Multifactorial Cellular Genetic Algorithm (MFCGA): Algorithmic Design, Performance Comparison and Genetic Transferability Analysis

Eneko Osaba[†*], Aritz D. Martinez[†*], Jesus L. Lobo[†], Javier Del Ser[†‡] and Francisco Herrera[§]
[†]TECNALIA, Basque Research and Technology Alliance (BRTA), 48160 Derio, Bizkaia, Spain
Email: [eneko.osaba, aritz.martinez, jesus.lopez, javier.delser]@tecnalia.com
[‡]University of the Basque Country (UPV/EHU), 48013 Bilbao, Bizkaia, Spain
[§]DaSCI Andalusian Institute of Data Science and Computational Intelligence. University of Granada. 18071 Granada, Spain
[*] Corresponding authors. These authors have equally contributed to the work presented in this paper.

***Abstract*—Multitasking optimization is an incipient research area which is lately gaining a notable research momentum. Unlike traditional optimization paradigm that focuses on solving a single task at a time, multitasking addresses how multiple optimization problems can be tackled simultaneously by performing a single search process. The main objective to achieve this goal efficiently is to exploit synergies between the problems (tasks) to be optimized, helping each other via knowledge transfer (thereby being referred to as Transfer Optimization). Furthermore, the equally recent concept of Evolutionary Multitasking (EM) refers to multitasking environments adopting concepts from Evolutionary Computation as their inspiration for the simultaneous solving of the problems under consideration. As such, EM approaches such as the Multifactorial Evolutionary Algorithm (MFEA) has shown a remarkable success when dealing with multiple discrete, continuous, single-, and/or multi-objective optimization problems. In this work we propose a novel algorithmic scheme for Multifactorial Optimization scenarios – the Multifactorial Cellular Genetic Algorithm (MFCGA) – that hinges on concepts from Cellular Automata to implement mechanisms for exchanging knowledge among problems. We conduct an extensive performance analysis of the proposed MFCGA and compare it to the canonical MFEA under the same algorithmic conditions and over 15 different multitasking setups (encompassing different reference instances of the discrete Traveling Salesman Problem). A further contribution of this analysis beyond performance benchmarking is a quantitative examination of the genetic transferability among the problem instances, eliciting an empirical demonstration of the synergies emerged between the different optimization tasks along the MFCGA search process.**

***Index Terms*—Transfer Optimization, Evolutionary Multitasking, Cellular Genetic Algorithm, Multifactorial Evolutionary Algorithm, Traveling Salesman Problem**

I. Introduction

Inspired by the roots of Transfer Learning [1] and Multitask Learning [2], Transfer Optimization is a relatively new knowledge field within the wider area of optimization, which is attracting great attention within the community in recent years [3]. The main idea is to exploit what has been learned for optimizing one given optimization problem, toward tackling another related or unrelated problem. Due to its relative youth, efforts devoted by the scientific community for advancing over this emerging research area are considerably fewer than those dedicated to Transfer and Multitask Learning, which address a similar problem for Machine Learning tasks. It has not been until recently when the transferability of knowledge among optimization problems have become a research priority, mainly due to the increasing complexity and scales of optimization problems, and the subsequent need for harnessing knowledge acquired beforehand.

Three different algorithmic categories can be identified in the Transfer Optimization panorama [4]. The first one is *sequential transfer*, in which optimization problems (tasks) are solved in a sequential fashion under the assumption that for optimizing a new problem/instance, the knowledge acquired when solving previous tasks is used as external information [5]. The second class is the so-called *multitasking*, which tackles different tasks of equal priority in a simultaneous way by dynamically exploiting existing synergies and complementarities among problems [6], [7]. The last category is referred to as *multiform optimization*, which aims at the resolution of a single task through the use of alternative problem formulations, which are solved in a simultaneous way. As can be observed in the literature background, in all these three categories the correlation among problem instances or tasks is crucial for positively capitalizing on the transference of knowledge over the search [8].

Among the above three categories, the most prolific one in the current literature is *multitasking*. The research presented in this study is centered on this category. Being more specific, we focus on multitasking optimization through the perspective of Evolutionary Multitasking (EM, [9]). In short, EM embraces the concepts, operators and search strategies conceived within Evolutionary Computation for simultaneously solving several problems at a time [10], [11]. As such, EM underlies Multifactorial Optimization (MFO, [8]), a particular realization of this paradigm that has demonstrated its potential in different environments encompassing continuous, discrete and multi-objective optimization problems [12]–[15]. In the current community, the majority of contributions around MFO gravitate on the Multifactorial Evolutionary Algorithm (MFEA, [8]), or variants of this algorithm.

978-1-7281-6929-3/20/$31.00 ©2020 IEEE

Bearing this background in mind, this work presents a new MFO approach coined as Multifactorial Cellular Genetic Algorithm (MFCGA). We take a step further over the state of the art by elaborating on several research directions:

- We introduce a new efficient meta-heuristic scheme for MFO that relies on the foundations of Cellular Automata and Celular Genetic Algorithms (cGAs, [16]) for controlling the mating process among different species (problems). Moreover, the search strategy of the proposed MFCGA solver favors the exploration and quantitative examination of synergies among the problems being solved, providing a sort of explainability interface for understanding the interactions between problems. To the best of our knowledge, MFCGA is the first cGA for being applied to EM, and to the wider Transfer Optimization domain.
- We conduct an extensive experimentation focused on the well-known Traveling Salesman Problem (TSP, [17]). Specifically, we compare the performance of our proposed solver with those obtained by the canonical MFEA, with the firm intention of demonstrating that our solver is a promising alternative to face MFO scenarios. To do that, we employ 8 different TSP instances, which have been used to generate 15 different scenarios. We also examine the genetic transferability among the used TSP instances, which poses a valuable addition to the state of the art [18], [19], and provides useful insights for further research.

The rest of the paper is organized as follows: Section II presents a brief overview of the background related to Evolutionary Multitasking, MFEA and cGAs. Next, in Section III, we describe the main characteristics of our proposed MFCGA in detail. Experimental results obtained with the developed method are discussed in Section IV, along with a description of the benchmark and the experimental setup under consideration. Finally, Section V concludes the paper by drawing conclusions and outlining future research lines.

II. Background

As stated above, this section is devoted to providing a brief background about the three main concepts addressed in this paper: EM and MFO (Section II-A), MFEA (Section II-B), and the area of cGAs (Section II-C).

A. Evolutionary Multitasking and Multifactorial Optimization

In contrast to sequential transfer, in which a single optimization task is addressed at a time, multitasking focuses on simultaneously addressing several tasks. While sequential transfer optimization seeks an unidirectional transfer of knowledge from previously completed tasks to new ones, multitasking is characterized by omnidirectional knowledge transfer among tasks, pursuing a more synergistic completion of the tasks under consideration [4].

Within this landscape, EM has emerged as a promising paradigm for dealing with simultaneous transfer optimization scenarios. Two main characteristics motivated the first formulation of the EM paradigm. The first feature is the inherent parallelism offered by a population of individuals, which allows for efficient computational means to deal with concurrent optimization tasks faced simultaneously. It is precisely this simultaneous treatment what permits latent relationships between problems to be automatically harnessed during the search process [3]. The second interesting characteristic is that the constant transfer of genetic material along the evolutionary search allows all tasks to benefit each other, even for tasks that are not strongly correlated [8], [20].

It was not until late 2017 when the concept of EM was only formalized through the perspective of the MFO paradigm [21]. Firstly introduced in [8], this incipient branch of the evolutionary computation field is grasping notable interest in terms of new algorithmic schemes, such as hybrid solvers [22], multifactorial heuristic engines encompassing modern metaphors [23] or multi-population methods [24] under the development of a novel multitasking multi-swarm optimization. Despite this recent upsurge of new MFO schemes, MFEA has dominated the knowledge stream of MFO since its conception.

Mathematically speaking, MFO can be formally described as an EM environment in which K optimization tasks are simultaneously optimized. This environment is characterized in this way by the existence of multiple search spaces, each related to a single task. Assuming that all tasks are minimization problems, for the k-th task T_k its objective function is characterized as $f_k : \Omega_k \rightarrow \mathbb{R}$, where Ω_k denotes the solution space of T_k. This being said, the main goal of MFO is to find a set of solutions $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_K\}$ such that $\mathbf{x}_k = \arg\min_{\mathbf{x}\in\Omega_k} f_k(\mathbf{x})$. However, instead of tackling K independent search processes in isolation, MFO pursues to find $\{\mathbf{x}_k\}_{k=1}^{K}$ by exploring a single, unified search space Ω'. Therefore, solutions $\mathbf{x}' \in \Omega'$ can be encoded and decoded to represent a task-specific solution $\mathbf{x}_k$ for any of the K optimization tasks under consideration.

Moreover, MFO is based on four different definitions, associated to each individual $\mathbf{x}'_p \in \Omega'$ within a P-sized population:

Definition 1 (Factorial Cost): the factorial cost Ψ_k^p of a population member $\mathbf{x}'_p$ is equal to the value of the fitness function for a given task T_k. Each individual counts with a list $\{\Psi_1^p, \Psi_2^p, \ldots, \Psi_K^p\}$ of factorial costs, each related to an optimization task.

Definition 2 (Factorial Rank): the factorial rank r_k^p of an individual $\mathbf{x}_p$ in a given task T_k is the index of this individual within the population sorted in ascending order of Ψ_k^p. Each population member has a factorial rank list $\{r_1^p, r_2^p, \ldots, r_K^p\}$.

Definition 3 (Scalar Fitness): the scalar fitness φ^p of $\mathbf{x}'_p$ is calculated by using the best factorial ranks over all the tasks, i.e., $\varphi^p = 1/\left(\min_{k\in\{1\ldots K\}} r_k^p\right)$. As will be exposed in Section II-B, the scalar fitness is used for comparing individuals in MFEA.

Definition 4 (Skill Factor): The skill factor τ^p is the task in which $\mathbf{x}'_p$ performs best, namely, $\tau^p = \arg\min_{k\in\{1,\ldots,K\}} r_k^p$. The skill factor plays a crucial role in MFEA by establishing which population members are selected for crossover.

The research activity around MFO and MFEA has been vibrant in the last few years. In [25] MFEA is applied to different discrete problems, such as the job shop scheduling

problem and the TSP. This paper also introduces the unified discrete encoding strategy, which is embraced in this work. A similar study is proposed in [26], where MFEA is put to practice to deal with vehicle routing problems. In [15], a multiobjective variant of MFEA is proposed, proving its efficiency over continuous benchmark functions, as well as a real-world manufacturing process design problem. An interesting discrete MFEA is also developed in [12] for Semantic Web Service Composition. An improved version of MFEA was proposed in [13], which endowed the algorithm with a dynamic resource allocating strategy. Likwewise, the enhanced MFEA presented in [14] follows a similar philosophy by incorporating opposition-based learning. Further works around MFO and MFEA can be found in [27], [28] and [14].

B. Multifactorial Evolutionary Algorithm

MFEA is a recently proposed MFO method for solving EM problems using bio-cultural schemes of multifactorial inheritance [8]. The basic workflow of MFEA is depicted in Algorithm 1. For deeper details on the algorithmic operators, we refer reader to [8]. For simultaneously dealing with all optimizing tasks, MFEA has four cornerstone characteristics: unified solution representation, assortative mating, selective evaluation, and scalar fitness based selection:

- The design of the representation strategy for $\mathbf{x}'_p$ that yields the unified search space Ω' is subject to the characteristics of the K problems under consideration. Specifically for this work, the TSP is used as the family of benchmark problems for assessing the performance of both MFEA and the proposed MFCGA. For this reason, the well-known permutation encoding is used as the unified representation for $\mathbf{x}'_p$ [29]. Following [25], if K TSP problems are to be simultaneously solved, and by denoting the size of each TSP problem T_k (i.e. the number of *cities*) as D_k, an individual $\mathbf{x}'_p$ is encoded as a permutation of the integer set $\{1, 2, \ldots, D_{max}\}$, where $D_{max} = \max_{k\in\{1,\ldots,K\}} D_k$, namely, the maximum problem size among the K tasks. In this way, when $\mathbf{x}'_p$ is to be evaluated for a task T_k whose $D_k < D_{max}$, only integers lower than D_k are considered for producing the argument solution $\mathbf{x}_k$ of $f_k(\cdot)$. These integers maintain the same order as in $\mathbf{x}'_p$.
- Assortative mating establishes that individuals prefer to interact with other mates belonging to similar cultural background [8]. Thus, as described in [15], [27], [30], genetic operators used in the MFEA promote mating among individuals with the same skill factor τ^p. We recommend consulting these papers for deeper details on how this breeding mechanism is implemented in MFEA.
- Selective evaluation implies that each generated individual is measured only for one task, instead of evaluating it for every task. Specifically, the produced offspring is evaluated in task $T_{\tau^p_*}$, where τ^p_* is the skill factor of its parent (or the skill factor selected at random among the two parents of the offspring). This means that the factorial cost Ψ^p_k is set to ∞ $\forall k \in \{1, \ldots, \tau^p_* - 1, \tau^p_* + 1, \ldots, K\}$.
- Finally, the scalar fitness based selection is a survivor function similar to those used in basic Genetic Algorithms. In this case, MFEA is based on an elitist strategy, i.e. the best P individuals in terms of scalar fitness σ^p among those in the current population and the newly produced offspring survive for the next generation.

Algorithm 1: Pseudocode of the canonical MFEA

```
Randomly generate a population of P individuals
Evaluate each generated individual for the K problems
Calculate the skill factor (τ^p) of each individual x'_p
repeat
    Apply genetic operators on P to get the offspring subpopulation P_*
    Evaluate the generated offspring for the best task τ^p_* of their parent(s)
    Combine P and P_* in intermediate population Q
    Update the scalar fitness φ^p_k and skill factor τ^p for each individual in Q
    Build the next population by selecting the best P individuals in Q in terms of scalar fitness
until termination criterion not reached
Return the best individual for each task T_k
```

C. Cellular Genetic Algorithm

Briefly explained, cGAs are a sub-type of the canonical GAs in which the population is structured in a specific topology based on small-sized neighborhoods [16]. Thereby, individuals can only interact with their neighbors, which enhances the exploration of the search space through the induced slow diffusion of solutions across the population. On the other hand, exploitation is carried out inside each neighborhood [31]. Therefore, while in classical GAs the population is structured in a unique panmictic group, in cGAs the whole population is arranged over a grid (typically two-dimensionals), on which the aforementioned neighborhood relation is defined. Two are the most frequently used neighborhood structures: i) NEWS, linear5, or Von Neumann, in which the neighborhood of each individual is composed by its North (N), East (E), West (W), and South (S) individuals; and ii) C9, or Moore, in which the neighborhood is given by NW, N, NE, W, E, SW, S and SE individuals. We recommend [32] for additional information about cellular grid structures, and [33]–[35] for an excerpt of different theoretical works and applications of this particular kind of evolutionary algorithms.

As pointed, in cGAs each individual can only interact with its assigned neighbors. Thus, the genetic crossover operates inside the neighborhoods, modifying each individual with one of its neighbors. Furthermore, newly generated individuals are not introduced in the population. On the contrary, they replace the current individual upon the fulfillment of a given criterion (for example, an improvement in the fitness function). Additionally, two cGA types can be distinguished depending on the update policy of the population: synchronous cGA and

asynchronous cGA. On the one hand, synchronous cGAs are characterized by implementing all the replacements in parallel. On the other hand, in asynchronous cGAs individuals are sequentially updated, thus overriding any need for auxiliary populations and adapting faster to the newly generated genetic material. These are the main reasons why we have chosen this second scheme for our research work.

III. Multifactorial Cellular Genetic Algorithm

As we have been identified previously, the four pillars on which the operation of the MFEA is based are unified representation, assortative mating, selective evaluation, and scalar fitness based selection. These concepts have been embraced and reformulated in this work to yield the workflow of the proposed MFCGA shown in Algorithm 2, which are inspired by both MFEA and cGAs.

Algorithm 2: Pseudocode of the proposed MFCGA

```
Randomly generate a population of P individuals
Evaluate each generated individual for the K problems
Calculate the skill factor (τ^p) of each individual x'_p
Let X^⊛_p denote the set of neighbors of x'_p
while termination criterion not reached do
    for p = 1, ..., P do
        Randomly choose a neighbor x_j from X^⊛_p
        x^crossover_p ← crossover(x'_p, x_j)
        x^mutation_p ← mutation(x'_p)
        Evaluate x^crossover_p and x^mutation_p for τ^p
        x'_p ← best(x'_p, x^crossover_p, x^mutation_p)
        Update φ^p and τ^p of the evolved x'_p
    end
end
Return the best individual for each task T_k
```

First, as unified representation, the same philosophy and encoding as in the case of the MFEA has been used. Regarding the genetic operators, they are based on the classical evolutionary crossover and mutation procedures: at every generation, each individual $\mathbf{x}'_p$ goes through these two phases (without using any crossover or mutation probabilities), producing two new individuals: $\mathbf{x}_p^{crossover}$ and $\mathbf{x}_p^{mutation}$. The first of these newly created individuals is the result of mating $\mathbf{x}'_p$ with a randomly chosen neighbor $\mathbf{x}_j$ from the cellular neighborhood $\mathbf{X}_p^{\circledast}$ of $\mathbf{x}'_p$. Correspondingly, the mutation operator applied to $\mathbf{x}'_p$ gives rise to $\mathbf{x}_p^{mutation}$.

Once $\mathbf{x}_p^{crossover}$ and $\mathbf{x}_p^{mutation}$ have been generated, their quality is evaluated by using the same selective evaluation described in Subsection II-B. In this way, we ensure that MFGCA is as computationally efficient as MFEA. It should be mentioned here that both $\mathbf{x}_p^{crossover}$ and $\mathbf{x}_p^{mutation}$ are evaluated for task T_{τ^p}, where τ^p is the skill factor of $\mathbf{x}'_p$. This implies a significant difference with respect to MFEA, since in MFGCA individuals are devoted to the optimization of the same single task along the whole execution, not changing at all from one task to another. Moreover, the first complete evaluation and sorting of the population, based on the factorial rank and scalar factor, ensures the equilibrium between the population, allocating a similar number of individuals to each of the tasks.

A final aspect of the proposed MFCGA is the local improvement selection mechanism, by which the newly generated $\mathbf{x}_p^{crossover}$ or $\mathbf{x}_p^{mutation}$ can only substitute their parent $\mathbf{x}'_p$. In fact, the individual that survives to the next generation is the best one among $\mathbf{x}'_p$, $\mathbf{x}_p^{crossover}$ and $\mathbf{x}_p^{mutation}$. The other two produced individuals are automatically discarded.

IV. Experimental Setup and Results

We proceed by describing an experimentation conducted for comparing both MFEA and MFCGA solvers, properly analyzing the genetic transfer within MFCGA and examining the synergies between the chosen tasks. As mentioned previously, the experimentation has been done over the well-known TSP [36]. Since its inception, the TSP has become one of the most popular benchmark problems for the performance assessment of discrete optimization algorithms, from traditional metaheuristics such as GAs [37] or Ant Colony Optimization [38], to more recently introduced bio-inspired solvers, such as the Firefly Algorithm [39], Bat Algorithm [40], or the Water Cycle Algorithm [41], among others. In the context of the present study, our main objective is not to find the optimal solution to the TSP problems under consideration. Instead, we aim to statistically compare the performance of both MFEA and MFCGA using same problem instances and conditions.

Specifically, The performance of the developed MFEA and MFCGA has been gauged over 15 different combinations (*test cases*) of the Krolak/Felts/Nelson set of TSP instances contained in the renowned TSPLIB repository [42]. It is important to highlight that these cases have been selected not only because of their wide acceptance by the community, but also since the different levels of genetic complementarities in their structure. This complementarity is measured using the percentage of nodes that instances share between them. Thus, our intention is to explore the impact of this complementarities in the genetic exchange inherent to EM schemes. In Table I, a summary of genetic complementarities is shown for all the datasets considered in the experimentation.

TABLE I
Summary of genetic complementarities for all the datasets employed in the experimentation

Instance	kroA100	kroB100	kroC100	kroD100	kroE100	kroA150	kroA200	kroB150
kroA100		1%	2%	1%	2%	**80%**	**66%**	1%
kroB100			1%	2%	1%	1%	0%	0%
kroC100				1%	1%	1%	**66%**	**80%**
kroD100					1%	1%	2%	1%
kroE100						**40%**	0%	**40%**
kroA150							**57%**	1%
kroA200								**57%**

Each of the 15 multitasking test cases implies that the modeled approaches should solve all the tasks assigned to that scenario. As shown in Table II, 10 of these test cases are comprised by four TSP instances, 4 are composed by 6 TSP instances, and the last one contemplates the resolution of

all the 8 TSP problems under consideration. Two have been the main reasons of building these tests cases: i) to ensure the heterogeneity and variety of the configurations, meaning that each TSP instance is part of exactly the same number of test cases; and ii) to examine how the genetic synergies depicted in Table I are exploited during the search process by the proposed MFCGA approach.

TABLE II
SUMMARY OF THE 15 TEST CASES BUILT FOR THE EXPERIMENTATION

Test Case	Tasks involved
TC_4_1	kroA100, kroA150, kroA200, kroC100
TC_4_2	kroB100, kroB150, kroD100, kroE100
TC_4_3	kroA100, kroA150, kroD100, kroE100
TC_4_4	kroA200, kroC100, kroB100, kroB150
TC_4_5	kroA100, kroA200, kroB100, kroD100
TC_4_6	kroA150, kroC100, kroB150, kroE100
TC_4_7	kroA100, kroA150, kroB100, kroB150
TC_4_8	kroA200, kroC100, kroD100, kroE100
TC_4_9	kroA100, kroC100, kroB100, kroD100
TC_4_10	kroA150, kroA200, kroB150, kroE100
TC_6_1	kroA100, kroA150, kroA200, kroB100, kroC100, kroB150
TC_6_2	kroA200, kroB100, kroC100, kroB150, kroD100, kroE100
TC_6_3	kroA100, kroA150, kroA200, kroB150, kroD100, kroE100
TC_6_4	kroA100, kroA150, kroB100, kroC100, kroD100, kroE100
TC_8	kroA100, kroA150, kroA200, kroB100, kroC100, kroB150, kroD100, kroE100

Regarding the algorithmic setup, we have used similar parameters and the same operators for all the implemented algorithms to ensure a fair comparison. For the sake of reproducibility of the presented results, the parameterization used for both MFEA and MFCGA are listed in Table III. For this parameterization, studies focused on cGAs and MFEA have been used as inspiration [25], [31], along with the methodological guidelines given in [43]. Accordingly, results are reported on the basis of 20 independent runs for every test case to inspect the statistical significance of eventual performance gaps. In addition, both MFEA and MFCGA are stopped after $500 \cdot 10^3$ objective function evaluations. All experiments have been executed on an Intel Xeon E52650 v3 2.30 GHz processor with 32 GB RAM.

TABLE III
PARAMETRIZATION OF MFEA AND MFCGA

Parameter	MFEA	MFCGA
Population size	200	
crossover(·)	Order crossover [44]	
mutation(·)	2-opt	
Crossover probability	0.9	1.0
Mutation probability	0.1	1.0
Type of grid		Moore

A Java implementation of the MFCGA has been made publicly available in *https://git.code.tecnalia.com/aritz.martinez/mfcga*, together with the scripts that generate the results next discussed.

A. Results and Discussion

We begin our discussion by analyzing Table IV, which summarizes graphically the comparisons between the outcomes obtained by both MFCGA and MFEA in all the 15 test cases described above. Specifically, an orange circle ● indicates that MFCGA outperforms MFEA in terms of fitness average for a given TSP problem instance. On the other hand, the gray circle ● denotes that MFEA has reached better average outcomes. Let us take TC_4_2 as an example: in this case, and considering the order of instances within the test case provided in Table II, we can observe that MFCGA performs better in kroB100, kroB150, and kroE100, while MFEA dominates only in the kroE100 instance. By extrapolating this analysis to the remaining content of the table, we conclude that MFCGA elicits a better performance for tackling these test cases, outperforming MFEA in all but six problem instances. It is also important to underscore that for TC_8, MFCGA attains better performance scores in all its compounding 8 TSP instances.

TABLE IV
COMPARISON OF THE RESULTS FOR THE 15 TEST CASES (●: MFCGA OUTPERFORMS MFEA; ●: MFEA OUTPERFORMS MFCGA).

Test Case	MFGCA versus MFEA
TC_4_1	●●●●
TC_4_2	●●●●
TC_4_3	●●●●
TC_4_4	●●●●
TC_4_5	●●●●
TC_4_6	●●●●
TC_4_7	●●●●
TC_4_8	●●●●
TC_4_9	●●●●
TC_4_10	●●●●
TC_6_1	●●●●●●
TC_6_2	●●●●●●
TC_6_3	●●●●●●
TC_6_4	●●●●●●
TC_8	●●●●●●●●

Table V exemplifies the process by which the above results have been produced for the TC_8 test case. In this table we depict, for each TSP instance in the test case, the average, best and standard deviation of the fitness value achieved by MFEA and MFCGA computed over 20 independent runs. Additionally, we also represent the known optima for each instance. It is straightforward to note that MFCGA clearly outperforms MFEA in terms of average results. Furthermore, regarding the best solution found over the 20 independent runs, MFCGA also dominates the benchmark, obtaining a better performance in 5 out of the 8 cases. Finally, it is also interesting to notice that the difference between the known optima and the average outcomes obtained by MFCGA ranges between 3.8% and 4.7% in problem instances with 100 nodes, and between 4.7% and 9.3% for problems of larger size.

In order to verify the statistical significance between the results returned by MFCGA and MFEA, the Wilcoxon Rank-Sum test has been applied to their fitness outcomes. The confidence interval has been set at 95%. For properly building

this statistical test, we have compared the results reached in all the 8 datasets separately, depicting graphically the outcomes of these Wilcoxon Rank-Sum tests in the last row of Table V. In this row, an orange circle ● means that MFCGA outperforms MFEA with statistical significance. On the other hand, the gray circle ● denotes that there is no enough evidence to claim that the improvement is statistically relevant. As a summary of all these tests, the obtained average z-value is -1.96, with an average p-value equal to 0.04888. Taking into account the critical z_c value is equal to -1.64, and since $-1.96 < -1.64$ and $0.04888 < 0.05$, these results support the significance of the difference at 95% confidence level. Therefore, the difference is significant at this confidence level, thereby concluding that MFCGA is statistically better than MFEA for this test case.

TABLE V
RESULTS OBTAINED BY MFCGA AND MFEA FOR THE 8 INSTANCES IN TC_8, AND GRAPHICAL RESULTS OF THE WILCOXON RANK-SUM TEST.

Method	kroA100	kroA150	kroA200	kroB100	kroC100	kroB150	kroD100	kroE100
MFCGA	**22099.1** 21746.0 203.89	**28588.1** **27893.0** 394.96	**32109.0** **31162.0** 547.66	**23168.9** 22815.0 249.66	**21494.7** **20852.0** 337.44	**27780.5** **27307.0** 207.34	**22257.7** 21648.0 398.46	**23069.4** **22587.0** 211.52
MFEA	22404.6 **21460.0** 703.53	28817.0 28385.0 299.44	32769.8 31856.0 596.42	23790.0 **22330.0** 609.08	21956.3 21157.0 739.15	28512.0 27394.4 788.14	22713.3 **21539.0** 548.83	23239.3 22607.0 533.68
Optima	21282.0	26524.0	29368.0	22141.0	20749.0	26524.0	21294.0	22068.0
Wilcoxon rank-sum test	●	●	●	●	●	●	●	●

B. Analysis of the Genetic Transfer between Tasks

We now analyze the genetic transfer across the 8 TSP tasks considered in the complete experimentation, focusing on our proposed MFCGA. The main objective with this study is i) to get a glimpse of the positive knowledge transfer among problem instances; ii) to discover synergies between them; and iii) to empirically gauge inter-task interactions occurred along the 20 repetitions of `TC_8`. We have chosen this test case since it is the one in which the 8 TSP tasks are optimized jointly.

It should be pointed here that the novel MFCGA presented in this paper is especially interesting for analyzing the genetic transfer held through the algorithm execution. This is so due to the replacement strategy employed in MFCGA. In our method, an individual $\mathbf{x}'_p$ of the population is replaced if and only if any of the individuals generated through the `crossover` ($\mathbf{x}_p^{crossover}$) and `mutation` ($\mathbf{x}_p^{mutation}$) operators outperform $\mathbf{x}'_p$ in terms of its best performing task (i.e. its skill factor). Thus, if $\mathbf{x}_p^{crossover}$ replaces $\mathbf{x}'_p$, a positive transfer of genetic material has occurred from $\mathbf{x}_j$ to $\mathbf{x}'_p$ (we refer to Algorithm 2 and Section III for notation details). In the context of the TSP, this transfer is realized through the direct insertion of part of the neighboring solution $\mathbf{x}_j$ into $\mathbf{x}'_p$, which can be conceived as a positive contribution of task τ^j to task τ^p.

Bearing the above explanation in mind, Figure 1 represents the number of positive genetic transfer episodes (through the `crossover` operator) between every pair of TSP tasks. The radius of every orange circle in this plot is proportional to the average number of times per execution in which an individual having the skill factor indicated in the column label has exchanged some of its genetic material with an individual whose skill factor is given in the row label. Moreover, circles located in the diagonal represent the sum of all the inter-task (orange portion) and intra-task exchanges (gray portion), the latter quantifying the genetic transfer between individuals featuring the same skill factor.

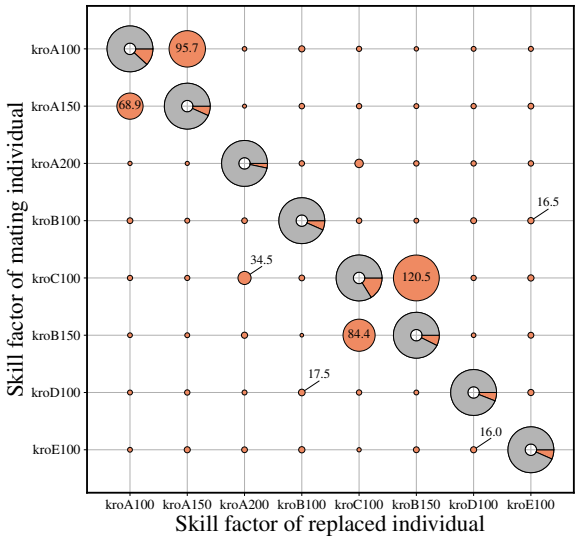

Fig. 1. Average intensities of genetic transfer between TSP instances.

Two main conclusions can be drawn after analyzing this figure. The first one is the confirmation of synergy in three different pairs of TSP instances, namely, {kroA100, kroA150}, {kroC100, kroA200} and {kroC100, kroB150}. Thus, the genetic transfer between these tasks positively contributes to the multi-task search process. The second conclusion is that for the rest of task pairs, the intensity of genetic material exchange is almost nonexistent. This fact unveils that transfers between these tasks can be considered as negative [45], and that they do not contribute for the search process. These findings a priori contradict the information depicted in Table I, in which we summarized the genetic complementarity of the 8 tasks. For instance, we saw that tasks such as kroA100 and kroA200 have a high level of complementarity among them (66% as per the table). However, the inter-task interaction for this pair depicted in Figure 1 is practically nonexistent. Similar conclusions hold for other task pairs, such as {kroA150, kroA200} or {kroA200, kroB150}. This contradiction collides with some previously published studies [8]. In fact, by analyzing the correlation in the landscapes of the aforementioned task pairs, we can confirm that the so-called partial domain overlap exists [4]. This statement is confirmed since the domains of task pairs {kroA100, kroA150}, {kroA150, kroA200} and {kroA200, kroB150} partially overlap, existing a subset of features that are common to both tasks of every pair.

In order to shed light on this unexpected mismatch, a deeper analysis of the considered 8 TSP instances has been made. However, in this case we focus our attention on the correlation among the best known solutions of such instances. Measur-

ing the distance between best solutions has been previously proposed in recent works on continuous problems [18], [21]. We summarize in Table VI the genetic complementarities in the optimal solutions of the 8 TSP tasks in use. Cells corresponding to task pairs that have shown a higher inter-task genetic transfer in Figure 1 have been highlighted in orange. As shown in this table, the best known solutions of {kroA100, kroA150}, {kroC100, kroA200} and {kroC100, kroB150} present a partial degree of intersection, which means that *the global optima of the two tasks are identical in the unified search space with respect to a subset of variables only, and different with respect to the remaining variables* [21]. At the same time, these three pairs are the ones that evince a higher intensity of interaction in the conducted experiments.

TABLE VI
GENETIC COMPLEMENTARITIES AMONG THE BEST KNOWN SOLUTIONS OF THE TSP INSTANCES UTILIZED IN THE EXPERIMENTATION

Instance	kroA100	kroB100	kroC100	kroD100	kroE100	kroA150	kroA200	kroB150
kroA100		0%	0%	0%	0%	**32%**	**5%**	0%
kroB100			0%	0%	0%	0%	0%	0%
kroC100				0%	0%	0%	**21%**	**10%**
kroD100					0%	0%	0%	0%
kroE100						**3%**	0%	**2%**
kroA150							**3%**	0%
kroA200								**8%**

This last analysis leads to the ultimate finding of our experimentation: the confirmation that for the TSP, positive inter-task genetic transfer is likely to happen among pairs of optimization tasks in which the degree of intersection in their best solution is, at least, partial. Specifically, our experiments elucidate that the degree of intersection should be above 10% for the transfer between tasks to be beneficial. In other words, the raw complementarity in the structure of the TSP scenario is irrelevant for the genetic transfer. For this reason, we conclude that TSP instances which do not partially share a fraction of their best solutions are prone to negative inter-task interactions in EM environments.

V. CONCLUSIONS AND FUTURE WORK

This work has elaborated on the design, implementation and performance assessment of a novel Multifactorial Cellular Genetic Algorithm (MFCGA) for Evolutionary Multitasking. Our proposed meta-heuristic approach is inspired by the well-known MFEA, and the influential cellular Genetic Algorithm (cGA). Specifically, the meta-heuristic search strategy relies on a neighborhood relationship induced on a grid arrangement of the individuals of the population, which restricts the coverage of the evolutionary crossover operator. For assessing the quality of our method, we have compared the performance of the MFCGA to that of MFEA along 15 tests cases comprising 8 different TSP instances. The obtained experimental outcomes support the preliminary conclusion that MFCGA is a promising method for solving EM environments. An equally important contribution of this work is the inter-task genetic transfer analysis conducted over the MFCGA, aimed at uncovering synergistic relationships among TSP instances that are exploited over the search process. Our main conclusion on this regard is that the genetic exchange can be positive whenever tasks present a minimum degree of intersection in the structure of their best solutions.

We plan to devote further efforts in a manifold of interesting research paths rooted on this initial study. In the short term, we will continue using the TSP as benchmarking problem using larger instances and test cases, targeting to assess the scalability of the developed MFCGA. Furthermore, additional search mechanisms for our method will be investigated and tested, such as heuristic local search methods or alternative survivor strategies. In the longer term, we will explore the application of the MFCGA to other fields [46] and additional discrete optimization problems, such as the vehicle routing [47], or community detection problems [48], [49]. In those cases, a similar analysis of the intra-task genetic transfer will be undertaken, possibly by resorting to other means for computing the similarity between solutions. Finally, a closer look will be taken at adaptive means to efficiently exploit synergies between solutions, by potentially optimizing the distribution of individuals over the grid according to such intra-task relationships. We will also try to adapt additional existing methods to this field, such as the firefly algorithm [50], bat algorithm [51] or grey wolf optimizer [52], [53].

ACKNOWLEDGMENTS

Eneko Osaba, Aritz D. Martinez, Jesus L. Lobo and Javier Del Ser would like to thank the Basque Government for its funding support through the EMAITEK and ELKARTEK programs. Javier Del Ser receives funding support from the Consolidated Research Group MATHMODE (IT1294-19) granted by the Department of Education of the Basque Government.

REFERENCES

[1] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.
[2] R. Caruana, "Multitask learning," *Machine Learning*, vol. 28, no. 1, pp. 41–75, 1997.
[3] Y.-S. Ong and A. Gupta, "Evolutionary multitasking: a computer science view of cognitive multitasking," *Cognitive Computation*, vol. 8, no. 2, pp. 125–142, 2016.
[4] A. Gupta, Y.-S. Ong, and L. Feng, "Insights on transfer optimization: Because experience is the best teacher," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 1, pp. 51–64, 2017.
[5] L. Feng, Y.-S. Ong, A.-H. Tan, and I. W. Tsang, "Memes as building blocks: a case study on evolutionary optimization+ transfer learning for routing problems," *Memetic Computing*, vol. 7, no. 3, pp. 159–180, 2015.
[6] A. Gupta and Y.-S. Ong, "Genetic transfer or population diversification? deciphering the secret ingredients of evolutionary multitask optimization," in *IEEE Symposium Series on Computational Intelligence (SSCI)*, 2016, pp. 1–7.
[7] Y.-W. Wen and C.-K. Ting, "Parting ways and reallocating resources in evolutionary multitasking," in *IEEE Congress on Evolutionary Computation (CEC)*, 2017, pp. 2404–2411.
[8] A. Gupta, Y.-S. Ong, and L. Feng, "Multifactorial evolution: toward evolutionary multitasking," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 3, pp. 343–357, 2015.
[9] Y.-S. Ong, "Towards evolutionary multitasking: a new paradigm in evolutionary computation," in *Computational Intelligence, Cyber Security and Computational Models*. Springer, 2016, pp. 25–26.
[10] T. Bäck, D. B. Fogel, and Z. Michalewicz, *Handbook of evolutionary computation*. CRC Press, 1997.

[11] J. Del Ser, E. Osaba, D. Molina, X.-S. Yang, S. Salcedo-Sanz, D. Camacho, S. Das, P. N. Suganthan, C. A. C. Coello, and F. Herrera, "Bio-inspired computation: Where we stand and what's next," *Swarm and Evolutionary Computation*, vol. 48, pp. 220–250, 2019.

[12] C. Wang, H. Ma, G. Chen, and S. Hartmann, "Evolutionary multitasking for semantic web service composition," 2019, arXiv:1902.06370.

[13] M. Gong, Z. Tang, H. Li, and J. Zhang, "Evolutionary multitasking with dynamic resource allocating strategy," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 5, pp. 858–869, 2019.

[14] Y. Yu, A. Zhu, Z. Zhu, Q. Lin, J. Yin, and X. Ma, "Multifactorial differential evolution with opposition-based learning for multi-tasking optimization," in *IEEE Congress on Evolutionary Computation (CEC)*, 2019, pp. 1898–1905.

[15] A. Gupta, Y.-S. Ong, L. Feng, and K. C. Tan, "Multiobjective multifactorial optimization in evolutionary multitasking," *IEEE Transactions on Cybernetics*, vol. 47, no. 7, pp. 1652–1665, 2016.

[16] B. Manderick, "Fine-grained parallel genetic algorithms," in *Proc. 3rd International Conference on Genetic Algorithms*, 1989, pp. 428–433.

[17] E. L. Lawler, J. K. Lenstra, A. R. Kan, and D. B. Shmoys, *The traveling salesman problem: a guided tour of combinatorial optimization*. Wiley New York, 1985, vol. 3.

[18] L. Zhou, L. Feng, J. Zhong, Z. Zhu, B. Da, and Z. Wu, "A study of similarity measure between tasks for multifactorial evolutionary algorithm," in *Proceedings of the ACM Genetic and Evolutionary Computation Conference Companion*, 2018, pp. 229–230.

[19] A. Gupta, Y.-S. Ong, B. Da, L. Feng, and S. D. Handoko, "Landscape synergy in evolutionary multitasking," in *IEEE Congress on Evolutionary Computation (CEC)*, 2016, pp. 3076–3083.

[20] S. J. Louis and J. McDonnell, "Learning with case-injected genetic algorithms," College of Engineering, University of Nevada, Reno, Tech. Rep., 2004.

[21] B. Da, Y.-S. Ong, L. Feng, A. K. Qin, A. Gupta, Z. Zhu, C.-K. Ting, K. Tang, and X. Yao, "Evolutionary multitasking for single-objective continuous optimization: Benchmark problems, performance metric, and baseline results," 2017, arXiv:1706.03470.

[22] H. Xiao, G. Yokoya, and T. Hatanaka, "Multifactorial pso-fa hybrid algorithm for multiple car design benchmark," in *IEEE International Conference on Systems, Man and Cybernetics (SMC)*, 2019, pp. 1926–1931.

[23] X. Zheng, Y. Lei, M. Gong, and Z. Tang, "Multifactorial brain storm optimization algorithm," in *International Conference on Bio-Inspired Computing: Theories and Applications*. Springer, 2016, pp. 47–53.

[24] H. Song, A. Qin, P.-W. Tsai, and J. Liang, "Multitasking multi-swarm optimization," in *IEEE Congress on Evolutionary Computation (CEC)*, 2019, pp. 1937–1944.

[25] Y. Yuan, Y.-S. Ong, A. Gupta, P. S. Tan, and H. Xu, "Evolutionary multitasking in permutation-based combinatorial optimization problems: Realization with TSP, QAP, LOP, and JSP," in *IEEE Region 10 Conference (TENCON)*, 2016, pp. 3157–3164.

[26] L. Zhou, L. Feng, J. Zhong, Y.-S. Ong, Z. Zhu, and E. Sha, "Evolutionary multitasking in combinatorial search spaces: A case study in capacitated vehicle routing problem," in *IEEE Symposium Series on Computational Intelligence (SSCI)*, 2016, pp. 1–8.

[27] G. Li, Q. Zhang, and W. Gao, "Multipopulation evolution framework for multifactorial optimization," in *Proceedings of the ACM Genetic and Evolutionary Computation Conference Companion*, 2018, pp. 215–216.

[28] L. Zhou, L. Feng, K. Liu, C. Chen, S. Deng, T. Xiang, and S. Jiang, "Towards effective mutation for knowledge transfer in multifactorial differential evolution," in *IEEE Congress on Evolutionary Computation (CEC)*, 2019, pp. 1541–1547.

[29] C. Bierwirth, D. C. Mattfeld, and H. Kopfer, "On permutation representations for scheduling problems," in *International Conference on Parallel Problem Solving from Nature*. Springer, 1996, pp. 310–318.

[30] H. T. Binh, P. D. Thanh, T. B. Trung *et al.*, "Effective multifactorial evolutionary algorithm for solving the cluster shortest path tree problem," in *IEEE Congress on Evolutionary Computation (CEC)*, 2018, pp. 1–8.

[31] E. Alba and B. Dorronsoro, "Solving the vehicle routing problem by using cellular genetic algorithms," in *European Conference on Evolutionary Computation in Combinatorial Optimization*. Springer, 2004, pp. 11–20.

[32] ——, *Cellular genetic algorithms*. Springer Science & Business Media, 2009, vol. 42.

[33] F. Luna, R. M. Luque-Baena, J. Martínez, J. F. Valenzuela-Valdés, and P. Padilla, "Addressing the 5G cell switch-off problem with a multi-objective cellular genetic algorithm," in *IEEE 5G World Forum (5GWF)*, 2018, pp. 422–426.

[34] M. H. Afshar and R. Hajiabadi, "A novel parallel cellular automata algorithm for multi-objective reservoir operation optimization," *Water resources management*, vol. 32, no. 2, pp. 785–803, 2018.

[35] A. Nebro, J. Durillo, F. Luna, B. Dorronsoro, and E. Alba, "Mocell: A cellular genetic algorithm for multiobjective optimization," *International Journal of Intelligent Systems*, vol. 24, no. 7, pp. 726–746, 2009.

[36] M. Bellmore and G. L. Nemhauser, "The traveling salesman problem: a survey," *Operations Research*, vol. 16, no. 3, pp. 538–558, 1968.

[37] J. Grefenstette, R. Gopal, B. Rosmaita, and D. Van Gucht, "Genetic algorithms for the traveling salesman problem," in *International Conference on Genetic Algorithms and their Applications*, 1985, pp. 160–168.

[38] M. Dorigo and L. M. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 53–66, 1997.

[39] S. N. Kumbharana and G. M. Pandey, "Solving travelling salesman problem using firefly algorithm," *International Journal for Research in science & Advanced Technologies*, vol. 2, no. 2, pp. 53–57, 2013.

[40] E. Osaba, X.-S. Yang, F. Diaz, P. Lopez-Garcia, and R. Carballedo, "An improved discrete bat algorithm for symmetric and asymmetric traveling salesman problems," *Engineering Applications of Artificial Intelligence*, vol. 48, pp. 59–71, 2016.

[41] E. Osaba, J. Del Ser, A. Sadollah, M. N. Bilbao, and D. Camacho, "A discrete water cycle algorithm for solving the symmetric and asymmetric traveling salesman problem," *Applied Soft Computing*, vol. 71, pp. 277–290, 2018.

[42] G. Reinelt, "Tsplib: A traveling salesman problem library," *ORSA Journal on Computing*, vol. 3, no. 4, pp. 376–384, 1991.

[43] E. Osaba, R. Carballedo, F. Diaz, E. Onieva, A. Masegosa, and A. Perallos, "Good practice proposal for the implementation, presentation, and comparison of metaheuristics for solving routing problems," *Neurocomputing*, vol. 271, pp. 2–8, 2018.

[44] L. Davis, "Job shop scheduling with genetic algorithms," in *Proceedings of an International Conference on Genetic Algorithms and their Applications*, vol. 140, 1985, pp. 136–140.

[45] E. V. Bonilla, K. M. Chai, and C. Williams, "Multi-task gaussian process prediction," in *Advances in Neural Information Processing Systems*, 2008, pp. 153–160.

[46] R.-E. Precup and R.-C. David, *Nature-Inspired Optimization Algorithms for Fuzzy Controlled Servo Systems*. Butterworth-Heinemann, 2019.

[47] J. Caceres-Cruz, P. Arias, D. Guimarans, D. Riera, and A. A. Juan, "Rich vehicle routing problem: Survey," *ACM Computing Surveys (CSUR)*, vol. 47, no. 2, p. 32, 2015.

[48] C. Pizzuti, "Evolutionary computation for community detection in networks: a review," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 3, pp. 464–483, 2017.

[49] E. Osaba, J. Del Ser, D. Camacho, M. N. Bilbao, and X.-S. Yang, "Community detection in networks using bio-inspired optimization: Latest developments, new results and perspectives with a selection of recent meta-heuristics," *Applied Soft Computing*, vol. 87, p. 106010, 2020.

[50] X.-S. Yang, "Firefly algorithm, stochastic test functions and design optimisation," *International Journal of Bio-Inspired Computation*, vol. 2, no. 2, pp. 78–84, 2010.

[51] ——, "A new metaheuristic bat-inspired algorithm," in *Nature inspired cooperative strategies for optimization (NICSO 2010)*. Springer, 2010, pp. 65–74.

[52] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Advances in engineering software*, vol. 69, pp. 46–61, 2014.

[53] R.-E. Precup, R.-C. David, and E. M. Petriu, "Grey wolf optimizer algorithm-based tuning of fuzzy control systems with reduced parametric sensitivity," *IEEE Transactions on Industrial Electronics*, vol. 64, no. 1, pp. 527–534, 2016.