th Annu. Conf. Genet. Evol. Comput. (GECCO)*, 2008, p. 555.
[19] K. Price, R. Storn, and J. Lampinen, "Appendix A. 1: Unconstrained unimodal test functions," in *Differential Evolution: A Practical Approach to Global Optimization* (Natural Computing Series), K. Price, R. Storn, and J. Lampinen, Eds. Berlin, Germany: Springer, 2005, pp. 514–533.
[20] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster," *IEEE Trans. Evol. Comput.*, vol. 3, no. 2, pp. 82–102, Jul. 1999.
[21] S. K. Mishra, "Performance of repulsive particle swarm method in global optimization of some important test functions: A fortran program," Social Sci. Res. Netw. SSRN, Rochester, NY, USA, Tech. Rep. 924339, 2006.
[22] M. T. Melo, S. Nickel, and F. Saldanha-Da-Gama, "Facility location and supply Chain management—A review," *Eur. J. Oper. Res.*, vol. 196, no. 2, pp. 401–412, 2009.
[23] Y. Chen and J. Yan, "A study on cyclic bandwidth sum," *J. Combinat. Optim.*, vol. 14, nos. 2–3, pp. 295–308, Oct. 2007.
[24] E. Rodriguez-Tello, V. Narvaez-Teran, and F. Lardeux, "Comparative study of different memetic algorithm configurations for the cyclic bandwidth sum problem," in *Parallel Problem Solving From Nature*. Cham, Switzerland: Springer, 2018, pp. 82–94.
[25] E. Rodriguez-Tello, V. Narvaez-Teran, and F. Lardeux, "Dynamic multi-armed bandit algorithm for the cyclic bandwidth sum problem," *IEEE Access*, vol. 7, pp. 40258–40270, 2019.
[26] A. Fialho, L. Da Costa, M. Schoenauer, and M. Sebag, "Dynamic multi-armed bandits and extreme value-based rewards for adaptive operator selection in evolutionary algorithms," in *Learning and Intelligent Optimization*, vol. 5851, T. Stützle, Ed. Berlin, Germany: Springer, 2009, pp. 176–190.
[27] E. Rodriguez-Tello, F. Lardeux, A. Duarte, and V. Narvaez-Teran, "Alternative evaluation functions for the cyclic bandwidth sum problem," *Eur. J. Oper. Res.*, vol. 273, no. 3, pp. 904–919, 2019.
[28] A. Mead, "Review of the development of multidimensional scaling methods," *J. Roy. Stat. Soc., D, Statistician*, vol. 41, no. 1, pp. 27–39, 1992.
[29] I. Borg, P. J. Groenen, and P. Mair, *Applied Multidimensional Scaling and Unfolding* (SpringerBriefs in Statistics) 2nd ed. Cham, Switzerland: Springer, 2018.
[30] D. C. Porumbel, J.-K. Hao, and P. Kuntz, "A search space 'cartograph' for guiding graph coloring heuristics," *Comput. Oper. Res.*, vol. 37, no. 4, pp. 769–778, Apr. 2010.
[31] J. Diaz, J. Petit, and M. Serna, "A survey of graph layout problems," *ACM Comput. Surv.*, vol. 34, no. 3, pp. 313–356, 2002.
[32] F. R. K. Chung, "Labelings of graphs," in *Selected Topics in Graph Theory*, vol. 3, L. W. Beineke and R. J. Wilson, Eds. London, U.K.: Academic, 1988, ch. 7, pp. 151–168.
[33] S. N. Bhatt and F. T. Leighton, "A framework for solving VLSI graph layout problems," *J. Comput. Syst. Sci.*, vol. 28, no. 2, pp. 300–343, 1984.
[34] F. Shahrokhi, O. Sýkora, L. A. Székely, and I. Vrťo, "On bipartite drawings and the linear arrangement problem," *SIAM J. Comput.*, vol. 30, no. 6, pp. 1773–1789, Jan. 2001.
[35] P. Mutzel, "A polyhedral approach to planar augmentation and related problems," in *Proc. 3rd Annu. Eur. Symp. Algorithms*, in Lecture Notes in Computer Science, vol. 979, P. Spirakis, Ed. Berlin, Germany: Springer, 1995, pp. 494–507.
[36] L. H. Harper, "Optimal assignment of numbers to vertices," *SIAM J. Appl. Math.*, vol. 12, no. 1, pp. 131–135, 1964.
[37] C. Zhao and B. Parhami, "Virtual network embedding through graph eigenspace alignment," *IEEE Trans. Netw. Service Manage.*, vol. 16, no. 2, pp. 632–646, Jun. 2019.
[38] B. Monien and I. Sudborough, "Embedding one interconnection network in another," *Comput. Graph Theory, Comput. Supplementum*, vol. 7, pp. 257–282, Feb. 1990.
[39] M. Sinnl, "A note on computational approaches for the antibandwidth problem," *Central Eur. J. Oper. Res.*, vol. 1, pp. 1–21, Jun. 2020.
[40] V. Liberatore, "Multicast scheduling for list requests," in *Proc. 21st Annu. Joint Conf. IEEE Comput. Commun. Soc.*, vol. 2, New York, NY, USA, Jun. 2002, pp. 1129–1137.
[41] C. H. Papadimitriou, "The NP-completeness of the bandwidth minimization problem," *Computing*, vol. 16, no. 3, pp. 263–270, Sep. 1976.
[42] J. Y.-T. Leung, O. Vornberger, and J. D. Witthoff, "On some variants of the bandwidth minimization problem," *SIAM J. Comput.*, vol. 13, no. 3, pp. 650–667, Jul. 1984.
[43] Y. Lin, "The cyclic bandwidth problem," *J. Syst. Sci. Complex.*, vol. 7, no. 3, p. 282, 1994.
[44] I. Safro, D. Ron, and A. Brandt, "Graph minimum linear arrangement by multilevel weighted edge contractions," *J. Algorithms*, vol. 60, no. 1, pp. 24–41, Jul. 2006.
[45] F. Gavril, "Some NP-complete problems on graphs," in *Proc. 11st Conf. Inf. Sci. Syst.* Baltimore, MD, USA: Johns Hopkins Univ., 1977, pp. 91–95.
[46] E. G. Pardo, N. Mladenović, J. J. Pantrigo, and A. Duarte, "Variable formulation search for the cutwidth minimization problem," *Appl. Soft Comput.*, vol. 13, no. 5, pp. 2242–2252, May 2013.
[47] P. C. B. Lam, W. C. Shiu, and W. H. Chan, "On bandwidth and cyclic bandwidth of graphs," *Ars Combinatoria*, vol. 47, no. 3, pp. 147–152, 1997.
[48] J. Yuan, "Cyclic arrangement of graphs," in *Graph Theory Notes New York*. New York, NY, USA: New York Academy of Sciences, 1995, pp. 6–10.
[49] H. Schröder, O. Sýkora, and I. Vrťio, "Cyclic cutwidths of the two-dimensional ordinary and cylindrical meshes," *Discrete Appl. Math.*, vol. 143, nos. 1–3, pp. 123–129, Sep. 2004.
[50] H. Jianxiu, "Cyclic bandwidth sum of graphs," *Appl. Math.-A J. Chin. Univ.*, vol. 16, no. 2, pp. 115–121, 2001.
[51] D. Satsangi, K. Srivastava, and Gursaran, "General variable neighbourhood search for cyclic bandwidth sum minimization problem," in *Proc. Students Conf. Eng. Syst.*, Allahabad, India, Mar. 2012, pp. 1–6.
[52] R. Hamon, P. Borgnat, P. Flandrin, and C. Robardet, "Relabelling vertices according to the network structure by minimizing the cyclic bandwidth sum," *J. Complex Netw.*, vol. 4, no. 4, pp. 534–560, 2016.

[53] C. Blum and G. Ochoa, "A comparative analysis of two matheuristics by means of merged local optima networks," *Eur. J. Oper. Res.*, vol. 290, no. 1, pp. 36–56, Apr. 2021.
[54] V. A. Cicirello and R. Cernera, "Profiling the distance characteristics of mutation operators for permutation-based genetic algorithms," in *Proc. 26th Int. Florida Artif. Intell. Res. Soc. Conf.*, St. Pete Beach, FL, USA, May 2013, pp. 1–6.
[55] X. Lai and J.-K. Hao, "Iterated maxima search for the maximally diverse grouping problem," *Eur. J. Oper. Res.*, vol. 254, no. 3, pp. 780–800, Nov. 2016.
[56] G. Csárdi and T. Nepusz, "The igraph software package for complex network research," *InterJ., Complex Syst.*, vol. 1695, no. 5, pp. 1–9, 2006.
[57] T. Kamada and S. Kawai, "An algorithm for drawing general undirected graphs," *Inf. Process. Lett.*, vol. 31, no. 1, pp. 7–15, 1989.
[58] T. M. J. Fruchterman and E. M. Reingold, "Graph drawing by force-directed placement," *Softw., Pract. Exper.*, vol. 21, no. 11, pp. 1129–1164, 1991.
[59] M. Gagliolo and J. Schmidhuber, "Algorithm selection as a bandit problem with unbounded losses," in *Learning and Intelligent Optimization*, C. Blum and R. Battiti, Eds. Berlin, Germany: Springer, 2010, pp. 82–96.
[60] J. Belluz, M. Gaudesi, G. Squillero, and A. Tonda, "Operator selection using improved dynamic multi-armed bandit," in *Proc. Annu. Conf. Genet. Evol. Comput.*, New York, NY, USA, Jul. 2015, pp. 1311–1317.
[61] T. L. Lai and H. Robbins, "Asymptotically efficient adaptive allocation rules," *Adv. Appl. Math.*, vol. 6, no. 1, pp. 4–22, Mar. 1985.
[62] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Mach. Learn.*, vol. 47, no. 2, pp. 235–256, 2002.
[63] D. V. Hinkley, "Inference about the change-point from cumulative sum tests," *Biometrika*, vol. 53, no. 3, pp. 509–523, 1971.
[64] M. López-Ibánez, J. Dubois-Lacoste, T. Stützle, and M. Birattari, "The irace package: Iterated race for automatic algorithm configuration," Institut Recherches Interdisciplinaires Développements en Intelligence Artificielle, Univ. Libre Bruxelles, Brussels, Belgium, Tech. Rep. TR/IRIDIA/2011-004, 2011.

VALENTINA NARVAEZ-TERAN received the B.S. degree in information technologies engineering from the Polytechnic University of Victoria, Mexico, in 2014, and the M.S. degree in engineering sciences and computer technologies from the Center for Research and Advanced Studies, National Polytechnic Institute (Cinvestav), Mexico, in 2016. She is currently pursuing the Ph.D. degree in engineering sciences and computer technologies with the Cinvestav Tamaulipas, Victoria, Mexico. Her research interests include combinatorial optimization, graph embedding problems, and metaheuristics design.

GABRIELA OCHOA received the Ph.D. degree from the University of Sussex, U.K. She is currently a Professor in computing science with the University of Stirling, U.K. She has held academic and research positions with University Simon Bolivar, Venezuela, and the University of Nottingham, U.K. Her recent work on network-based models of fitness landscapes has enhanced their descriptive and visualization capabilities, producing a number of publications, including four best-paper awards and four other nominations at leading venues. She collaborates cross-disciplines in the use of computational intelligence in healthcare and conservation. Her Google scholar citations currently report over 5,600 citations with a H-index of 38. Her research interests include the foundations and applications of evolutionary algorithms and metaheuristics, with emphasis on autonomous search, fitness landscape analysis, and combinatorial optimisation. She has been active in organization and editorial roles within leading Evolutionary Computation venues, including the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION and the *Evolutionary Computation* journal.

EDUARDO RODRIGUEZ-TELLO (Member, IEEE) was born in Mexico City, Mexico, in 1973. He received the M.S. degree in computer science from the ITESM, Cuernavaca, Mexico, in 1999, and the Ph.D. degree in informatics from the University of Angers, France, in 2007.

Since 2008, he has been an Associate Professor (Cinvestav-3B Researcher) with the Cinvestav Tamaulipas, Victoria, Mexico. He has authored and coauthored a book and over 50 technical papers and book chapters. His publications currently report over 711 citations in Google Scholar with a H-index of 14. His current research interests include evolutionary computation and the design and implementation of effective metaheuristic algorithms for solving large-scale combinatorial optimization problems arising in various application areas like bioinformatics and graph theory.