IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 20, 2019, accepted March 11, 2019, date of publication March 21, 2019, date of current version April 9, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2906840

Dynamic Multi-Armed Bandit Algorithm for the Cyclic Bandwidth Sum Problem

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

[1]CINVESTAV Tamaulipas, Ciudad Victoria 87130, Mexico
[2]LERIA, Université d'Angers, 49045 Angers, France

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

This work was supported in part by the Mexican Secretariat of Public Education under Grant SEP-CINVESTAV (2019–2020) 00114.

ABSTRACT Memetic algorithms (MAs) are a powerful resource when dealing with optimization problems, combining the diversification of the population-based approaches with the intensification of local search. However, their success depends on the combination of operators and their ability to cope with the intrinsic difficulties of a problem. Choosing the most suitable combination of operators that better suits a given problem (or a set of instances of a problem) has proved to be a defiant and time-consuming task. An approach to this task is the adaptive operator selection (AOS), based on the idea of choosing operators during execution time based on some reward system related to their performance. In this paper, we continue our previous work on studying the effectiveness of several operators of an MA to solve the cyclic bandwidth sum problem (CBSP), now extending the operator set and incorporating the dynamic multi-armed bandit (DMAB) framework to adaptively adjust the MA's operators. The resulting technique, named DMAB+MA, is compared to the independent MA versions in a full factorial experiment and with respect to two reference algorithms of the literature. It was found that the quality of the solutions achieved by DMAB+MA significantly improved the best-known results provided by the state-of-the-art algorithms while keeping the competitive execution times with respect to the independent MA versions. Moreover, DMAB+MA was able to provide optimal/best-known solutions for the 40 tested graphs (with different topologies) and to establish new better upper bounds for 12 of them.

INDEX TERMS Cyclic bandwidth sum problem, dynamic multi-armed bandit, adaptive operator selection, memetic algorithms.

I. INTRODUCTION

In this work we deal with an $\mathcal{NP}$-hard combinatorial optimization problem known as the Cyclic Bandwidth Sum Problem (CBSP), which is formally defined as follows. Let $G = (V, E)$ be a finite undirected graph (the guest) of order n 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 $\mathrm{Cbs}(G, \varphi^*)$ is minimum, i.e., $\varphi^* = arg\,min_{\varphi\in\Phi}\{\mathrm{Cbs}(G, \varphi)\}$ with Φ denoting the set of all possible embeddings.

Some relevant practical applications of the CBSP include VLSI designs [1], [2], code designs [3], simulation of network topologies for parallel computer systems [4], scheduling in broadcasting based networks [5], and compressed sensing in sensor networks [6].

The CBSP was first described by Yuan [7]. It belongs to the Graph Embedding Problems (GEP), which goal is to find the optimal way of embedding a guest graph into a host graph [8]. One of the most studied GEP is the Bandwidth Problem (BP) [9], which consist in embedding a graph into a path, while minimizing the maximal distance between pairs of adjacent guest's vertices. Several other GEP can be derived

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

from the BP. For example, if we are looking towards minimizing the sum of distances instead of the maximum distance, we have the Bandwidth Sum Problem (BSP) [10]. If the host graph is a cycle instead of a path, we have the Cyclic Bandwidth Problem (CBP) [11]. And if the host is a cycle and also the sum of distances is to be minimized, then we have the CBSP [12]. Several other GEP, as well as the relationships among them, have been compiled by Diaz *et al.* [13] and Lam *et al.* [14]. While the theoretical bounds of the value of the optimum for a given GEP can be used to derive upper and lower bounds of other GEP, in the practice, algorithms able to efficiently solve one GEP are not implicitly successful to solve other of them. This is the case for the CBSP in relation with the BSP and the CBP, therefore it is necessary to study its difficulties and particularities independently from other GEP.

In a previous work [15] 24 Memetic Algorithms (MA) versions for the CBSP, produced by the combination of a small set of genetic operators, were extensively evaluated. The experimental results showed that all of them were able to produce significant better results than the state-of-the-art reference methods. However, there were hints of premature convergence and large performance variations depending on the graph topologies tested. Motivated by these preliminary findings in this paper we studied the design of MA and their combination with an Adaptive Operator Selection (AOS) approach for solving the CBSP in the general case.[1]

Our approach consists in implementing the Dynamic Multi-Armed Bandit (DMAB) paradigm to automatically alternate among 96 MA independent versions (each characterized by a different set of genetic operators). We chose the DMAB paradigm because it offered documented good performance, and it was easy to incorporate into our main MA framework due to its relative simplicity and few parameters to be tuned. The results of this strategy, called DMAB+MA, were compared to those of the MA independent best performing configurations and with the state-of-the-art methods over a set of 40 instances with diverse graph topologies.

The remaining sections of this work are organized as follows. Section II presents a brief compilation of the most relevant works related to the CBSP and an overview of the DMAB paradigm. In Section III we present the main aspects of our approach to the design of DMAB+MA. Experimental setup and performances analysis are presented in Section IV. The DMAB+MA behavior is further discussed in Section V. Finally our conclusions are presented in Section VI.

II. RELATED WORK

A. CYCLIC BANDWIDTH SUM PROBLEM

The existence of a deterministic polynomial time algorithm able to produce the optimum in the general case of a known $\mathcal{NP}$-Hard problem such as the CBSP [7] is unlikely. In fact, to the best of our knowledge, no exact algorithm for solving it has ever been reported. Most of the early research about the CBSP focused in the calculation of the optimal cost value depending on the topology of graph G. In this matter, Jianxiu [12] and later Chen and Yan [16] have calculated the exact cost value of the optimal solution for *paths*, *cycles*, *wheels*, *k-th power of cycles*, and *complete bipartite* graphs. Jianxiu [12] also established the upper bounds for the graph resulting of the Cartesian product of two graphs, being those graphs paths, cycles, or complete graphs.

Only recently the CBSP has received more attention in the optimization and operation research communities where the following approaches have been proposed: a local search based approach [17], a constructive greedy heuristic [18], hybrid metaheuristic algorithms [15], and a reformulation of the evaluation function [19].

The first metaheuristic approach based in local search was a General Variable Neighborhood Search (GVNS) [17]. This algorithm starts from a lexicographical embedding which is improved using a Reduced Variable Neighborhood Search (RVNS). Then, two neighborhood operators and six perturbation operators are applied in the GVNS phase. Computational experiments, carried out on a set of graphs of order $n \leq 200$, showed that GVNS achieves optimal results for path, wheel, star, and cycle topologies. For graphs with $n \leq 64$ vertices, resulting of the Cartesian product of paths, cycles, and complete graphs, GVNS was able to produce solutions with cost under the theoretical upper bounds.

Later, Hamon *et al.* [18] designed MACH, a constructive heuristic. MACH operates by decomposing the graph into a list of disjoint paths by means of a depth-first search inspired mechanism. This mechanism traverses the graph using the Jaccard similarity index [20] as criterion to choose the next vertex to add to the path. An embedding is constructed incrementally by the aggregation of paths to a partial solution following a greedy strategy. For most of the tested instances, MACH consistently improved the solution quality achieved by GVNS, as well as its running time.

In our previous work we tested the combinations of a smaller set of genetic operators, resulting in 24 MA versions [15]. While all of them were able to produce significant better results than MACH and GVNS, the best MA version showed hints of premature convergence related to the survival selection scheme. Therefore, we considered the possibility of further improving its results by varying this genetic operator.

Moreover, in all the algorithms mentioned above the evaluation function was a direct implementation of (1). However, as pointed out by Rodriguez-Tello *et al.* [19], this function is susceptible to produce a fitness landscape with large plateaus because of the limited number of equivalence classes that it is able to induce. In an attempt to devise a consistent evaluation scheme having increased discriminative capabilities with respect to the conventional evaluation function, three alternative evaluation functions were introduced and tested. All three alternative functions presented a higher discriminative capability, but only one of them ensured full compatibility with respect to (1). This function also demonstrated a significant improvement in the solution quality produced when tested in conjunction with basic local search algorithms.

[1] For any graph topology.

Therefore, it is of our interest to investigate the effects of this alternative evaluation scheme in our MA versions and to evaluate if it can help to achieve better final results.

B. DYNAMIC MULTI-ARMED BANDIT

The Multi-Armed Bandit (MAB) problem originated in the Game Theory area [21], [22]. In this paradigm the bandit is a machine with k arms. When an arm A_i is played at time t it receives a reward equal to 1 with probability p_i. The MAB problem consists in finding how to select an arm that maximizes the cumulative reward.

Since in practice the distribution of reward probabilities is unknown, the approaches to the MAB problem calculate estimations instead. A widespread approach for estimating reward chances in the MAB problem is the Upper Confidence Bound (UCB) and its multiple variants [22]. The main idea is to assign to each arm an estimation of its rewards probabilities, i.e., a confidence, so that the arm with the best estimation (higher confidence) is played every time. Under the UCB1 variant the confidence for arm i at time $t+1$ is estimated as:

$$\hat{p}_{i,t+1} = \hat{p}_{i,t} + C\sqrt{\frac{2\log\sum_{j=1}^{k} n_{j,t}}{n_{i,t}}}\,, \tag{2}$$

where $n_{i,t}$ is the number of times that arm i has been played at time t, $p_{i,t}$ is the current reward, and C is a scaling factor to control the trade-off between exploitation of the current best arm and the exploration of other arms. Although C parameter is not present in the original definition of UCB1 [22], its use is extended [23] and it is critical for the exploitation-exploration balance [24].

MAB implementations using UCB1 have been successfully employed as an approach to the Adaptive Operator Selection (AOS) problem [25], [26] and as a way to select low level heuristics within an hyperheuristic framework [27]. Differently from the original 0-1 reward system, an arm representing an operator (or combination of operators [27]) is rewarded in function of the magnitude of the improvement it brought to a solution (or population of solutions) after having been played. Numerous alternatives to compute the value of such reward have been devised [28]–[30]. Our proposal implements a credit assignment mechanism known as extreme value-based reward [31], which is discussed further in Section III.

The Dynamic Multi-Armed Bandit (DMAB) [32] was later proposed as a method to cope with scenarios where the optimal operator to apply varies among time. While the original MAB could take considerable time to detect those variations, DMAB incorporates the Page-Hinkley (PH) test [33] to quickly detect an abrupt change in the reward of the current best arm, therefore recognizing whenever the optimal operator has just changed and then restarting the reward and confidence records. Such an approach has been known to produce good results in the fields of hyperheuristics [34] and evolutionary algorithms [23].

III. DYNAMIC MULTI-ARMED BANDIT AND MEMETIC ALGORITHMS FOR THE CBSP

In this section we describe the algorithmic operators involved in our MA versions, as well as their integration within the DMAB paradigm.

Algorithm 1 Memetic Algorithm

```
1:  input A set of operators {s, c, m, f, ss}
2:  output A solution g
3:  P ← initializePopulation(P)
4:  O ← ∅
5:  g ← P_best
6:  repeat
7:    for j ← 1 to μ do
8:      P_a, P_b ← selection(s, P)
9:      o ← crossover(c, P_a, P_b, prob_c)
10:     o' ← mutation(m, o, prob_m)
11:     o'' ← inversion(o', prob_i)
12:     O ← O ∪ o''
13:     g ← fittest individual among g, o, o', and o''
        under function f
14:   end for
15:   P ← survival(ss, P, O)
16:   O ← ∅
17:   P_best ← localsearch(P_best, tries)
18:   g ← fitter individual among current g and P_best
19: until stop criterion is met
20: return g
```

A. GENERAL MEMETIC ALGORITHM FRAMEWORK

The main body of our MA versions is presented in Algorithm 1. It takes as input the set of operators $\{s, c, m, f, ss\}$ for selection, crossover, mutation, evaluation function, and survival strategy, respectively.

In our previous work (see [15]) we implemented four selection schemes (*roulette*, *stochastic*, *random*, and *binary tournament*), two crossover mechanisms (*cyclic* and *order-based*), as well as three mutation operators (*cyclic insertion*, *reduced 3-swap*, and *cumulative swap*). Survival of individuals was determined by the $(\mu + \lambda)$ strategy, allowing only the fittest among parents and offspring to survive. For more details about those operators we refer the reader to [15]. Additionally to those operators, in this work we incorporated the (μ, λ) survival strategy that directly replaces the parent population by the offspring.

It is also of our interest to investigate the suitability of the best performing of the three alternative evaluation functions introduced by Rodriguez-Tello *et al.* [19]. This function, shown in (3), adds a floating point part to the calculation of Cbs, which allows us to distinguish among solutions of similar cost under the conventional evaluation scheme. The floating point part of the expression is a weighted sum of the frequency of occurrences d_k of every cyclic distance of cost k. Decreasing value cyclic distances have increasing weights

assigned to them. The strategy penalizes the occurrences of small value cyclic distances, considering that large cyclic distances are easier to break. For more details related to the alternative evaluation functions see [19].

$$f_1(G, \varphi) = \text{Cbs}(G, \varphi) + \sum_{k=1}^{\lfloor n/2 \rfloor} \left(\frac{1}{n2^k} \right) \cdot d_k \ . \tag{3}$$

In our MA versions the potential solutions were represented by means of the permutation encoding, since it is straightforward for this type of problem. The initial population of individuals is confirmed by μ randomly generated individuals (line 3 in Algorithm 1). After having created an initial population, the following operations take place through the generations. Firstly, the selection mechanism chooses a couple of individuals from population P for recombination (line 8). Each couple produces only one offspring o, either via crossover conditioned to probability $prob_c$, or as a copy of the fitter parent otherwise (line 9). Then, the chromosome of o is altered by a two phase mutation with independent probabilities, $prob_m$ and $prob_i$, respectively (lines 10 and 11). The first phase uses one of the three mutation operators previously mentioned (resulting in o'); the second phase acts as an additional perturbation employing a fixed cyclic inversion operator (resulting in o'').

The individual o'' is added to population O (line 12). Then, the best solution found so far (g) is updated by obtaining the fittest individual among o, o', o'', and g. The purpose of this is to avoid losing any possible improvement, since the temporary chromosomes o and o' are not actually stored in O. The best-found solution g is stored independently of the populations P and O. The survival phase (line 15) determines the group of individuals that will remain in the population for the next generation. In contrast with the common approach, we do not apply local search to all offspring individuals. Instead, local search operates for a maximum number of iterations only with the fitter individual in the surviving population, as the last step in the generation (line 17). The stop criterion for experiments with independent MA versions was a fixed number T of maximum fitness function evaluations.

B. DYNAMIC MULTI-ARMED BANDITS

In the DMAB main framework, every MA version is defined by a combination of the operators mentioned in the previous section, and it represents a different algorithm, therefore a different arm of the bandit. An arm is a set of operators $A_i = \{s_i, c_i, m_i, f_i, ss_i\}$ denoting selection s_i, crossover c_i, mutation m_i, fitness function f_i, and survival strategy ss_i. Playing an arm one time means to execute its correspondent MA version for exactly one generation as stop criterion.

At the begging of the execution all k arms are played over the initial population of individuals. Arms get a raw reward assigned by the reward mechanism. After this initial procedure, at each iteration an arm is selected based on the estimated confidence. The selected arm is then played its reward gets updated. The PH-test is applied to detected

Algorithm 2 DMAB Algorithm

```
1:  P ← initializePopulation(P)
2:  Set confidence and number of times arms have been
    played to zero
3:  for i ← 1 to K do
4:     P' ← playArm(A_i, P)
5:     Assign initial reward to A_i
6:  end for
7:  repeat
8:     Compute confidence for all arms
9:     A_s ← selectArm()
10:    P ← playArm(A_s, P)
11:    Update A_s reward and increase the number of
       times it has been played
12:    if PH-test is triggered then
13:       Set confidence and number of times arms have been
          played to zero
14:       for i ← 1 to K do
15:          P' ← playArm(A_i, P)
16:          Assign initial reward to A_i
17:       end for
18:    end if
19:    g ← fitter individual among current g and P_best
20: until stop criterion is met
21: return g
```

statistical abrupt changes in the average reward of the current best arm. The triggering of the test is taken as an indicative that the current arm is no longer the best, causing a restart of the arms. This continues until some stop criterion is met. For experiments with DMAB+MA, the stop criterion was a fixed maximum execution time s.

1) RAW REWARD AND CREDIT ASSIGNMENT MECHANISM

Whenever an arm is played its raw reward rr_i is calculated based on some measure of the improvement in the population's fitness. The method to quantify such improvement may vary among implementations. In our case, it is calculated as the normalized difference between the fitness of the best individual in the population before (f_{old}) and after (f_{new}) playing arm A_i.

$$rr_i = \frac{f_{old} - f_{new}}{f_{old}} \times 100 \tag{4}$$

The mechanism for credit assignment implemented is the extreme value-based reward [31]. It consists of maintaining a register of the raw rewards an arm has got and assigning an arm credit equal to the maximum of them. The core idea behind this is to base credit not only in how good an arm has performed the last time it was applied, but also historically. The reward register works in a FIFO fashion and its maximum size is the parameter W. However, the assumption that $W > 1$ would result in a better performance was proven wrong by our

parameter tuning experiments (see Section IV-C).

$$r_i = \max_{j=1}^{W}\{rr_j\} \tag{5}$$

2) ARM SELECTION

The arm selection is controlled by the UCB1 mechanism. The selected arm is the one that maximizes the confidence. The confidence of arm A_i is calculated as:

$$confidence_i = empRew_i + C\sqrt{\frac{2\log\sum_{j=1}^{k} plays_j}{plays_i}}, \tag{6}$$

where $empRew_i$ is the empirical reward of arm A_i, equal to the average credit it has received, $plays_i$ is the number of times arm A_i has been played, and C is an scaling factor that controls the weight of the right part of the expression. As opposed to the left part of the expression ($empRew_i$) favoring the arm with the greatest empirical reward getting the most confidence, the right part favors the use of arms that have been played the fewest times. The role of parameter C is to balance between exploitation of the best performing arms and the exploration of underused arms.

3) PAGE-HINKLEY TEST

The Page-Hinkely test is a mechanism to detect an abrupt change in the underlying success probability distribution of arms, which estimation is provided by the UCB1 method discussed previously. Such a change reflects in the way that after playing the most trusted arm its reward is significantly different from its historical values. There are two key parameters involved in the PH-test: δ, which provides tolerance for slowly varying scenarios, and λ, which is a trade-off between false alarms and unnoticed changes (false positives and false negatives) [32].

Let A_i be the arm played at time t, the average standard deviation of the reward values for A_i up to time t is $avgDev_{i,t} = avgDev_{i,t-1} + (empRew_i - rr_i + \delta)$. The maximum value of $avgDev_{i,t}$ is updated as $maxDev_{i,t} = \max(avgDev_{i,t}, maxDev_{i,t-1})$. The PH-test is triggered if $maxDev_{i,t} - avgDev_{i,t} > \lambda$. Whenever the PH-test is triggered the arms are restarted, meaning that their rewards, confidence, and number of times played records are back to zero, as well as the values $avgDev_{i,t}$ and $maxDev_{i,t}$.

IV. EXPERIMENTAL RESULTS

In this section we present the conditions under which our experiments took place. We describe the set of instances, algorithm input parameters, and metrics involved, and provide visual and statistical performance comparisons. Firstly we compare the best MA version with respect to our previous work. Then, the performance of this version is compared with the DMAB+MA and the other state-of-the-art methods.

A. EXPERIMENTAL SETUP

A first step towards determining the effectiveness of our MA implementations for solving the CBSP is to identify the best performing version. For this purpose, all the possible combinations of genetic operators were tested under a common instance benchmark, parameters and initial populations. This includes the algorithms reported in our previous work [15]. The algorithms were coded in C language and compiled in gcc 4.4.7 with the O3 flag. All experiments were ran sequentially on the same platform, an Intel Xeon CPU X5550 at 2.67 GHz and 16 GB in RAM.

TABLE 1. Operators keys.

Operator	Key	Operator	Key
Stochastic selection	S1	*Reduced triple-swap* mutation	M2
Roulette selection	S2	*Cumulative swap* mutation	M3
Random selection	S3	(μ, λ) survival strategy	SS1
Binary tournament selection	S4	$(\mu + \lambda)$ survival strategy	SS2
Cyclic crossover	C1	Conventional evaluation function	V1
Oder-Based crossover	C2	Alternative evaluation function	V2
Cyclic insertion mutation	M1		

Table 1 summarizes the operator keys that allow us to identify a specific MA version by the combination of operators that it implements. For example, identifier S4_C2_M1_SS2_V1 stands for a MA version that implements *binary tournament* selection (S4), *cyclic* crossover (C1), *cyclic insertion* mutation (M1), $(\mu + \lambda)$ survival strategy (SS2), and the conventional evaluation function (V1).

The instances set includes 40 representative, topologically diverse graphs that have been often used in the GEP or CBSP literature [17]–[19]. The set includes 6 Cartesian products of graphs, 24 sparse matrices from the Harwell-Boeing collection, 2 *paths*, 2 *cycles*, 2 *wheels*, and 4 *k-th powers of cycle* ($k \in \{2, 10\}$). The instances size varies from 24 to 715 vertices and from 59 to 3,720 edges.

TABLE 2. Parameter settings for the MA algorithms.

Parameter	Value	Parameter	Value
Population size μ	20	Mutation probability $prob_m$	0.543
Max. evaluations T	4.0E+08	Inversion probability $prob_i$	0.240
Crossover probability $prob_c$	0.788	Max. local search iterations $tries$	10

For all the MA versions being tested we provided a fixed set of parameter values (Table 2), which were derived from the literature and from our automatized tuning experiments using the irace utility [35].

B. IDENTIFYING THE BEST MA

The best MA was identified by comparing all 96 versions in terms of the solution quality they achieved with respect to the best-know solutions. For each instance being solved by a particular algorithm, we computed the relative root mean square error (RMSE) for 31 independent executions. The tested algorithms included our MA versions, as well as GVNS [17] and MACH [18]. The RMSE metric for a certain

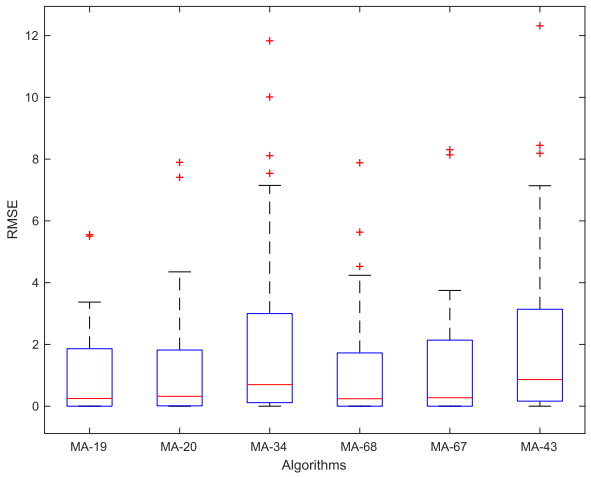

FIGURE 1. RMSE distribution for Top-5 MA group and MA-34 among the instance set.

TABLE 3. O-RMSE ranking for the Top-5 MA versions, MA-34 MACH and GVNS.

Rank	Algorithm	Operator configuration	O-RMSE (%)	Average run time (s)	Time to local opt. (s)
1	MA-19	S4_C1_M1_SS1_V1	0.983	264.526	118.634
2	MA-20	S4_C1_M2_SS1_V1	1.099	193.316	87.110
3	MA-68	S4_C1_M2_SS1_V2	1.112	273.548	153.704
4	MA-67	S4_C1_M1_SS1_V2	1.126	372.276	200.842
5	MA-43	S4_C1_M1_SS2_V1	2.031	85.555	28.749
6	MA-34	S2_C2_M1_SS2_V1	2.045	85.851	29.976
	MACH	N/A	2.659	7.725	N/A
	GVNS	N/A	4.427	612.380	900.145

test instance t is defined as:

$$\text{RMSE}(t) = 100\%\sqrt{\frac{\sum_{r=1}^{R}\left(\frac{\text{Cbs}_r(t)-\text{Cbs}^*(t)}{\text{Cbs}^*(t)}\right)^2}{R}}, \qquad (7)$$

where $\text{Cbs}_r(t)$ is the best solution quality achieved by the algorithm at execution r, R is the total number of executions, and $\text{Cbs}^*(t)$ is the best-know quality solution for instance t, achieved either by GVNS, MACH, or by any of our MA versions. An RMSE equal to 0% means the algorithm achieved the best-known solution quality in all the R executions, and therefore it is the preferred value.

For comparing the algorithm performance among the complete instance set $\mathcal{T}$, the overall root mean square error (O-RMSE) is computed as the average RMSE value for $|\mathcal{T}| = 40$ instances in the testing set.

$$\text{O-RMSE} = \frac{1}{|\mathcal{T}|}\sum_{t\in\mathcal{T}}\text{RMSE}(t)\ . \qquad (8)$$

The five MA configurations presenting the best performance in terms of solution quality are those with the lower O-RSME rates. This group is hereafter referred as Top-5 MA. Their distribution of RMSE values among instances is presented in Fig. 1. Table 3 describes the operator configuration of algorithms in the Top-5 MA, their achieved O-RMSE rate, average execution time, and average time to reach the best solution found. It also includes the reference methods from the literature, (GVNS and MACH), as well as MA-34[2] which was the best MA resulting from our preliminary work reported in [15]. The stop criterion for GVNS was a maximum running time of 900s. As reference, there are 21 out of 96 MA versions leading to better O-RMSE values than MACH, and 56 of them with better O-RMSE values than GVNS.

The presence of some operators among the Top-5 MA is consistent with the achievement of high quality solutions. Algorithms in the Top-5 MA differ from one another at most in three operators: mutation scheme, survival strategy, and evaluation scheme; while *binary tournament* and *cyclic* crossover remain present, clearly showing dominance over their competitors.

Selection by *binary tournament* ensures that the relatively fitter individuals are chosen for crossover, without imposing an excessive selection pressure, since one individual only needs to be fitter -or at least as fit- as some other to be chosen. While this may help to prevent offspring from quickly losing diversity, recombination by *cyclic* crossover prevents implicit mutations, ensuring that the offspring is not so different from their parents. The results suggest that for this particular problem, being able to select relatively fit individuals, and then inherit their genes in the absolute positions, is important and useful to produce building blocks of higher order and therefore have a fit and diverse offspring.

The only algorithm in the Top-5 MA implementing survival strategy SS2 $(\mu + \lambda)$ has also the wider O-RMSE leap with respect to the rest of the group. Lets also point out that MA-43 differs from the configuration with the best O-RMSE value (MA-19) only in the selection scheme, so the impact of the change is relevant. MA-43 is the fastest algorithm in the group, performing 4.0E+08 evaluations in only 85.555 seconds, while the other algorithms in the Top-5 MA group took more than twice this amount of time. Although, MA-43 stops improving after 28.749 seconds on average, and the best solutions it reported have worst quality than their pairs. This means that MA-43 was faster because it was not performing updates of the best-found solution, and the local search (having not found any improving solution for the best individual in the population) stopped at its first iteration. MA-43 got trapped in a locally optimal solution, and since that solution was prevented from leaving the population by the $(\mu + \lambda)$ survival strategy, it continued propagating its genes to the next generations. The fitter the new individuals became, the closer they were to the locally optimal solution, and the more chances they had to remain in the population, ultimately decreasing diversity.

We identified *binary tournament*, *cyclic* crossover, and (μ, λ) survival strategy as more successful in terms of

[2]The identification of MA versions was affected by the addition of new operators in this work. Thus, the version MA-10 from [15] is now referred here as MA-34.

TABLE 4. Statistical analysis for comparing the performance of the Top-5 MA group.

Instances	$\|V\|$	$\|E\|$	MA-19/ MA-20	MA-19/ MA-43	MA-19/ MA-67	MA-19/ MA-68	MA-20/ MA-43	MA-20/ MA-67	MA-20/ MA-68	MA-43/ MA-67	MA-43/ MA-68	MA-67 MA-68
path100	100	99	⋆	−	−	−	−	−	−	+	⋆	−
path200	200	199	+	+	+	⋆	⋆	+	−	⋆	−	−
cycle100	100	100	⋆	+	−	+	⋆	−	⋆	−	⋆	+
cycle200	200	200	+	+	+	+	+	⋆	⋆	−	−	−
wheel100	100	198	⋆	+	−	+	⋆	−	⋆	−	⋆	+
wheel200	200	398	+	+	+	+	+	⋆	−	−	−	−
cyclePow100-2	100	200	⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆
cyclePow200-2	200	400	⋆	+	+	⋆	+	⋆	⋆	−	−	−
cyclePow100-10	100	1000	⋆	⋆	⋆	+	⋆	⋆	⋆	⋆	⋆	+
cyclePow200-10	200	2000	⋆	⋆	⋆	⋆	⋆	⋆	+	⋆	⋆	⋆
c9c9	81	162	⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆
c9k9	81	405	⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆
k9k9	81	648	⋆	+	⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆
p9c9	81	153	⋆	+	+	⋆	+	⋆	⋆	⋆	−	⋆
p9k9	81	396	⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆
p9p9	81	144	⋆	+	⋆	⋆	+	⋆	⋆	−	−	⋆
jgl011	11	49	⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆
ash85	85	219	⋆	+	−	⋆	+	−	⋆	−	−	+
curtis54	54	124	⋆	+	⋆	⋆	+	⋆	⋆	−	−	⋆
ibm32	32	90	+	+	⋆	+	+	−	⋆	−	−	+
will57	57	127	⋆	+	⋆	⋆	+	⋆	⋆	−	−	⋆
impcol_b	59	281	⋆	+	⋆	⋆	+	⋆	⋆	−	−	⋆
impcol_d	425	1267	⋆	⋆	⋆	−	⋆	⋆	⋆	⋆	⋆	−
nos4	100	247	⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆
nos6	675	1290	−	+	+	−	+	+	+	−	−	−
494_bus	494	586	−	+	+	−	+	+	+	−	−	−
662_bus	662	906	−	+	+	−	+	+	+	−	−	−
685_bus	685	1282	−	+	+	−	+	+	+	−	−	−
can_24	24	68	⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆
can_144	144	576	⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆
can_292	292	1124	−	+	+	−	+	+	+	−	−	−
can_445	445	1682	−	+	+	−	+	+	+	−	−	−
can_715	715	2975	−	+	+	−	+	+	+	−	−	−
bcspwr01	39	46	⋆	+	⋆	⋆	+	⋆	⋆	−	−	⋆
bcspwr02	49	59	⋆	+	⋆	⋆	+	⋆	⋆	−	−	⋆
bcspwr03	118	179	−	+	−	−	+	⋆	−	−	−	−
bcsstk01	48	176	+	+	⋆	+	+	−	−	−	−	+
bcsstk06	420	3720	−	+	+	−	+	+	⋆	−	−	−
dwt_503	503	2762	−	+	+	−	+	+	−	−	−	−
dwt_592	592	2256	−	+	+	−	+	+	+	−	−	−
+			5	28	15	7	24	11	9	1	0	6
−			11	1	5	13	1	6	6	25	25	17
⋆			24	11	20	20	15	23	25	14	15	17
Overall winner			MA-20	MA-19	MA-19	MA-68	MA-20	MA-20	MA-20	MA-67	MA-68	MA-68

solution quality. However, for the roles of mutation and evaluation schemes we do not have obvious winners. While mutation by *cyclic insertion* and *reduced 3-swap* seem to be more competitive than mutation by *cumulative swap*, it is still unclear which one of them is the best by considering only their O-RMSE values. For example, if we focus in MA-19 and MA-20 we found that they only differ in the mutation scheme. The O-RMSE value of MA-19, which is lower than that of MA-20, may lead us to infer that *cyclic insertion* mutation (M1) is more suitable than *reduced 3-swap* (M2). However, that behaviour reverses when observing the results for MA-68 and MA-67 where the configuration implementing mutation M2 has a better O-RMSE value than the one reached with mutation M1.

In previous results from our preliminary work [15] it was observed that the algorithms incorporating *order-based* crossover and *cyclic insertion* mutation were consistently better than those incorporating *cyclic* crossover and *reduced 3-swap* mutation. However, the enlarged operator set that produces new MA versions and the instance set, which includes graphs of higher order and size, allow us to obtain new results that extend our previous knowledge on how the operators interact. This is specially remarkable for the survival strategy. For example, the best MA from our preliminary work, here named as MA-34 (S2_C2_M1_SS2_V1) ranked 6th, while MA-10 (S2_C2_M1_SS1_V1), varying only in survival strategy, ranked 87th. It is clear that no single operator is responsible for the success of the algorithm, but it is the interaction among them which truly determines it.

To further analyze the versions among the Top-5 MA, a statistical significance analysis was performed. Normality of data distributions was evaluated by the *Shapiro-Wilk* test. *Bartlett's* test was implemented to determine whether the variances of the normally distributed data were homogeneous or not. We applied *ANOVA* test in the case variance homogeneity was present and *Welch's t* parametric tests on the contrary. Meanwhile, *Kruskal-Wallis* test was implemented for non-normal data. In all cases the significance level considered was 0.05.

Table 4 presents the results of pair-wise statistical comparisons of Top-5 MA versions among the instance set.

First three columns describe the instance set by the order and size of the graphs being considered. The following ten columns compare a pair of Top-5 MA configurations, say A and B, which is denoted as A / B. The cases where version A presents a significant better performance than version B are marked in the corresponding cells as $+$, standing for a *victory* of A. Otherwise, if B outperforms A, the cell is marked with the symbol $-$, representing a *defeat* of A. The $\star$ symbol stands for *ties*: those cases where there is not statistical significant difference between A and B performance, i.e., $p_{value}(A, B) < 0.05$; and therefore the comparison cannot be decided in favor of any of them. The last three rows summarize the total victories ($+$), defeats ($-$) and ties ($\star$) for each pair of versions in the Top-5 MA group.

In these comparisons MA-43, the only algorithm in the Top-5 MA implementing $\mu+\lambda$ survival strategy (SS2), scored only 3 victories against another algorithm, and it was always defeated in overall victories, reinforcing our inferences from Table 3. However, we remark that the O-RMSE value of MA-43 is still better than the one of MACH. Meanwhile, algorithm MA-67 can only defeat MA-43 in overall victories and there are few instances for which it delivers statistical significant better results than the rest of the group.

It is interesting to observe and compare the behavior of MA-19, MA-20, and MA-68. These algorithms ranked as the three with smaller error rates and shorter times. The O-RMSE values for MA-20 and MA-68 are close to each other (0.9832 and 1.0991) differing only in 0.1159 and they vary only in evaluation scheme. When compared, MA-20 and MA-68 present the highest tie rate (25) and MA-20 is only better by a margin of 3 instances, confirming that the alternative evaluation scheme can indeed be useful. Moreover, MA-68, an algorithm implementing the alternative scheme, actually defeats MA-19 (the algorithm with the lower O-RMSE) in overall victories, and delivers results as good or even better for 33 out of 40 instances.

When it comes to comparing overall victories, MA-20 is never defeated by any other algorithm in the Top-5 MA, nor even MA-19. Focusing in one to one comparisons there are only 11 instances in which some other algorithm delivers better results than MA-20 and in general terms the harder instances cases are in its favor. This analysis allows us to identify MA-20 as the statically dominant method over the Top-5 MA, and in conjunction with O-RMSE values, as the best among all MA configurations tested.

C. COMPARING DMAB+MA RESULTS

The combination of DMAB with MA versions as arms is referred as DMAB+MA. The experiments with this technique were ran over the same set of instances previously mentioned. Similarly to the MA version, DMAB parameter values were set using the irace utility [35] for automatized parameter tuning. Table 5 shows the values assigned to each parameter. The extreme value-based reward mechanism was suppressed by this tuning process, since it set the length of the reward register to $W = 1$.

TABLE 5. Parameter settings for the DMAB+MA algorithms.

Parameter	Value	Parameter	Value
Crossover probability $prob_c$	0.812	Length of the reward register W	1
Mutation probability $prob_m$	0.761	PH-test tolerance δ	0.299
Inversion probability $prob_i$	0.012	PH-test triggering umbral λ	33.472
Scaling factor C	7.138	Max. running time for DMAB s	600s

TABLE 6. Results for the Top-5 MA versions.

Rank	MA rank	Algorithm	Operator configuration	O-RMSE (%)	Average run time (s)	Time to local opt. (s)
1		DMAB	N/A	0.076	600.013	132.345
2	1	MA-19	S4_C1_M1_SS1_V1	1.097	264.526	118.634
3	2	MA-20	S4_C1_M2_SS1_V1	1.199	193.316	87.110
4	3	MA-68	S4_C1_M2_SS1_V2	1.218	273.548	153.704
5	4	MA-67	S4_C1_M1_SS1_V2	1.247	372.276	200.842
6	5	MA-85	S3_C1_M1_SS2_V2	2.210	104.436	42.244
7	6	MA-43	S4_C1_M1_SS2_V1	2.213	85.555	28.749
9	8	MA-34	S2_C2_M1_SS2_V1	2.231	85.851	29.976
23		MACH	N/A	2.869	7.725	N/A
59		GVNS	N/A	4.419	900.145	612.380

Table 6 depicts the updated O-RMSE values that now include the best solutions reached by DMAB+MA. This updating is necessary since O-RMSE is a metric relative to best-known results. For this comparison, DMAB+MA ranked first among all the considered methods. Since DMAB+MA improved the best-known results for 12 instances, the ranking of the MA versions was affected, MA-43 was displaced from the Top-5 MA by MA-85 and the algorithm from our preliminary work [15] fell to rank 8. DMAB+MA reduced the error rate by over 1%, while keeping a very competitive average time to reach its best-found solutions (see last column in Table 6). Even if its average run time is 600.013 seconds, this corresponds to the predefined stop condition (s) of DMAB+MA.

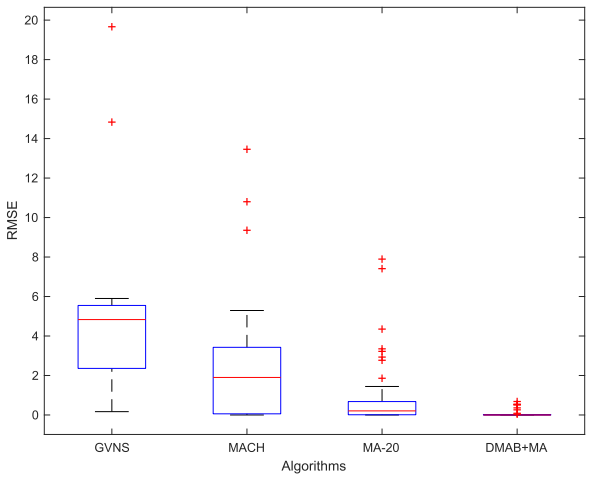

FIGURE 2. RMSE distribution for GVNS, MACH, the best MA independent version (MA-20) and DMAB+MA.

Fig. 2 shows RMSE values for GVNS, MACH, MA-20, and DMAB+MA over the instance set. Not only

TABLE 7. Statistical analysis for comparing the performance MACH, the MA-20 y DMAB+MA.

Instances	$\|V\|$	$\|E\|$	MA-20/ MACH	MA-20/ DMAB+MA	MACH / DMAB+MA
path100	100	99	−	−	⋆
path200	200	199	−	−	⋆
cycle100	100	100	−	−	⋆
cycle200	200	200	⋆	⋆	⋆
wheel100	100	198	⋆	⋆	⋆
wheel200	200	398	⋆	⋆	⋆
cyclePow100-2	100	200	⋆	⋆	⋆
cyclePow200-2	200	400	⋆	⋆	⋆
cyclePow100-10	100	1000	⋆	⋆	⋆
cyclePow200-10	200	2000	⋆	⋆	⋆
c9c9	81	162	⋆	⋆	⋆
c9k9	81	405	⋆	⋆	⋆
k9k9	81	648	⋆	⋆	⋆
p9c9	81	153	⋆	⋆	⋆
p9k9	81	396	⋆	⋆	⋆
p9p9	81	144	⋆	⋆	⋆
jgl011	11	49	+	⋆	−
ash85	85	219	+	−	−
curtis54	54	124	+	⋆	−
ibm32	32	90	+	−	−
will57	57	127	+	⋆	−
impcol_b	59	281	+	⋆	−
impcol_d	425	1267	+	−	−
nos4	100	247	+	⋆	−
nos6	675	1290	+	−	−
494_bus	494	586	⋆	−	−
662_bus	662	906	+	−	−
685_bus	685	1282	+	−	−
can_24	24	68	+	⋆	−
can_144	144	576	⋆	⋆	⋆
can_292	292	1124	⋆	⋆	⋆
can_445	445	1682	+	−	−
can_715	715	2975	+	−	−
bcspwr01	39	46	+	⋆	−
bcspwr02	49	59	+	⋆	−
bcspwr03	118	179	+	−	−
bcsstk01	48	176	+	−	−
bcsstk06	420	3720	+	−	−
dwt_503	503	2762	+	−	−
dwt_592	592	2256	+	−	−
+			21	0	0
−			3	17	22
⋆			16	23	18
Overall winner			MA-20	DMAB+MA	DMAB+MA

DMAB+MA reached an RMSE median equal to zero, but it also reduced the overall dispersion of the RMSE values. This means that, as expected, the DMAB+MA approach offers more robustness to instance variations. The higher RMSE value of DMAB+MA (0.681 for instance *662_bus*) almost halves the O-RMSE of MA-20 and most of its remaining values are under the median of MA-20.

Table 7 provides a comparison of the statistical significance of the results achieved by DMAB+MA, MA-20, and MACH. For instances of *cycle*, *path*, *wheel*, and *k-th power of cycle* topologies the three algorithms are typically tied, all of them reaching the optimal solutions in almost all cases. The only three instances where MA-20 is defeated by MACH become ties when the DMAB paradigm is incorporated. Both MA-20 and DMAB+MA represent a significant improvement with respect to MACH , specially when dealing with Harwell-Boeing instances. However, DMAB+MA outstanding results make it the undisputed overall winner, since no algorithm reached significant better solutions than it in any of the considered instances. For detailed results over each instance we refer the reader to Table 8 in Appendix .

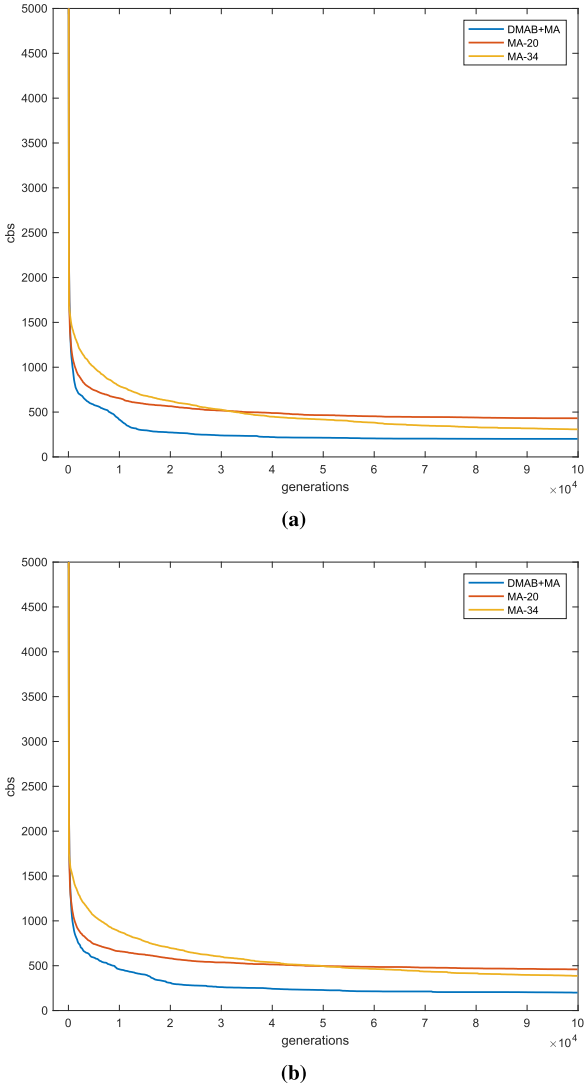

FIGURE 3. Convergence comparison among MA-34, MA-20, and DMAB+MA for two representative instances [(a) *path200* and (b) *cycle200*] along 100,000 generations.

V. DMAB DISCUSSION

Although in a general sense the MA versions improved the results of previous works such as MACH [18] and GVNS [17], they had problems to reach the known optimal Cbs value for some instances, particularly those of the *path*, *cycle*, and *wheel* topologies. It was intuited in [15] that premature convergence was responsible for this behaviour, due to an excessive selective pressure introduced by the $(\mu + \lambda)$ survival strategy. The introduction of (μ, λ) to the analyzed operator set, as well the combination of MA and DMAB were partially motivated to avoid this issue. Fig. 3 shows the average convergence of MA-34, MA-20 and DMAB+MA for instance *path200* along 100,000 generations. Compared with the MA versions, DMAB+MA has the ability to alternate among operators which prevents the loss of diversity in the population while still keeping the evolution towards fitter solutions.

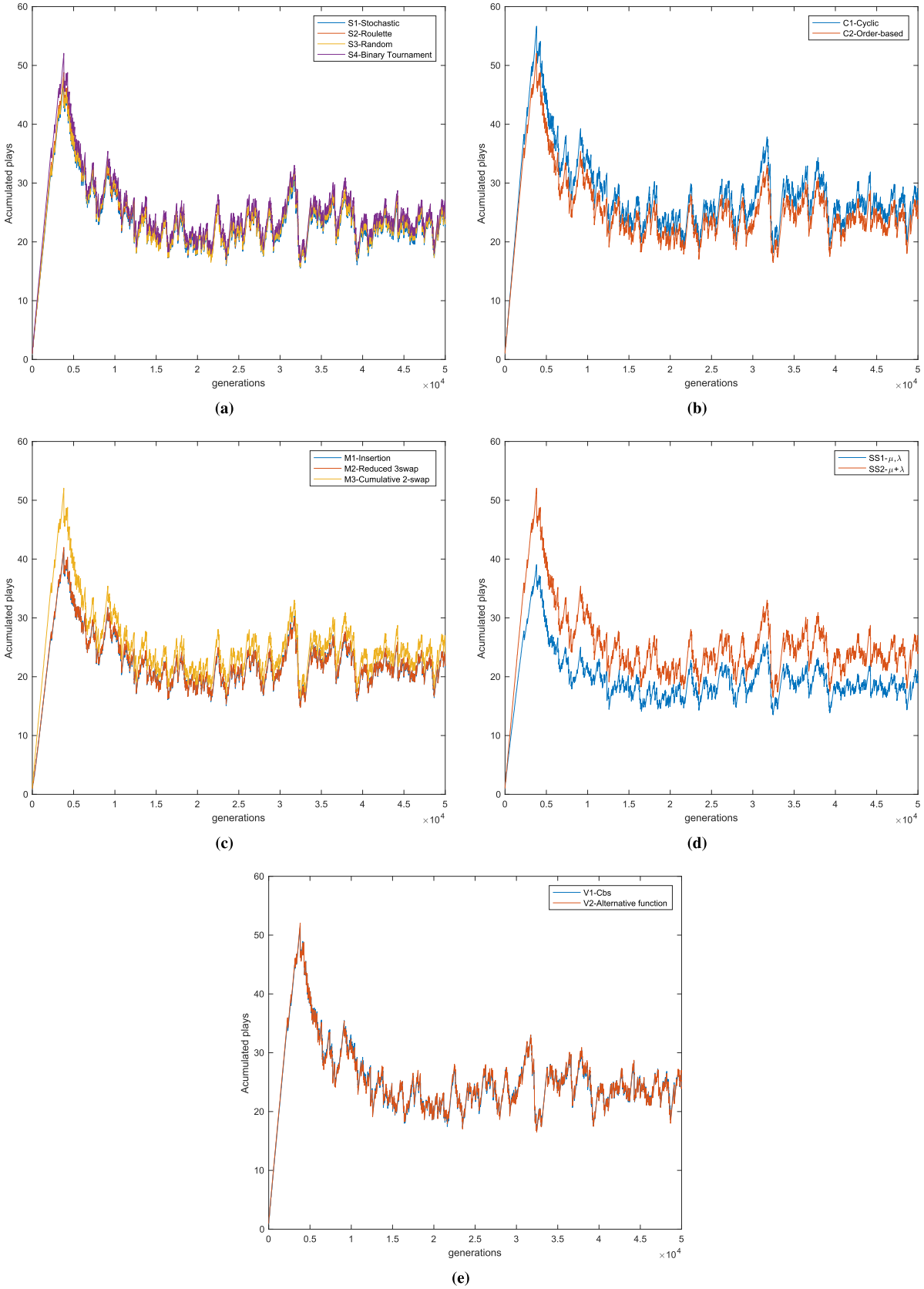

FIGURE 4. Average number of times that arms employing particular operators were played along 50,000 generations. (a) Selection operators. (b) Crossover operators. (c) Mutation operators. (d) Survival strategies. (e) Evaluation functions.

TABLE 8. Detailed performance comparison of MACH, MA-34, MA-20, and DMAB+MA.

					MACH				MA-20 (S4_C1_M2_SS1_V0)				MA-34 (S2_C2_M1_SS2_V1)				DMAB+MA			
Graph	$\|V\|$	$\|E\|$	d	UB/Opt^*	$Best$	Avg	Std	T	$Best$	Avg	Std	T	$Best$	Avg	Std	T	$Best$	Avg	Std	T
path100	100	99	0.020	99*	**99**	99.00	0.00	0.00	**99**	131.71	25.19	16.48	**99**	99.00	0.00	7.48	**99**	99.00	0.00	11.58
path200	200	199	0.010	199*	**199**	199.00	0.00	0.01	404	517.23	83.75	25.63	316	622.45	113.88	26.38	**199**	199.00	0.00	86.74
cycle100	100	100	0.020	100*	**100**	100.00	0.00	0.00	**100**	146.97	55.24	5.73	**100**	144.65	56.29	5.13	**100**	100.00	0.00	14.66
cycle200	200	200	0.010	200*	**200**	200.00	0.00	0.01	394	531.81	119.25	25.78	486	696.58	132.15	25.70	**200**	200.00	0.00	59.79
wheel100	100	198	0.040	2600*	**2600**	2600.00	0.00	0.01	**2600**	2646.71	47.96	9.10	**2600**	2633.42	45.94	11.71	**2600**	2600.00	0.00	2.88
wheel200	200	398	0.020	10200*	**10200**	10200.00	0.00	0.07	10400	10549.42	98.86	40.85	10484	10656.84	93.02	44.54	**10200**	10200.00	0.00	14.83
cyclePow100-2	100	200	0.040	300*	**300**	302.72	2.31	0.00	**300**	451.29	167.88	2.53	**300**	385.16	155.97	5.16	**300**	300.00	0.00	29.38
cyclePow200-2	200	400	0.020	600*	**600**	602.40	2.21	0.01	**600**	1086.26	324.60	29.67	**600**	1357.16	347.06	36.49	**600**	600.00	0.00	79.73
cyclePow100-10	100	1000	0.202	5500*	5598	5711.68	67.76	0.04	**5500**	5652.58	849.53	7.32	**5500**	5500.00	0.00	11.89	**5500**	5500.00	0.00	0.37
cyclePow200-10	200	2000	0.101	11000*	**11042**	11200.64	76.52	0.11	**11000**	15999.68	5252.26	2.99	**11000**	16734.19	5296.30	4.36	**11000**	11000.00	0.00	7.60
c9c9	81	162	0.050	873	991	1269.22	130.71	0.01	**873**	966.55	70.52	0.56	**873**	961.52	85.73	6.30	**873**	873.00	0.00	153.69
c9k9	81	405	0.125	7434	**1809**	1809.00	0.00	0.01	**1809**	1809.00	0.00	6.63	**1809**	1809.00	0.00	0.68	**1809**	1809.00	0.00	30.51
k9k9	81	648	0.200	8370	9424	9541.10	53.74	0.02	**8280**	8495.55	257.83	3.01	**8280**	8605.81	270.05	21.87	**8280**	8280.00	0.00	303.16
p9c9	81	153	0.047	7434	794	794.00	0.00	0.00	**745**	751.77	14.06	1.07	**745**	805.81	73.38	5.62	**745**	745.00	0.00	300.97
p9k9	81	396	0.122	7362	**1728**	1728.00	0.00	0.01	**1728**	1728.00	0.00	6.17	**1728**	1728.00	0.00	1.13	**1728**	1728.00	0.00	49.28
p9p9	81	144	0.044	720	944	1268.10	162.45	0.00	**516**	516.00	0.00	13.32	**516**	585.68	96.65	3.51	**516**	516.00	0.00	38.90
jgl011	11	49	0.891	147	142	142.00	0.00	0.00	**141**	141.00	0.00	0.00	**141**	141.00	0.00	0.00	**141**	141.00	0.00	0.00
ash85	85	219	0.061	4708	1214	1395.34	126.53	0.14	**913**	933.81	21.84	9.67	919	1036.58	89.64	22.89	**913**	913.00	0.00	48.88
curtis54	54	124	0.087	1705	448	622.54	86.12	0.03	**411**	411.00	0.00	8.00	**411**	422.90	20.66	8.54	**411**	411.00	0.00	62.34
ibm32	32	90	0.181	743	493	539.72	23.31	0.01	**405**	405.94	2.48	12.57	**405**	411.84	8.18	1.84	**405**	405.00	0.00	0.24
will57	57	127	0.080	1841	408	433.32	43.86	0.04	**335**	335.42	2.33	2.17	**335**	345.29	21.55	0.60	**335**	335.00	0.00	11.42
impcol_b	59	281	0.164	4215	2462	2841.06	214.84	0.07	**1822**	1822.00	0.00	0.23	**1822**	1829.74	9.90	0.16	**1822**	1822.00	0.00	0.72
impcol_d	425	1267	0.014	134935	23995	35083.34	5557.16	13.46	12960	16338.74	2484.91	231.19	12232	15932.90	3170.52	71.47	**12179**	12302.84	204.49	310.75
nos4	100	247	0.050	6237	1181	1448.04	264.25	0.06	**1031**	1031.00	0.00	2.23	**1031**	1031.00	0.00	6.03	**1031**	1031.00	0.00	55.43
nos6	675	1290	0.006	218010	35658	48573.80	4660.16	2.12	15443	16583.74	871.78	330.47	24230	32277.10	3088.31	47.12	**11702**	12877.87	823.97	339.73
494_bus	494	586	0.005	72517	4873	5701.84	410.44	9.95	5228	5515.58	185.15	180.39	9359	10784.65	993.45	35.79	**4496**	4975.90	220.70	307.36
662_bus	662	906	0.004	150169	12956	15354.50	1319.83	31.75	10956	12112.84	559.60	312.00	18762	24389.68	2411.77	40.40	**9238**	10475.84	672.78	283.74
685_bus	685	1282	0.005	219863	14300	17723.38	2083.32	37.46	15110	16320.00	658.46	319.46	22626	28396.52	2960.05	52.49	**11154**	11907.97	484.88	334.62
can___24	24	68	0.246	425	216	253.06	15.55	0.01	**182**	182.00	0.00	0.16	**182**	182.00	0.00	0.18	**182**	182.00	0.00	0.03
can__144	144	576	0.056	20881	2250	2258.94	6.11	0.02	**1776**	2587.61	749.96	8.25	**1776**	2339.00	787.03	9.48	**1776**	1776.00	0.00	49.30
can__292	292	1124	0.026	82333	23288	25903.86	1731.32	7.10	15139	15577.48	833.70	103.14	15763	18982.10	2148.92	75.81	**15115**	15126.94	7.92	283.30
can__445	445	1682	0.017	187543	41259	51235.26	4795.67	18.74	27865	28817.26	429.88	231.31	29799	34132.00	2492.48	86.83	**26704**	26789.32	54.07	397.47
can__715	715	2975	0.012	532525	91646	109431.58	10123.94	79.88	67316	70106.13	2146.42	449.09	72683	89289.74	8991.61	85.34	**61768**	65047.94	1440.59	309.54
bcspwr01	39	46	0.062	460	101	114.48	8.08	0.01	**98**	98.00	0.00	0.77	**98**	102.58	5.82	4.80	**98**	98.00	0.00	3.28
bcspwr02	49	59	0.050	737	158	179.40	19.47	0.02	**148**	148.00	0.00	1.14	**148**	151.94	5.93	10.53	**148**	148.00	0.00	4.23
bcspwr03	118	179	0.026	5325	766	927.02	70.56	0.26	663	666.58	1.71	24.71	664	713.19	53.72	14.67	**662**	663.42	0.76	321.30
bcsstk01	48	176	0.156	2156	1147	1336.48	107.99	0.02	**936**	938.71	3.78	20.08	**936**	954.45	13.43	21.32	**936**	936.00	0.00	4.24
bcsstk06	420	3720	0.042	391532	65017	83263.24	7489.16	30.87	53772	58924.52	4845.19	343.48	55140	67875.65	10377.12	177.82	**51847**	52092.23	138.83	294.99
dwt__503	503	2762	0.022	348012	55067	67128.50	6003.72	32.69	40998	42919.16	2932.07	344.80	43453	59031.48	7346.48	122.69	**37452**	37732.81	137.07	373.96
dwt__592	592	2256	0.013	334452	33659	44864.18	4826.16	44.03	29620	31218.29	1188.39	352.28	28856	42658.32	7073.52	84.30	**25076**	27280.55	1827.61	312.86

Fig. 4 plots the average accumulated number of times that an arm employing a particular operator was selected and played along 50,000 generations for a representative Harwell-Boeing instance (*dwt_503*). An arm is played only once per generation. The decline in the average accumulated plays indicates that arm statistics were restarted after the triggering of the PH-test whenever the best arm had changed. The graphics imply tendencies to play more often arms employing *cyclic* crossover (Fig. 4(b)), *cumulative 2-swap* mutation (Fig. 4(c)), and $(\mu + \lambda)$ survival strategy (Fig. 4(d)), while only a slight tendency for *binary tournament* (Fig. 4(a)), and no tendency for the evaluation function (Fig. 4(e)).

At the begging of the search process the DMAB mechanism is at an exploration phase, making an equal use of all available arms and showing no preference by any of them. As the search progresses and reward feedback is collected the DMAB slides into a phase of exploiting the best performing arms, and operator preferences become more remarkable towards the triggering of the PH-test. When a restart happens, all arms are tested, competing directly from the same search points. This allows the algorithm to choose an arm based not in reward estimations but in the actual performance. After the arm statistics are restarted the DMAB enters again in a exploration phase and the usage of operators becomes again more balanced. As an hyperheuristic approach, DMAB works in the domain of algorithms instead of directly over problem solutions. Its focus on balancing exploitation of good arms and exploration of underused arms, translated to the actual search space, results in providing good search directions, recognizing stagnation in local optimums and being able to escape from them.

VI. CONCLUSIONS AND FUTURE WORK

In this paper we contrasted the effectiveness of several MA configurations characterized by different sets of operators and an adaptive approach to online self alternation of operators.

We studied the behavior of 96 versions of a MA, both independently and when they are integrated into the DMAB paradigm. Results among the best five MA independent versions showed selection by *binary tournament*, *cyclic* crossover, and survival strategy (μ, λ) as more suitable operators for the CBSP with respect to their competitors, while mutation by *cyclic insertion* and *reduced 3-swap* were competitive with each other. It was also found that there are cases where the alternative evaluation scheme reported by Rodriguez-Tello *et al.* [19] can provide better guidance for the search process.

MA-20 was identified as the best stand-alone MA version, providing consistently and significantly better solutions for the general case of the CBSP than any other method previously reported. However, MA-20 results were further improved by the adaptive method DMAB+MA presenting more reduced error rates and a competitive amount of time needed to reach the best-know solutions.

Although the full factorial experimental design employed when comparing MA versions was informative, it demands high amounts of both time and computing power. If the operator set were to be extended it will be necessary to study how that new operator interactions will impact the performance. And as it was discussed in the previous section, the success of the MA versions depends not only on the combination of operators, but also on the instance topologies analyzed. While MA-20 results are outstanding against previously reported methods (including MA-34), there is no guarantee that it will be good enough for other new type of instances. The DMAB+MA approach proposed here is an alternative to address those issues. While the independent MA versions are limited to a fixed set of operators and potentially overspecialized for some instance topologies, DMAB+MA has all the operators available and the capacity to adapt in function of their success. Even if being able to exploit all of them comes at the expense of increasing computational demands and execution time, the results speak by themselves proving that DMAB+MA is a competitive approach that provides optimal/best-known solutions for all tested instances and to establish new better upper bounds for 12 of them.

These very promising results on the CBSP might encourage further work on the implementation of our DMAB+MA approach for efficiently solving other related graph embedding problems. It would also be worth investigating alternative implementations of the essential components of the DMAB paradigm, including: 1) Schemes to compute the value of the arms' reward [28]–[30], 2) Strategies for confidence estimation (other than UCB1) and regret-based feedback [32], and even 3) Arm encodings for managing heterogeneous low level heuristics for implementing a hyper-heuristic framework based on the DMAB paradigm [27].

APPENDIX A
DETAILED RESULTS

Table 8 presents detailed results of MACH, MA-20, MA-34, and DMAB+MA. The number of nodes $|V|$, edges $|E|$, density d, and upper-bound (UB) (or optimum value (Opt^*), when available) are listed for each instance in the set. For each algorithms we recorded the cost of the best found solution *Best* along 31 runs, the average cost *Avg* of the solutions, and its standard deviation *Std*, as well as the average execution time T.

ACKNOWLEDGMENT

The authors thankfully acknowledge the high performance computing resources (Neptuno cluster) and the technical assistance provided by CINVESTAV-Tamaulipas. The reviewers of the paper are greatly acknowledged for their constructive comments which have aided to improve the presentation of this paper.

REFERENCES

[1] 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.
[2] J. D. Ullman, *Computational Aspects of VLSI*. Rockville, MD, USA: Computer Science, 1984.
[3] L. H. Harper, "Optimal assignments of numbers to vertices," *J. Soc. Ind. Appl. Math.*, vol. 12, no. 1, pp. 131–135, 1964.
[4] B. Monien and H. Sudborough, "Embedding one interconnection network in another," in *Computational Graph Theory*, vol. 7, G. Tinhofer, E. Mayr, H. Noltemeier, and M. M. Syslo, Eds. Vienna, Austria: Springer-Verlag, 1990, pp. 257–282. doi: 10.1007/978-3-7091-9076-0_13.
[5] V. Liberatore, "Multicast scheduling for list requests," in *Proc. 21st Annu. Joint Conf. IEEE Comput. Commun. Soc.*, vol. 2. Jun. 2002, pp. 1129–1137.
[6] Y. Li and Y. Liang. (2017). "Compressed sensing in multi-hop large-scale wireless sensor networks based on routing topology tomography." [Online]. Available: https://arxiv.org/abs/1709.00604
[7] J. Yuan, "Cyclic arrangement of graphs," *Graph Theory Notes of New York*. New York, NY, USA: New York Academy of Sciences, 1995, pp. 6–10.
[8] F. R. K. Chung, "Labelings of graphs," in *Selected Topics in Graph Theory*, vol. 3, L. W. Beineke and R. J. Wilson, Eds. New York, NY, USA: Academic, 1988, ch. 7, pp. 151–168.
[9] P. Z. Chinn, J. Chvátalová, A. K. Dewdney, and N. E. Gibbs, "The bandwidth problem for graphs and matrices—A survey," *J. Graph Theory*, vol. 6, no. 3, pp. 223–254, 1982.
[10] D. Adolphson and T. C. Hu, "Optimal linear ordering," *SIAM J. Appl. Math.*, vol. 25, no. 3, pp. 403–423, 1973.
[11] 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.
[12] H. Jianxiu, "Cyclic bandwidth sum of graphs," *Appl. Math.-A J. Chin. Univ.*, vol. 16, no. 2, pp. 115–121, 2001.
[13] J. Díaz, J. Petit, and M. Serna, "A survey of graph layout problems," *ACM Comput. Surv.*, vol. 34, no. 3, pp. 313–356, 2002.
[14] P. C. B. Lam, W. C. Shiu, and W. H. Chan, "Characterization of graphs with equal bandwidth and cyclic bandwidth," *Discrete Math.*, vol. 242, no. 1, pp. 283–289, 2002.
[15] 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*, A. Auger, C. M. Fonseca, N. Lourenço, P. Machado, L. Paquete, and D. Whitley, Eds. Cham, Switzerland: Springer, 2018, pp. 82–94.
[16] Y. Chen and J. Yan, "A study on cyclic bandwidth sum," *J. Combinat. Optim.*, vol. 14, nos. 2–3, pp. 295–308, Oct. 2007.
[17] D. Satsangi, K. Srivastava, and Gursaran, "General variable neighbourhood search for cyclic bandwidth sum minimization problem," in *Proc. Students Conf. Eng. Syst.*, Piscataway, NJ, USA: IEEE Press, Mar. 2012, pp. 1–6.

[18] 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.
[19] 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.
[20] P. Jaccard, "The distribution of the flora in the alpine zone," *New Phytologist*, vol. 11, no. 2, pp. 37–50, 1912.
[21] T. L. Lai and H. Robbins, "Asymptotically efficient adaptive allocation rules," *Adv. Appl. Math.*, vol. 6, no. 1, pp. 4–22, Mar. 1985.
[22] 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, May 2002.
[23] 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*, T. Stützle, Ed. Berlin, Germany: Springer, 2009, pp. 176–190.
[24] J. Belluz, M. Gaudesi, G. Squillero, and A. Tonda, "Operator selection using improved dynamic multi-armed bandit," in *Proc. Annu. Conf. Genetic Evol. Comput. (GECCO)*, New York, NY, USA 2015, pp. 1311–1317.
[25] A. Fialho, L. Da Costa, M. Schoenauer, and M. Sebag, "Analyzing bandit-based adaptive operator selection mechanisms," *Ann. Math. Artif. Intell.*, vol. 60, no. 1, pp. 25–64, Oct. 2010.
[26] M. Gagliolo and J. Schmidhuber, "Algorithm selection as a bandit problem with unbounded losses," in *Proc. 4th Int. Conf. Learn. Intell. Optim. (LION)*, Berlin, Germany: Springer-Verlag, 2010, pp. 82–96.
[27] G. Guizzo, G. Fritsche, S. Vergilio, and A. Pozo, "A hyper-heuristic for the multi-objective integration and test order problem," in *Proc. Annu. Conf. Genetic Evol. Comput. (GECCOs)*, New York, NY, USA, 2015, pp. 1343–1350. [Online]. Available: http://doi.acm.org/10.1145/2739480.2754725
[28] L. Davis, "Adapting operator probabilities in genetic algorithms," in *Proc. 3rd Int. Conf. Genetic Algorithms*. San Francisco, CA, USA: Morgan Kaufmann, 1989, pp. 61–69.
[29] A. Tuson and P. Ross, "Adapting operator settings in genetic algorithms," *Evol. Comput.*, vol. 6, no. 2, pp. 161–184, Jun. 1998.
[30] H. J. C. Barbosa and A. M. Sa, "On adaptive operator probabilities in real coded genetic algorithms," in *Proc. Conf. Chilean Comput. Sci. Soc.*, 2000, pp. 1–6.
[31] A. Fialho, L. Da Costa, M. Schoenauer, and M. Sebag, "Extreme value based adaptive operator selection," in *Parallel Problem Solving from Nature*, G. Rudolph, T. Jansen, N. Beume, S. Lucas, and C. Poloni, Eds. Berlin, Germany:: Springer, 2008, pp. 175–184.
[32] L. DaCosta, A. Fialho, M. Schoenauer, and M. Sebag, "Adaptive operator selection with dynamic multi-armed bandits," in *Proc. 10th Annu. Conf. Genetic Evol. Comput. (GECCO)*, New York, NY, USA, 2008, pp. 913–920.
[33] D. V. Hinkley, "Inference about the change-point from cumulative sum tests," *Biometrika*, vol. 53, no. 3, pp. 509–523, 1971.
[34] N. R. Sabar, M. Ayob, G. Kendall, and R. Qu, "A dynamic multiarmed bandit-gene expression programming hyper-heuristic for combinatorial optimization problems," *IEEE Trans. Cybern.*, vol. 45, no. 2, pp. 217–228, Feb. 2015.
[35] M. López-Ibáñez, J. Dubois-Lacoste, L. P. Caceres, T. Stützle, and M. Birattari, "The irace package: Iterated racing for automatic algorithm configuration," *Oper. Res. Perspect.*, vol. 3, pp. 43–58, Jan. 2016.

EDUARDO RODRIGUEZ-TELLO (M'18) was born in Mexico City, Mexico, in 1973. He received the M.S. degree in computer science from 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 CINVESTAV Tamaulipas, Ciudad Victoria, Mexico. He has authored or co-authored a book and over 40 technical papers and book chapters. His publications currently report over 540 citations in Google Scholar with an H-index of 12. His current research interests include evolutionary computation as well as the design and implementation of effective metaheuristic algorithms for solving large-scale combinatorial optimization problems arising in various application areas, such as bioinformatics, graph theory, and software engineering.

VALENTINA NARVAEZ-TERAN received the B.S. degree in information technologies engineer 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 CINVESTAV Tamaulipas, Ciudad Victoria, Mexico.

In 2016, she did an Internship at University Rey Juan Carlos, Madrid, Spain. Her research interests include combinatorial optimization, graph embedding problems, and metaheuristics design.

FRÉDÉRIC LARDEUX received the M.S. and Ph.D. degrees in computer science from the University of Angers, France, in 2002 and 2005, respectively, where he has been an Assistant Professor with the Laboratoire d'étude et de Recherche en Informatique d'Angers, since 2006.

His research interests include evolutionary computation, autonomous search, model transformations, and logical analysis of data.