COEBA: A Coevolutionary Bat Algorithm for Discrete Evolutionary Multitasking

Eneko Osaba[1]([✉]), Javier Del Ser[1,2], Xin-She Yang[3], Andres Iglesias[4,5], and Akemi Galvez[4,5]

[1] TECNALIA, Basque Research and Technology Alliance (BRTA), 48160 Derio, Spain
`eneko.osaba@tecnalia.com`
[2] University of the Basque Country (UPV/EHU), 48013 Bilbao, Spain
[3] Middlesex University London, The Burroughs, London NW4 4BT, UK
[4] Universidad de Cantabria, 39005 Santander, Spain
[5] Toho University, Funabashi, Japan

Abstract. Multitasking optimization is an emerging research field which has attracted lot of attention in the scientific community. The main purpose of this paradigm is how to solve multiple optimization problems or tasks simultaneously by conducting a single search process. The main catalyst for reaching this objective is to exploit possible synergies and complementarities among the tasks to be optimized, helping each other by virtue of the transfer of knowledge among them (thereby being referred to as Transfer Optimization). In this context, Evolutionary Multitasking addresses Transfer Optimization problems by resorting to concepts from Evolutionary Computation for simultaneous solving the tasks at hand. This work contributes to this trend by proposing a novel algorithmic scheme for dealing with multitasking environments. The proposed approach, coined as Coevolutionary Bat Algorithm, finds its inspiration in concepts from both co-evolutionary strategies and the metaheuristic Bat Algorithm. We compare the performance of our proposed method with that of its Multifactorial Evolutionary Algorithm counterpart over 15 different multitasking setups, composed by eight reference instances of the discrete Traveling Salesman Problem. The experimentation and results stemming therefrom support the main hypothesis of this study: the proposed Coevolutionary Bat Algorithm is a promising meta-heuristic for solving Evolutionary Multitasking scenarios.

Keywords: Transfer Optimization · Evolutionary Multitasking · Bat Algorithm · Multifactorial Optimization · Traveling Salesman Problem

1 Introduction

By using as its inspiration concepts from Transfer Learning [34] and Multitask Learning [4], Transfer Optimization is an incipient knowledge field, which has

congregated an active scientific community in recent years [28]. The principal idea behind this field is to exploit what has been learned through the optimization of one specific problem or task, when tackling of another related or unrelated optimization task. Due to its relative youth, Transfer Optimization has not been studied as deeply as other research areas. It has not been until these last years when the transferability of knowledge among tasks has become a priority among researchers from the Evolutionary Computation arena.

Within the Transfer Optimization paradigm, three separated categories can be identified: *sequential transfer*, *multitasking* and *multiform optimization*. The first of these classes refers to those situations in which tasks are faced sequentially, assuming that for solving a new problem/instance, the knowledge collected when solving previous tasks is used as external information [11]. The second of these categories (*Multitasking*) deals with different optimization tasks simultaneously by dynamically scrutinizing existing complementarities and synergies among them [16,39]. Finally, *multiform optimization* aims at solving a single problem by resorting to different alternative problem formulations, which are optimized simultaneously. In all these categories, there is a clear consensus in the community on the capital importance of the correlation among the tasks to be solved for positively capitalizing on the transfer of knowledge over the search [17].

Among the three divisions pointed out above, *multitasking* is the one that has arguably grasped most attention by the community. The study presented in this manuscript is focused on this specific category. Specifically, we focus on multitasking optimization through the perspective of Evolutionary Multitasking (EM, [27]). In short, EM tackles the simultaneous optimization of several optimization tasks by relying on concepts and methods from Evolutionary Computation [1,8]. In the last years, a particular flavor of EM grounded on the so-called Multifactorial Optimization strategy (MFO, [17]) has shown a superior efficiency when dealing with different environments involving several continuous, discrete, single-optimization and multi-objective optimization problems and tasks [14,18,38,43]. The majority of the literature related to this area is focused on a solver belonging to this flavor: the Multifactorial Evolutionary Algorithm (MFEA, [17]). Unfortunately, alternatives to MFEA still remain scarce to date.

Bearing this in mind, the research work presented in what follows revolves on a novel EM meta-heuristic algorithm that adopts the Bat Algorithm (BA, [41]) at its core. Specifically, we present a Coevolutionary Bat Algorithm (COEBA) for discrete evolutionary multitasking. Through this proposal, we take a step further over the state of the art by elaborating on a new research direction in two different directions. On the one hand, we contribute to the EM area by introducing a new efficient meta-heuristic scheme. It is important to point out here that, unlike most articles published so far around EM, COEBA does not find its inspirational source in the MFO paradigm. On the other hand, COEBA is the first attempt at using BA for Transfer Optimization.

It is also relevant to underscore here that the experimentation carried out in this paper considers a less studied discrete environment comprising different

instances of the Traveling Salesman Problem (TSP, [21]). Concretely, we assess the performance of the proposed COEBA by comparing its performance to that obtained by MFEA. Our main purpose with this performance comparison is to elucidate that COEBA embodies a promising alternative to deal with EM scenarios. To this end, we have chosen 8 different TSP instances, giving rise to 15 multi-tasking environments with varying degrees of phenotypical relationship.

The rest of the paper is organized as follows. Section 2 introduces the background related to both Evolutionary Multitasking and the Bat Algorithm. Next, Sect. 3 exposes in detail the main features of the proposed COEBA. The experimentation setup, analysis and discussion of the results are given in Sect. 4. The study ends in Sect. 5 with conclusions and future research directions.

2   Background

This section is dedicated to providing a brief background on Evolutionary Multitasking (Sect. 2.1) and the Bat Algorithm (Sect. 2.2).

2.1   Evolutionary Multitasking

In recent years, EM has arisen as a promising paradigm for facing simultaneous optimization tasks. There are two main features that motivated the first formulation of EM. The first one is the parallelism inherent to the population of individuals, which eases the management of diverse concurrent optimization tasks faced simultaneously. Thanks to this feature, latent synergies between tasks can be automatically harnessed during the solving process [28]. The second feature is the continuous transfer of genetic material between the individuals, which allows all tasks to benefit from each other, even for those that are not strongly correlated with the rest of the pool [17].

It is widely accepted that the concept of EM was only materialized through the vision of the MFO until late 2017 [6]. Today, this nascent research stream is receiving interesting contributions in terms of new algorithmic schemes, such as the Coevolutionary Multitasking scheme proposed in [5], or the multitasking multi-swarm optimization described in [37]. Additional alternatives to MFEA have also been proposed, such as the multifactorial brain storm optimization algorithm presented in [45], the Multifactorial Differential Evolution in [10] or the hybrid particle swarm optimization-firefly algorithm introduced in [40]. Despite these recently proposed methods, MFO and its related MFEA have monopolized the research activity around this field since its inception. In fact, the authors of MFEA have recently introduced an adaptive variant of MFEA, coined as MFEA-II, thereby eliciting the momentum played by this algorithm in the field [2].

Going into mathematical details, we can formulate EM as an environment in which K tasks or problems should be optimized in a simultaneous fashion. This environment is characterized by the existence of as many search spaces as tasks. Thus, for the k-th task, its objective function T_k is characterized as $f_k : \Omega_k \to \mathbb{R}$, where Ω_k represents the search space of T_k. Let us assume that

all tasks are minimization problems, so that the main objective of EM is to find a set of solutions $\{\mathbf{x}_1, \ldots, \mathbf{x}_K\}$ such that $\mathbf{x}_k = \arg\min_{\mathbf{x}\in\Omega_k} f_k(\mathbf{x})$. A crucial aspect to properly understand EM is that all individuals $\mathbf{x}_p$ in the population P to be evolved belong to a unified search space Ω^U that relates to Ω_1 to Ω_K by means of a encoding/decoding mapping functions $\xi_k : \Omega_k \mapsto \Omega^U$. Therefore, each individual $\mathbf{x}_p \in \Omega^U$ in P can be decoded $(\xi_k^{-1}(\mathbf{x}_p))$ to represent a task-specific solution $\mathbf{x}_{p,k}$ for each of the K tasks. Shifting our attention on MFO and MFEA, four different definitions are associated with each individual $\mathbf{x}_p$ of the population P: Factorial Cost, Factorial Rank, Scalar Fitness and Skill Factor. With the intention accommodating this work to the extension requisites, we refer interested readers to [2,17] for additional deeper details on how these definitions are exploited during the search over the unified space Ω^U.

Several significant works have been recently published around EM and MFO. In [44], authors present an influential application of the MFEA to different discrete problems. This paper also introduces the discrete unified encoding, used as a reference in subsequent works. A related study is [47], where MFEA was applied to the Vehicle Routing Problem. Gong et al. presented in [14] and improved version of the MFEA, endowing the algorithm with a dynamic resource allocating strategy. An interesting discrete MFEA has been also developed in [38] for the composition of semantic web services. Gupta et al. presented in [18] a multi-objective variant of MFEA, giving evidence of its efficiency on a real-world manufacturing process design problem. Finally, the work in [43] follows a similar strategy by enhancing MFEA with the incorporation of opposition-based learning. Further theoretical studies on EM and MFEO can be found in [22,46].

2.2 Bat Algorithm

BA is a nature-inspired metaheuristic based on the echolocation system of bats. In the nature, bats emit ultrasonic pulses to the surrounding environment with navigation and hunting purposes. After the emission of these pulses, bats listen to the echoes, and based on them they can locate themselves and also identify and locate preys and obstacles. Besides that, each bat is able to find the most "nutritious" areas performing an individual search, or moving towards a "nutritious" location previously found by any other component of the swarm. It is important to mention that some rules have to be previously established with the aim of making an appropriate adaptation [41]:

1. All bats use echolocation to detect the distance, and they are assumed to be able to distinguish between an obstacle and a prey.
2. All bats fly randomly at speed v_i and position $\mathbf{x}_i$, emitting pulses with a fixed frequency f_{min}, varying wavelength λ and loudness A_i to search for a prey. In this idealized rule, it is assumed that every bat can adjust in an automatic way the frequency (or wavelength) of the pulses, emitted at a rate $r \in [0, 1]$. This automatic adjustment depends on the proximity of the targeted prey.
3. In the real world, the bats' emissions loudness can vary in many different ways. Nevertheless, we assume that this loudness can vary from a large positive A_0 to a minimum constant value A_{min}.

Since its proposal, BA has emerged as one of the most successful meta-heuristic solvers. It has been applied to a manifold of problems such as logistic [32], industry [24], or medicine [19]. The literature behind BA is huge and diverse, as manifested by comprehensive surveys on practical applications of BA [12,42].

3   Coevolutionary Bat Algorithm for Multitasking

Following concepts previously embraced by other alternatives in the literature [5], one of the main characteristics of the designed COEBA is its multi-population nature. By this we mean that COEBA is a method composed by a defined number of populations, or *demes* [25], comprised by the same number of individuals. More specifically, the number of groups is equal to K, i.e. the number of tasks to be optimized. Additionally, each of the K subpopulations concentrates on solving a specific task T_k. This means that bats corresponding to the k-th deme are only evaluated on task T_k.

As in MFEA, a unified representation Ω^U is used for encoding individuals. However, the most innovative aspect of COEBA is that each subpopulation has its own search space. This involves a slight size readjustment when different demes exchange individuals among them. We will hereafter use the TSP to show this size readjustment problem. Hence, we denote the size of each problem T_k (i.e. the number of *cities*) as D_k. Let us assume that individual $\mathbf{x}_i$ is encoded as a permutation of the integer set $\{1, 2, \ldots, D_k\}$. In this way, when $\mathbf{x}_p^k \in \Omega_k$ is migrated to a subpopulation in which the size of task T_k' to be solved is $D_k' < D_k$, only integers lower than D_k are considered, thus reducing the phenotype of the individual. These integers maintain the same order as in $\mathbf{x}_p^k$. The reverse procedure applies if $D_k' > D_k$. In that case, and considering that each time an individual $\mathbf{x}_p^k$ is transferred to a deme it replaces an alternative bat $\mathbf{x}_p^{k'}$, all integers between D_k and D_k' are inserted in $\mathbf{x}_p^k$ in the same positions as in $\mathbf{x}_p^{k'}$. This multiple search space strategy enhances the exploitation of the search over the demes, making the movement operators more effective.

With all these considerations in mind, Algorithm 1 shows the pseudocode of the designed COEBA. As can be seen, in the initialization process a number X of individuals are randomly generated. After initialization, each individual is evaluated over all K tasks. Then, within an iterative process, each subpopulation is built by choosing the top X/K individuals for the corresponding task (the same bat can be selected by different tasks). Once demes are composed, each one is evolved independently by following the concepts of the discrete version of the BA [32]. To be more concise, the distance between two different bats is measured by means of the Hamming Distance, namely, the number of non-corresponding elements in the sequence. Furthermore, the *inclination* mechanism is also used [31]. Thanks to this feature (lines 10–14 in Algorithm 1), the method intelligently selects the movement function suited for each bat at every iteration, depending on its specific situation regarding the leading bat of the swarm. As is shown in Algorithm 1, *2-opt* and *insertions* are used as movement functions.

Algorithm 1: Pseudocode of the proposed COEBA

---

1 Randomly generate an initial population of X bats
2 **for** *each bat x_i in the population* **do**
3 $\quad$ Initialize the pulse rate r_i, velocity v_i and loudness A_i

4 Evaluate each of the individual for all the K optimization tasks
5 Build the K number of subpopulations
6 **while** *termination criterion not reached* **do**
7 $\quad$ **for** *each population k* **do**
8 $\quad\quad$ **for** *each bat x_i in the subpopulation* **do**
9 $\quad\quad\quad$ Generate new solution
10 $\quad\quad\quad$ **if** $v_i^t < n/2$ **then**
11 $\quad\quad\quad\quad$ $x_i \leftarrow 2 - opt(x_i^{t-1}, v_i^t)$
12 $\quad\quad\quad$ **else**
13 $\quad\quad\quad\quad$ $x_i \leftarrow insertion(x_i^{t-1}, v_i^t)$
14 $\quad\quad\quad$ **if** $rand > r_i$ **then**
15 $\quad\quad\quad\quad$ Select one solution among the best ones
16 $\quad\quad\quad\quad$ Generate a new bat by selecting the best neighbor of the chosen bat
17 $\quad\quad\quad$ **if** $rand < A_i$ *and* $f(x_i) < f(x_*)$ **then**
18 $\quad\quad\quad\quad$ Accept the new solution
19 $\quad\quad\quad\quad$ Increase r_i and reduce A_i

20 $\quad\quad$ **if** *iteration is multiple of migr* **then**
21 $\quad\quad\quad$ Two random individuals are migrated from k to another randomly selected subpopulation

22 Return the best individual in P for each task T_k

Moreover, every *migr* iterations, each group transfers two individuals to a randomly selected population. These two bats are selected by following this criterion: the first one is selected uniformly at random among the best 10 individuals of the population, while the second one is drawn from the complete subpopulation. These two individuals substitute two randomly chosen bats, not considering the 10 best ones of the deme where the replacement is done. Finally, COEBA finishes its execution after I iterations, returning as its solution the best bat of each subpopulation. Any other stopping criterion can be adopted with no further consequences to the design of the algorithm.

4   Experimentation and Results

To assess the performance of the designed COEBA solver, an extensive experimentation has been carried out, which is described in depth in this section. As such, Subsect. 4.1 elaborates on the group of TSP instances used in the experiments, whereas Subsect. 4.2 details the experimentation setup. Finally, the obtained results are analyzed and critically examined in Subsect. 4.3.

4.1 Benchmark Problems

As introduced in Sect. 1, the experiments performed in this work consider the TSP as their benchmark problem to be optimized simultaneously. Readers interested on the formulation and theoretical aspects of this classical problem are referred to [3] or [26]. Arguably, TSP has become one of the most often used problems for performance analysis of discrete optimization algorithms. A plethora of meta-heuristic solvers have been applied to the TSP, or to any of its variants, from traditional techniques such as the Genetic Algorithm [15], Ant Colony Optimization [9] or Tabu Search [13], to modern discrete solvers such as Firefly Algorithm [20], Cuckoo Search [33], or the Water Cycle Algorithm [30]. Before proceeding further, it is important to bear in mind that the goal of the experiments is not to reach the optimal solution of the TSP instances under consideration, but to statistically compare the performance of both MFEA and COEBA when using the same instances and evaluation conditions.

This being said, the performance of COEBA and its counterpart MFEA has been measured over 8 TSP instances, which are combined to yield 15 different test scenarios. All instances have been retrieved from the TSPLIB repository [36]. Specifically, the first 8 instances of the Padberg/Rinaldi benchmark have been employed: `pr76`, `pr107`, `pr124`, `pr136`, `pr144`, `pr152`, `pr226`, and `pr264`.

4.2 Experimental Setup

For the sake of fairness in the comparisons, similar parameters and operators have been used for both MFEA and COEBA. This way, we can objectively conclude which solver reaches better outcomes using similar evaluation conditions. To ensure the replicability of this study, parameters employed for the implemented algorithms are depicted in Table 1. For this parameter setting, not only

Table 1. Parameter values set for MFEA and COEBA.

MFEA		COEBA	
Parameter	Value	Parameter	Value
Population size	200	Population size	200
Crossover Function	Order Crossover [7]	Short movement function	2-opt [23]
Mutation Function	2-opt [23]	Short movement function	Insertion [31]
Crossover Prob.	0.9	Initial A_i^0	Random number in [0.8,1.0]
Mutation Prob.	0.1	Initial r_i^0	Random number in [0.0,0.4]
$migr$	100	$migr$	100
		α & γ	0.98

works focused on MFEA and BA have been considered [17,42,44], but also good practices reported in the community for tackling routing problems [29]. In addition, all bats are initialized uniformly at random. As the termination criterion, each algorithm is stopped after $I = 500 \cdot 10^3$ objective function evaluations.

As mentioned before, 15 different test scenarios have been built for the experimentation. Each of these multitasking configurations implies that both COEBA and MFEA should face the resolution of all the tasks assigned to that scenario simultaneously. Among these test cases, 10 of them are composed by 4 TSP instances, 4 scenarios are comprised by 6 TSP instances, and the last one includes all the 8 instances. Table 2 summarizes all the considered configurations. The main rationale for building these tests scenarios is twofold: i) to reach conclusions over a diverse and heterogeneous set of multitasking scenarios, involving each TSP instance in exactly the same number of cases, and ii) to exploit the possible genetic complementarities of the instances.

Table 2. Summary of the 15 tests cases built for the experimentation.

Test case	pr76	pr107	pr124	pr136	pr144	pr152	pr226	pr264
Test_Case_4_1	×	×	×	×				
Test_Case_4_2					×	×	×	×
Test_Case_4_3	×	×					×	×
Test_Case_4_4			×	×	×	×		
Test_Case_4_5	×		×	×			×	
Test_Case_4_6		×			×	×		×
Test_Case_4_7	×	×		×		×		
Test_Case_4_8			×		×		×	×
Test_Case_4_9	×			×	×		×	
Test_Case_4_10		×	×			×		×
Test_Case_6_1	×	×	×	×	×	×		
Test_Case_6_2			×	×	×	×	×	×
Test_Case_6_3	×	×			×	×	×	×
Test_Case_6_4	×	×		×	×		×	×
Test_Case_8	×	×	×	×	×	×	×	×

Finally, all tests have been carried out on an Intel Xeon E5 – 2650 v3 computer, with 2.30 GHz and a RAM of 32 GB. Moreover, each test case has been run 20 times to account for the statistical significance of performance gaps encountered during the experimentation.

4.3   Results and Discussion

Table 3 depicts the comparisons in the results reached by COEBA and MFEA. Due to lack of space, we omit all the average outcomes for each test case. Instead,

we show graphically the comparison using two colored circles. A green circle
(●) implies that COEBA has performed better than MFEA in terms of fitness
average. On the contrary, a red circle (●) means that MFEA has achieved better
results on average. Using *Test_Case_4_3* as an example, and considering Table 2,
we observe that COEBA performs better in `pr76`, `pr226`, and `pr264`, while
MFEA is better in `pr107`. Thus, analyzing the content of the Table 3, we conclude
that COEBA clearly outperforms MFEA over these EM scenarios, being superior
to MFEA in all but 4 TSP instances evolved jointly. It is also crucial to highlight
that COEBA obtains better outcomes in all the eight instances evolved jointly
in *Test_Case_8*.

Table 3. Comparison of the results for the 15 tests cases built for the experimentation.
(●) means COEBA outperforms MFEA. (●) means MFEA performs better.

Test Case	COEBA vs. MFEA comparison
Test_Case_4_1	●-●-●-●
Test_Case_4_2	●-●-●-●
Test_Case_4_3	●-●-●-●
Test_Case_4_4	●-●-●-●
Test_Case_4_5	●-●-●-●
Test_Case_4_6	●-●-●-●
Test_Case_4_7	●-●-●-●
Test_Case_4_8	●-●-●-●
Test_Case_4_9	●-●-●-●
Test_Case_4_10	●-●-●-●
Test_Case_6_1	●-●-●-●-●-●
Test_Case_6_2	●-●-●-●-●-●
Test_Case_6_3	●-●-●-●-●-●
Test_Case_6_4	●-●-●-●-●-●
Test_Case_8	●-●-●-●-●-●-●-●

For extending the coverage and insights provided by this experimentation, we
depict in Table 4 the outcomes obtained by COEBA and MFEA for the 8 TSP
instances that compose *Test_Case_8*. We show the average, best and standard
deviation of results for each instance. Furthermore, we also provide the best
known optima reported for each TSP instance in the literature. These results
confirm that the proposed COEBA is a promising meta-heuristic for solving EM
environments, outperforming MFEA in terms of both average and best outcomes
in this context. Even though it is not the goal of this work, it is also relevant to
note that the difference between the optimal outcomes and the average results
obtained by COEBA ranges between 0.4% and 5.6%, thereby showing that our
proposal not only performs competitively for multitasking environments, but
also gets close to optimality of the tasks under consideration.

Table 4. Results (best/average/standard deviation of the fitness over 20 runs) obtained by COEBA and MFEA for the 8 instances in the *Test_Case*_8, and results of the Wilcoxon Rank-Sum test. Best results between COEBA and MFEA achieved over each TSP instances are highlighted in bold.

Method	pr76	pr107	pr124	pr136	pr144	pr152	pr226	pr264
	108602.4	**44927.3**	**59380.8**	**99741.1**	**59045.5**	**74819.1**	**81425.7**	**51924.3**
COEBA	108234.0	44610.0	59087.0	99741.1	58771.0	74000.0	81048.0	51079.0
	402.54	242.27	226.89	534.30	244.37	420.50	248.55	458.87
	113116.5	47110.5	62104.2	106729.3	62179.2	76117.3	84586.3	54031.7
MFEA	111073.0	46052.0	61419.0	104998.0	60534.0	74294.0	82320.0	52728.0
	2355.08	858.99	601.06	1461.53	1770.25	1756.13	6065.24	3489.31
Optima	108159.0	44303.0	59030.0	96772.0	58537.0	73682.0	80369.0	49135.0
Wilcoxon test	🟢	🟢	🟢	🟢	🟢	🟢	🟢	🟢

In order to buttress our conclusions with the statistical significance of these identified gaps, the Wilcoxon Rank-Sum test has been applied, rendering the results depicted in the last row of Table 4. The confidence interval has been set to 95%. We have compared the outcomes obtained for all the 8 TSP instances separately. Accordingly, the last row of Table 4 represent the outcomes of these statistical tests. Specifically, a green circle (🟢) means that COEBA outperforms MFEA with statistical significance. On the contrary, the red circle (🔴) would have indicated the non-existence of evidences for ensuring the statistical significance of a gap between MFEA and COEBA. As can be seen in this table, Wilcoxon Rank-Sum test confirms that COEBA significantly outperforms MFEA in all the 8 instances embedded in this test scenario. The obtained average z-value is -2.68, with an average p-value equal to 0.00888. Considering that the critical z_c value is -1.64, and because $-2.68 < -1.64$ and $0.00888 < 0.05$, these outcomes support the significance of the performance differences at 95% confidence level. Thus, the difference is significant at this confidence level, thereby concluding that the COEBA is statistically better than MFEA for this test scenario.

5 Conclusions and Future Work

This manuscript has elaborated on the design, implementation and validation of a novel approach for solving evolutionary multitasking environments, wherein tasks are optimization problems. For reaching this goal, we have introduced the Coevolutionary Bat Algorithm (COEBA), which finds its source of inspiration from the concepts of evolutionary co-evolution and the discrete adaptation of the Bat Algorithm. A subpopulation is devoted for the optimization of each problem, with a migration policy that allow exchanging genotype information and exploiting synergies among problems. For showcasing the application of the proposed multitasking approach, an experimental setup has been devised embracing instances of the Traveling Salesman Problem as benchmark problems to be jointly solved. We have compared the outcomes attained by COEBA with the ones furnished by Multifactorial Evolutionary Algorithm (MFEA) over 15

different multitasking test cases. The results validate our hypothesis: COEBA is a promising meta-heuristic for addressing multitasking scenarios.

Several research lines have been planned to gain insight beyond the findings reported in this study. In the short term, we will gauge the scalability of COEBA by analyzing its performance and computational efficiency when simultaneously solving test cases comprising TSP instances of larger dimensionality. We also plan to design additional search mechanisms (such as alternative migration strategies), all targeted at reinforcing the transfer of knowledge among related tasks (*positive transfer*), and lowering the genotype exchange among those tasks that are not related to each other (correspondingly, *negative transfer*). In the longer term, we will explore the application of COEBA to problems stemming from other research fields [35] with discrete optimization problems at their core, such as community detection in social networks.

References

1. Bäck, T., Fogel, D.B., Michalewicz, Z.: Handbook of Evolutionary Computation. CRC Press (1997)
2. Bali, K.K., Ong, Y.S., Gupta, A., Tan, P.S.: Multifactorial evolutionary algorithm with online transfer parameter estimation: MFEA-II. IEEE Trans. Evol. Comput. **24**(1), 69–83 (2020)
3. Bellmore, M., Nemhauser, G.L.: The traveling salesman problem: a survey. Oper. Res. **16**(3), 538–558 (1968)
4. Caruana, R.: Multitask learning. Mach. Learn. **28**(1), 41–75 (1997)
5. Cheng, M.Y., Gupta, A., Ong, Y.S., Ni, Z.W.: Coevolutionary multitasking for concurrent global optimization: with case studies in complex engineering design. Eng. Appl. Artif. Intell. **64**, 13–24 (2017)
6. Da, B., et al.: Evolutionary multitasking for single-objective continuous optimization: benchmark problems, performance metric, and baseline results. arXiv preprint arXiv:1706.03470 (2017)
7. Davis, L.: Job shop scheduling with genetic algorithms. In: International Conference on Genetic Algorithms and their Applications, vol. 140 (1985)
8. Ser, J., et al.: Bio-inspired computation: where we stand and what's next. Swarm Evol. Comput. **48**, 220–250 (2019)
9. Dorigo, M., Gambardella, L.M.: Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Trans. Evol. Comput. **1**(1), 53–66 (1997)
10. Feng, L., et al.: An empirical study of multifactorial PSO and multifactorial DE. In: IEEE Congress on Evolutionary Computation, pp. 921–928 (2017)

11. Feng, L., Ong, Y.S., Tan, A.H., Tsang, I.W.: Memes as building blocks: a case study on evolutionary optimization+ transfer learning for routing problems. Memetic Comput. **7**(3), 159–180 (2015)

12. Fister Jr., I., Yang, X.S., Fister, I., Brest, J., Fister, D.: A brief review of nature-inspired algorithms for optimization/kratki pregled algoritmov po vzoru iz narave za optimizacijo. Elektrotehniski Vestnik **80**(3), 116 (2013)

13. Gendreau, M., Laporte, G., Semet, F.: A tabu search heuristic for the undirected selective travelling salesman problem. Eur. J. Oper. Res. **106**(2), 539–545 (1998)

14. Gong, M., Tang, Z., Li, H., Zhang, J.: Evolutionary multitasking with dynamic resource allocating strategy. IEEE Trans. Evol. Comput. **23**(5), 858–869 (2019)

15. Grefenstette, J., Gopal, R., Rosmaita, B., Van Gucht, D.: Genetic algorithms for the traveling salesman problem. In: Proceedings of the first International Conference on Genetic Algorithms and their Applications, pp. 160–168. Lawrence Erlbaum, New Jersey (1985)

16. Gupta, A., Ong, Y.S.: Genetic transfer or population diversification? Deciphering the secret ingredients of evolutionary multitask optimization. In: IEEE Symposium Series on Computational Intelligence, pp. 1–7 (2016)

17. Gupta, A., Ong, Y.S., Feng, L.: Multifactorial evolution: toward evolutionary multitasking. IEEE Trans. Evol. Comput. **20**(3), 343–357 (2015)

18. Gupta, A., Ong, Y.S., Feng, L., Tan, K.C.: Multiobjective multifactorial optimization in evolutionary multitasking. IEEE Trans. Cybern. **47**(7), 1652–1665 (2016)

19. Ibrahim, S., Thangamani, M.: Enhanced singular value decomposition for prediction of drugs and diseases with hepatocellular carcinoma based on multi-source bat algorithm based random walk. Measurement **141**, 176–183 (2019)

20. Kumbharana, S.N., Pandey, G.M.: Solving travelling salesman problem using firefly algorithm. Int. J. Res. Sci. Adv. Technol. **2**(2), 53–57 (2013)

21. Lawler, E.L., Lenstra, J.K., Kan, A.R., Shmoys, D.B.: The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization. Wiley, New York (1985)

22. Li, G., Zhang, Q., Gao, W.: Multipopulation evolution framework for multifactorial optimization. In: Genetic and Evolutionary Computation Conference Companion, pp. 215–216 (2018)

23. Lin, S.: Computer solutions of the traveling salesman problem. Bell Syst. Tech. J. **44**(10), 2245–2269 (1965)

24. Lu, Y., Jiang, T.: Bi-population based discrete bat algorithm for the low-carbon job shop scheduling problem. IEEE Access **7**, 14513–14522 (2019)

25. Luque, G., Alba, E.: Parallel Genetic Algorithms: Theory and Real World Applications, vol. 367. Springer, Heidelberg (2011)

26. Miller, C.E., Tucker, A.W., Zemlin, R.A.: Integer programming formulation of traveling salesman problems. J. ACM **7**(4), 326–329 (1960)

27. Ong, Y.-S.: Towards evolutionary multitasking: a new paradigm in evolutionary computation. In: Senthilkumar, M., Ramasamy, V., Sheen, S., Veeramani, C., Bonato, A., Batten, L. (eds.) Computational Intelligence, Cyber Security and Computational Models. AISC, vol. 412, pp. 25–26. Springer, Singapore (2016). https://doi.org/10.1007/978-981-10-0251-9_3

28. Ong, Y.S., Gupta, A.: Evolutionary multitasking: a computer science view of cognitive multitasking. Cogn. Comput. **8**(2), 125–142 (2016)

29. Osaba, E., Carballedo, R., Diaz, F., Onieva, E., Masegosa, A., Perallos, A.: Good practice proposal for the implementation, presentation, and comparison of metaheuristics for solving routing problems. Neurocomputing **271**, 2–8 (2018)

30. Osaba, E., Del Ser, J., Sadollah, A., Bilbao, M.N., Camacho, D.: A discrete water cycle algorithm for solving the symmetric and asymmetric traveling salesman problem. Appl. Soft Comput. **71**, 277–290 (2018)
31. Osaba, E., Yang, X.S., Diaz, F., Lopez-Garcia, P., Carballedo, R.: An improved discrete bat algorithm for symmetric and asymmetric traveling salesman problems. Eng. Appl. Artif. Intell. **48**, 59–71 (2016)
32. Osaba, E., Yang, X.S., Fister Jr., I., Del Ser, J., Lopez-Garcia, P., Vazquez-Pardavila, A.J.: A discrete and improved bat algorithm for solving a medical goods distribution problem with pharmacological waste collection. Swarm Evol. Comput. **44**, 273–286 (2019)
33. Ouaarab, A., Ahiod, B., Yang, X.: Discrete cuckoo search algorithm for the travelling salesman problem. Neural Comput. Appl. **24**, 1659–1669 (2013). https://doi.org/10.1007/s00521-013-1402-2
34. Pan, S.J., Yang, Q.: A survey on transfer learning. IEEE Trans. Knowl. Data Eng. **22**(10), 1345–1359 (2009)
35. Precup, R.E., David, R.C.: Nature-Inspired Optimization Algorithms for Fuzzy Controlled Servo Systems. Butterworth-Heinemann (2019)
36. Reinelt, G.: TSPLIB: a traveling salesman problem library. ORSA J. Comput. **3**(4), 376–384 (1991)
37. Song, H., Qin, A., Tsai, P.W., Liang, J.: Multitasking multi-swarm optimization. In: IEEE Congress on Evolutionary Computation, pp. 1937–1944 (2019)
38. Wang, C., Ma, H., Chen, G., Hartmann, S.: Evolutionary multitasking for semantic web service composition. arXiv preprint arXiv:1902.06370 (2019)
39. Wen, Y.W., Ting, C.K.: Parting ways and reallocating resources in evolutionary multitasking. In: IEEE Congress on Evolutionary Computation, pp. 2404–2411 (2017)
40. Xiao, H., Yokoya, G., Hatanaka, T.: Multifactorial PSO-FA hybrid algorithm for multiple car design benchmark. In: IEEE International Conference on Systems, Man and Cybernetics, pp. 1926–1931 (2019)
41. Yang, X.S.: A new metaheuristic bat-inspired algorithm. In: González, J.R., Pelta, D.A., Cruz, C., Terrazas, G., Krasnogor, N. (eds.) Nature Inspired Cooperative Strategies for Optimization (NICSO 2010). Studies in Computational Intelligence, vol 284. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-12538-6_6
42. Yang, X.S., He, X.: Bat algorithm: literature review and applications. Int. J. Bio-Inspired Comput. **5**(3), 141–149 (2013)
43. Yu, Y., Zhu, A., Zhu, Z., Lin, Q., Yin, J., Ma, X.: Multifactorial differential evolution with opposition-based learning for multi-tasking optimization. In: IEEE Congress on Evolutionary Computation, pp. 1898–1905 (2019)
44. Yuan, Y., Ong, Y.S., Gupta, A., Tan, P.S., Xu, H.: Evolutionary multitasking in permutation-based combinatorial optimization problems: realization with TSP, QAP, LOP, and JSP. In: IEEE Region 10 Conference, pp. 3157–3164 (2016)
45. Zheng, X., Lei, Y., Gong, M., Tang, Z.: Multifactorial brain storm optimization algorithm. In: Gong, M., Pan, L., Song, T., Zhang, G. (eds.) BIC-TA 2016. CCIS, vol. 682, pp. 47–53. Springer, Singapore (2016). https://doi.org/10.1007/978-981-10-3614-9_6
46. Zhou, L., et al.: Towards effective mutation for knowledge transfer in multifactorial differential evolution. In: IEEE Congress on Evolutionary Computation, pp. 1541–1547 (2019)
47. Zhou, L., Feng, L., Zhong, J., Ong, Y.S., Zhu, Z., Sha, E.: Evolutionary multitasking in combinatorial search spaces: a case study in capacitated vehicle routing problem. In: IEEE Symposium on Computational Intelligence, pp. 1–8 (2016)