Multifactorial Evolution: Toward Evolutionary Multitasking

Abhishek Gupta, Yew-Soon Ong, and Liang Feng

***Abstract*—The design of evolutionary algorithms has typically been focused on efficiently solving a single optimization problem at a time. Despite the implicit parallelism of population-based search, no attempt has yet been made to multitask, i.e., to solve multiple optimization problems simultaneously using a single population of evolving individuals. Accordingly, this paper introduces *evolutionary multitasking* as a new paradigm in the field of optimization and evolutionary computation. We first formalize the concept of evolutionary multitasking and then propose an algorithm to handle such problems. The methodology is inspired by biocultural models of *multifactorial inheritance*, which explain the transmission of complex developmental traits to offspring through the interactions of genetic and cultural factors. Furthermore, we develop a cross-domain optimization platform that allows one to solve diverse problems concurrently. The numerical experiments reveal several potential advantages of *implicit genetic transfer* in a multitasking environment. Most notably, we discover that the creation and transfer of refined genetic material can often lead to accelerated convergence for a variety of complex optimization functions.**

***Index Terms*—Continuous optimization, discrete optimization, evolutionary multitasking, memetic computation.**

I. Introduction

EVOLUTIONARY algorithms (EAs) are optimization metaheuristics that work on Darwinian principles of natural selection or survival of the fittest [1]. The algorithm begins with a population of individuals that undergo computational analogues of sexual reproduction and mutation to produce a generation of offspring. The procedure reiterates itself with the aim of preserving genetic material, which makes an individual fitter with respect to a given environment while eliminating that which makes it weaker. The term "environment" is used herein as a metaphor for the landscape of the objective function being optimized.

Manuscript received April 12, 2015; revised June 15, 2015; accepted July 3, 2015. Date of publication July 17, 2015; date of current version May 26, 2016. This work was supported by the Singapore Institute of Manufacturing Technology–Nanyang Technological University (SIMTech-NTU) Joint Laboratory and Collaborative research Programme on Complex Systems under A*Star-TSRP, and the Computational Intelligence Laboratory, NTU. *(Corresponding author: Yew-Soon Ong.)*

A. Gupta and Y.-S. Ong are with the Computational Intelligence Laboratory, School of Computer Engineering, Nanyang Technological University, Singapore 639798 (e-mail: abhishekg@ntu.edu.sg; asysong@ntu.edu.sg).

L. Feng is with the College of Computer Science, Chongqing University, Chongqing 400044, China (e-mail: liangf@cqu.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TEVC.2015.2458037

Over the years, EAs have been successfully applied to solve a variety of optimization problems in science, operations research (OR), and engineering. These problems can generally be classified into two groups: 1) single-objective optimization (SOO) [2], where every point in the search space maps to a scalar objective value, or 2) multiobjective optimization (MOO) [3]–[7], where every point in the search space maps to a vector-valued objective function. In this paper, we introduce a third category of problems that we label as *multifactorial optimization* (MFO). In principle, MFO provides the scope for fully exploiting the implicit parallelism of population-based search, in a manner that has so far remained unexplored in the field of evolutionary computation.

MFO is an *evolutionary multitasking* paradigm that is characterized by the concurrent existence of multiple search spaces corresponding to different tasks (which may or may not be interdependent), each possessing a unique function landscape. *The nomenclature is therefore inspired by the observation that every task contributes a unique factor influencing the evolution of a single population of individuals*. As an illustrative example, imagine simultaneously tackling two distinct real-world supply chain tasks, namely: 1) job shop scheduling (production optimization) and 2) vehicle routing (logistics optimization), in a single evolutionary solver. Such a combination is considered a form of MFO. With this background, the aim of this paper is to first establish MFO as a new concept in optimization and evolutionary computation that differs from the existing ideas of SOO and MOO. Thereafter, we seek to devise a generic EA for MFO, one that is capable of multitasking across multiple optimization problems in the most efficient manner possible.

Following the premise of classical EAs, we resort to the field of evolutionary biology for algorithmic inspiration. In particular, the techniques explored in this paper are inspired by the models of *multifactorial inheritance*, which put forward that complex developmental traits among offspring are influenced by interactions of genetic and cultural factors [8], [9]. In other words, genetic and cultural transmission cannot be treated independently. For example, while what an individual learns may depend on its genotype, the selection acting on the genetic system may be generated or modified by the spread of a cultural trait [10], [11]. These cultural traits typically originate from social learning or from parents passing down certain customs and preferences to their offspring. The computational equivalent of multifactorial inheritance, for the purpose of efficient evolutionary multitasking, is established by considering each optimization task to contribute a distinct environment in

which offspring can be reared. Thus, MFO leads to the coexistence of multiple blocks of cultural bias (or *memes* [12]), one corresponding to each optimization task. The subsequent evolution of encoded individuals in the composite landscape is simulated through an interplay of genetic and cultural factors. While the basic structure of the proposed algorithm is similar to that of a classical EA, it is augmented with features that are borrowed from the models of multifactorial inheritance. Befittingly, the proposed algorithm is referred to hereafter as a *multifactorial evolutionary algorithm* (MFEA).

A strong practical motivation for the MFO paradigm is derived from the rapidly expanding popularity of the cloud computing industry. In particular, we imagine a cloud-based on-demand service providing customers with access to state-of-the-art optimization software. In this regard, we notice that a cloud service faces the natural phenomenon wherein multiple optimization tasks can be received from multiple users at the same time. These tasks can either have similar properties or belong to entirely different domains. Therefore, for the MFEA to be deployable as a cloud-based multitasking engine, it must possess *cross-domain optimization* capabilities (i.e., the ability to simultaneously handle continuous as well as discrete optimization problems). The pursuit for such generality is accomplished in this paper by proposing a *unified representation scheme* in which every variable is simply encoded by a random key between 0 and 1 [13]. While the decoding of such a representation is straightforward for continuous problems, we discuss techniques that encompass various discrete problems as well, thereby ensuring a cross-domain optimization platform. It is, however, emphasized that the concept of MFO is not fundamentally tied to cross-domain multitasking. Domain-specific solvers can indeed be used when all constitutive tasks belong to the same domain.

Details of the notions presented above are organized in this paper as follows. Section II formalizes the concept of MFO. Section III highlights the theoretical distinction between MFO and MOO. In Section IV, the MFEA is described in detail. Numerical examples in continuous optimization are presented in Section V to showcase the benefits of harnessing latent genetic complementarities between tasks, as is facilitated by evolutionary multitasking. Techniques that extend the paradigm to incorporate the field of discrete optimization are discussed in Section VI. Section VII contains further experimental results that highlight the potential of evolutionary multitasking as a cloud-based computational intelligence tool, drawing particular attention to its ability for cross-domain optimization. Finally, some important directions for future research are revealed in Section VIII.

II. Multifactorial Optimization

In this section, we formalize the concept of MFO and describe how the relative performance of an individual in a population is judged in a multitasking environment.

Before proceeding further, it is important to first consider the relationship between constitutive tasks in MFO. It is worthwhile to mention that the notion of evolutionary multitasking does not impose any strict constraint on the intertask relationship. In particular, the tasks of interest may either be distinct, or be interdependent components of a large multicomponent problem. Nevertheless, in this introductory work, our discussion shall be focused on the former case, i.e., when there exists no prior knowledge of any intertask dependencies. In other words, we consider multitasking across optimization problems that have traditionally been viewed independently (as distinct tasks). Thus, the purpose of MFO is not to find optimum trade-offs among the constitutive objective functions. Rather, the goal is to fully and concurrently optimize each task, aided by the implicit parallelism of population-based search.

Within the aforementioned scope, consider a situation wherein K optimization tasks are to be performed simultaneously. Without loss of generality, all tasks are assumed to be minimization problems. The jth task, denoted T_j, is considered to have a search space $\boldsymbol{X}_j$ on which the objective function is defined as $f_j : \boldsymbol{X}_j \rightarrow \mathbb{R}$. In addition, each task may be constrained by several equality and/or inequality conditions that must be satisfied for a solution to be considered feasible.

In such a setting, we define MFO as an evolutionary multitasking paradigm that builds on the implicit parallelism of population-based search with the aim of finding $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{K-1}, \boldsymbol{x}_K\} = \text{argmin}\{f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_{K-1}(\boldsymbol{x}), f_K(\boldsymbol{x})\}$, where $\boldsymbol{x}_j$ is a feasible solution in $\boldsymbol{X}_j$. Herein, each f_j is treated as an additional factor influencing the evolution of a single population of individuals. For this reason, the composite problem is referred to hereafter as a K-factorial problem.

In order to design evolutionary solvers for MFO, it is necessary to formulate a general technique for comparing population members in a multitasking environment. To this end, we first define a set of properties for every individual p_i, where $i \in \{1, 2, \ldots, |P|\}$, in a population P. Note that the individuals are encoded in a unified search space encompassing $\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_K$, and can be decoded into a task-specific solution representation with respect to each of the K optimization tasks. The decoded form of p_i can thus be written as $\{\boldsymbol{x}_1^i, \boldsymbol{x}_2^i, \ldots, \boldsymbol{x}_K^i\}$, where $\boldsymbol{x}_1^i \in \boldsymbol{X}_1, \boldsymbol{x}_2^i \in \boldsymbol{X}_2, \ldots,$ and $\boldsymbol{x}_K^i \in \boldsymbol{X}_K$.

Definition 1 (Factorial Cost): For a given task T_j, the factorial cost Ψ_j^i of individual p_i is given by $\Psi_j^i = \lambda \cdot \delta_j^i + f_j^i$; where λ is a large penalizing multiplier, f_j^i and δ_j^i are the objective value and the total constraint violation, respectively, of p_i with respect to T_j. Accordingly, if p_i is feasible with respect to T_j (zero constraint violation), we have $\Psi_j^i = f_j^i$.

Definition 2 (Factorial Rank): The factorial rank r_j^i of p_i on task T_j is simply the index of p_i in the list of population members sorted in ascending order with respect to Ψ_j.

While assigning factorial ranks, whenever $\Psi_j^a = \Psi_j^b$ for a pair of individuals p_a and p_b, the parity is resolved by random tie-breaking. However, since the performance of the two individuals is equivalent with respect to the jth task, we label them as being *j-counterparts*.

Definition 3 (Scalar Fitness): The list of factorial ranks $\{r_1^i, r_2^i, \ldots, r_K^i\}$ of an individual p_i is reduced to a scalar fitness φ_i based on its best rank over all tasks; i.e., $\varphi_i = 1/\min_{j \in \{1,\ldots,K\}}\{r_j^i\}$.

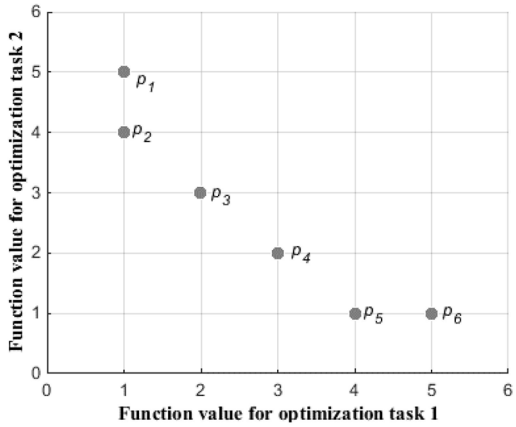


Fig. 1. Sample points in combined objective space of two hypothetical optimization tasks.

Definition 4 (Skill Factor): The skill factor τ_i of p_i is the one task, amongst all other tasks in MFO, on which the individual is most effective, i.e., $\tau_i = \text{argmin}_j\{r_j^i\}$, where $j \in \{1, 2, \ldots, K\}$.

Once the fitness of every individual has been scalarized according to Definition 3, performance comparison can be carried out in a straightforward manner. For example, p_a is considered to dominate p_b in multifactorial sense simply if $\varphi_a > \varphi_b$. We denote this relation between the two individuals as $p_a \gg p_b$. In the event that two individuals have the same skill factor, i.e., $\tau_a = \tau_b = j$, and they also happen to be j-counterparts, we label them as being *strong counterparts*.

It is important to note that the procedure described heretofore for comparing individuals is not absolute. As the factorial rank of an individual (and implicitly its scalar fitness and skill factor) depends on the performance of every other individual in the population, the comparison is in fact population dependent. Nevertheless, the procedure guarantees that if an individual p^* maps to the global optimum of any task, then $\varphi^* \geq \varphi_i$ for all $i \in \{1, 2, \ldots, |P|\}$. Therefore, it can be said that the proposed technique is indeed consistent with the ensuing definition of multifactorial optimality.

Definition 5 (Multifactorial Optimality): An individual p^*, with a list of objective values $\{f_1^*, f_2^*, \ldots, f_K^*\}$, is considered optimum in multifactorial sense iff $\exists j \in \{1, 2, \ldots, K\}$ such that $f_j^* \leq f_j(\boldsymbol{x}_j)$, for all feasible $\boldsymbol{x}_j \in \boldsymbol{X}_j$.

III. MFO and MOO: Similarities and Distinctions

In some cases, it can be argued that the standard EAs for MOO (i.e., MOEAs) are applicable for the purpose of MFO. However, it must be observed that there exists a fundamental difference between the principles of the two paradigms. While MFO aims to leverage the implicit parallelism of population-based search to exploit latent genetic complementarities between multiple tasks, MOEAs attempt to efficiently resolve conflicts among competing objectives of the same task. The concept of Pareto optimality [14], [15] is thus absent in the prescribed scope for MFO as multifactorial optimality does not depend on finding a good tradeoff among different objectives. Instead, it depends on finding the global optimum of at least one constitutive objective function.

Algorithm 1 Basic Structure of the MFEA

1. Generate an initial population of individuals and store it in *current-pop* (P).
2. Evaluate every individual with respect to every optimization task in the multitasking environment.
3. Compute the skill factor (τ) of each individual.
4. **while** (stopping conditions are not satisfied) **do**
   i. Apply genetic operators on *current-pop* to generate an *offspring-pop* (C). Refer to Algorithm 2.
   ii. Evaluate the individuals in *offspring-pop* for selected optimization tasks only (see Algorithm 3).
   iii. Concatenate *offspring-pop* and *current-pop* to form an *intermediate-pop* ($P \cup C$).
   iv. Update the scalar fitness (φ) and skill factor (τ) of every individual in *intermediate-pop*.
   v. Select the fittest individuals from *intermediate-pop* to form the next *current-pop* (P).
5. **end while**

In order to further emphasize the distinction, we refer to the objective space of a hypothetical two-factorial problem depicted in Fig. 1. From the principles of nondominated sorting used in MOEAs [16], [17], it follows that individuals $\{p_2, p_3, p_4, p_5\}$ belong to the first nondominated front while $\{p_1, p_6\}$ belong to the second nondominated front. In other words, individuals $\{p_2, p_3, p_4, p_5\}$ are incomparable to each other and are always preferred over $\{p_1, p_6\}$. However, based on the definitions in Section II, the individuals p_1 and p_2 (and also, p_5 and p_6) are labeled as being strong counterparts. Moreover, $\{p_1, p_2, p_5, p_6\} \gg \{p_3, p_4\}$. In other words, $\{p_1, p_2, p_5, p_6\}$ are considered incomparable to each other in the multifactorial sense and are always preferred over $\{p_3, p_4\}$. Thus, there emerges a disagreement about the relative performance of individuals as deduced from the principles of MOO and MFO.

IV. Multifactorial Evolutionary Algorithm

The MFEA is inspired by the biocultural models of multifactorial inheritance. As the working of the algorithm is based on the transmission of biological and cultural building blocks (genes and memes) [20], [21] from parents to their offspring, the MFEA is regarded as belonging to the realm of *memetic computation* [18], [19]. In particular, cultural effects are incorporated via two features of multifactorial inheritance acting in concert, namely 1) *assortative mating* and 2) *vertical cultural transmission*. Details of these features and their computational analogues shall be discussed herein.

Although the basic structure of the MFEA (presented in Algorithm 1) is similar to that of a classical elitist EA [15], the aforementioned memetic augmentations transform it into an effective MFO solver.

A. Population Initialization

Assume that in K optimization tasks to be performed simultaneously, the dimensionality of the jth task is given by D_j.

Accordingly, we define a unified search space with dimensionality ($D_{\text{multitask}}$) equal to $\max_j\{D_j\}$. During the population initialization step, every individual is thus endowed with a vector of $D_{\text{multitask}}$ random variables (each lying within the fixed range [0, 1]). This vector constitutes the chromosome (the complete genetic material) of that individual. Essentially, the ith dimension of the unified search space is represented by a random key y_i, and the fixed range represents the box-constraint of the unified space. While addressing task T_j, we simply refer to the first D_j random keys of the chromosome. The motivation behind using such an encoding technique, in place of simply concatenating the variables of each optimization task to form a giant chromosome of $D_1 + D_2 + \cdots + D_K$ elements, is twofold.

1) From a practical standpoint, it helps circumvent the challenges associated with the curse of dimensionality when several tasks with multidimensional search spaces are to be solved simultaneously.
2) On theoretical grounds, it is considered to be an effective means of accessing the power of population-based search. As the schemata (or genetic building blocks) [22] corresponding to different optimization tasks are contained within a unified pool of genetic material, they get processed by the EA in parallel. Most importantly, this encourages the discovery and implicit transfer of useful genetic material from one task to another in an efficient manner. Moreover, as a single individual in the population may inherit genetic building blocks corresponding to multiple optimization tasks, the analogy with multifactorial inheritance becomes more meaningful.

B. Genetic Mechanisms

Canonical EAs employ a pair of genetic operators, namely crossover and mutation [23], [24], which are analogous to their biological namesakes. A key feature of the MFEA is that certain conditions must be satisfied for two randomly selected parent candidates to undergo crossover. The principle followed is that of nonrandom or assortative mating [8], [9], which states that individuals prefer to mate with those belonging to the same cultural background. In the MFEA, the skill factor (τ) is viewed as a computational representation of an individual's cultural bias. Thus, two randomly selected parent candidates can freely undergo crossover if they possess the same skill factor. Conversely, if their skill factors differ, crossover only occurs as per a prescribed random mating probability (*rmp*), or else mutation kicks in. Steps for creating offspring according to these rules are provided in Algorithm 2.

The occurrence of assortative mating in the natural world is used in the models of multifactorial inheritance to explain pedigreed traits that extend over several generations [8]. In our case, the parameter *rmp* is used to balance exploitation and exploration of the search space. A value of *rmp* close to 0 implies that only culturally alike individuals are allowed to crossover, while a value close to 1 permits completely random mating. In the former case, the predominantly intracultural mating (i.e., between parent candidates having the same skill factor) and the small genetic variations produced by mutation

Algorithm 2 Assortative Mating

Consider two parent candidates p_a and p_b randomly selected from *current-pop*.

1. Generate a random number *rand* between 0 and 1.
2. **if** ($\tau_a == \tau_b$) or ($rand < rmp$) **then**
   i. Parents p_a and p_b crossover to give two offspring individuals c_a and c_b.
3. **else**
   i. Parent p_a is mutated slightly to give an offspring c_a.
   ii. Parent p_b is mutated slightly to give an offspring c_b.
4. **end if**

(which occurs when parent candidates have different skill factors) facilitate the scanning of confined regions of the search space. As a result, however, there is always the tendency for solutions to get trapped in local optima. In contrast, the increased cross-cultural mating that occurs under larger values of *rmp* (closer to 1) enables enhanced exploration of the entire search space, thereby facilitating escape from local optima. Moreover, exclusive mating between individuals belonging to the same cultural background could lead to the loss of good and diverse genetic material available from other cultural backgrounds. Therefore, the *rmp* must be chosen so as to ensure a good balance between thorough scanning of smaller regions within the search space and exploration of the entire space.

Finally, while choosing crossover and mutation operators, it must be kept in mind that the random keys presented earlier are always interpreted as continuous variables, even if the underlying optimization problem is discrete. This encourages the use of existing real-coded genetic operators, or the design of new operators for improved navigation of the composite landscapes associated with MFO.

C. Selective Evaluation

While evaluating an individual for task T_j, the first step is to decode its random keys into a meaningful input for that task. In other words, the random key representation serves as a unified search space from which problem-specific solution representations can be deduced. For the case of continuous optimization, this is achieved in a straightforward manner. For instance, consider the ith variable (x_i) of the actual problem to be bounded within the range $[L_i, U_i]$. If the corresponding random key of an individual takes a value y_i, then its mapping into the search space of the actual problem is given by $x_i = L_i + (U_i - L_i)\cdot y_i$. In contrast, for the case of discrete optimization, the chromosome decoding scheme is usually problem dependent. In Section VI, we present several illustrations on well-known combinatorial problems.

It goes without saying that evaluating every individual for every problem being solved will often be computationally too expensive, and is therefore undesirable. In order to make MFO computationally practical, the MFEA must be designed to be efficient. This is achieved herein by reducing the total number

Algorithm 3 Vertical Cultural Transmission Via Selective Imitation

An offspring 'c' will either have two parents (p_a and p_b) or a single parent (p_a or p_b)—see Algorithm 2.

1. **if** ('c'has 2 parents) **then**
   i. Generate a random number *rand* between 0 and 1.
   ii. **if** ($rand < 0.5$) **then**
   'c' imitates $p_a \rightarrow$ The offspring is evaluated only for task τ_a (the skill factor of p_a).
   iii. **else**
   'c' imitates $p_b \rightarrow$ The offspring is evaluated only for task τ_b (the skill factor of p_b).
   iv. **end if**
2. **else**
   i. 'c' imitates its single parent $\rightarrow$ The offspring is evaluated only for that task that is its parent's skill factor.
3. **end if**
4. Factorial costs of 'c' with respect to all unevaluated tasks are artificially set to ∞ (a very large number).

of function evaluations by as many as possible. An important observation we have made is that an individual generated in the MFEA is unlikely to be high performing across all tasks. Hence, an individual must ideally be evaluated for only selected tasks on which it is most likely to perform well. For incorporating this feature into the MFEA in a simple manner, we draw inspiration from the memetic concept of vertical cultural transmission [18]. In multifactorial inheritance, vertical cultural transmission is a mode of inheritance that operates side by side with biological inheritance in a manner such that the phenotype of an offspring gets directly affected by the phenotype of its parents [8]–[11]. The computational analogue of the aforementioned phenomenon is realized in the MFEA by allowing offspring to imitate the skill factor (cultural trait) of any one of their parents. This feature, labeled as *selective imitation*, is achieved numerically by following the steps in Algorithm 3. Thus, instead of evaluating an offspring for every task, it is evaluated for only one task.

It is worth noting that the incorporation of cultural effects in the prescribed manner significantly reduces the total number of function evaluations required. In fact, for a K-factorial problem, the function evaluations are reduced almost by a factor of K compared to what would be the case if an individual was evaluated for all tasks.

D. Selection Operation

As shown in Algorithm 1, the MFEA follows an elitist strategy that ensures that the best individuals survive through the generations. In order to identify the best individuals, we employ the technique presented in Section II for comparing population members while multitasking.

E. Summarizing Salient Features of MFEA

Standard EAs typically generate a large population of candidate solutions, all of which are unlikely to be competent for the task at hand. *In contrast, in a multitasking environment, it is intuitively more probable that a randomly generated or genetically modified individual is competent for at least one task. The mechanism of the MFEA builds on this observation by effectively splitting the population into different skill groups, each excelling at a different task. More interestingly and importantly, it is possible that the genetic material created within a particular group turns out to be useful for another task as well. Thus, in such situations, the scope for genetic transfer across tasks can potentially lead to the discovery of otherwise hard to find global optima*. In the MFEA, the transfer of genetic material is facilitated by allowing different skill groups to communicate with each other in a controlled manner, via occasional chromosomal crossover. This is achieved implicitly by two components of the algorithm operating in concert, namely: 1) the *rmp*, which allows individuals with distinct skill factors to mate with some probability, thereby creating a multicultural environment for offspring to be reared in and 2) the fact that the generated offspring can then randomly select a parental skill factor for imitation (see Algorithm 3). We contend that while excessive communication can be disruptive to focused search, prohibiting communication is also undesirable as it constrains exploration and the power of implicit parallelism offered by the entire population.

V. Boon of Multitasking

In theory, multitasking promises several features that are particularly appealing in situations wherein several jobs are posed simultaneously; in our case, each job is an optimization problem to be solved. A case in point is a cloud service provider facing service demands from multiple customers at the same time. The features that are of particular interest in this context include: 1) minimizing the *makespan* (i.e., the time required to complete all jobs) [25], [26] and 2) providing high quality solutions. In this section, we present numerical experiments that show the efficacy of MFO in this respect, thereby encouraging further development of the concept.

We begin by choosing three popular benchmark functions from continuous optimization.

1) Sphere function

$$\sum_{i=1}^{D} x_i^2. \tag{1}$$

2) Ackley's function [27], [28]

$$20 + e - 20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} z_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi z_i)\right); \quad \boldsymbol{z} = \mathbf{M}_A \times (\boldsymbol{x} - \boldsymbol{O}_A). \tag{2}$$

3) Rastrigin's function [27], [28]

$$\sum_{i=1}^{D}\left(z_i^2 - 10\cos(2\pi z_i) + 10\right); \quad \boldsymbol{z} = \mathbf{M}_R \times (\boldsymbol{x} - \boldsymbol{O}_R). \tag{3}$$

Note that in (2) and (3), $\mathbf{M}_A$ and $\mathbf{M}_R$ are randomly generated rotation matrices. Moreover, $\boldsymbol{O}_A$ and $\boldsymbol{O}_R$ are the global optima of the respective functions.

It is well known from the literature that the complexity of the chosen problems is not uniform. The convex sphere function is the simplest, while the multimodal Rastrigin's function is considerably more challenging to optimize. These features provide an ideal platform to demonstrate how genetic material created in relatively simple problems can potentially be utilized to aid the optimization of complex functions.

A. Multitasking Across Functions With Intersecting Optima

In the first set of examples, we examine cases in which the global optima of the chosen functions are intersecting. This is achieved by setting $\boldsymbol{O}_A = \boldsymbol{O}_R = 0$ (origin) and by considering homogenous search spaces for all three functions. The latter condition is satisfied by assuming the extent of the search space to be given by $[-50, 50]$ in every dimension, for all experiments in this subsection. In other words, we ensure that there indeed exists some useful genetic material that can be transferred from one problem to the other.

In the numerical experiments that follow, we consider 20-D and 30-D variants of the functions. These are referred to in shorthand by concatenating the problem dimensionality with the initial letter of the function name, i.e., as 20S, 30S, 20A, 30A, 20R, and 30R. While multitasking, if, for example, 20S and 20A are solved simultaneously, then the composite two-factorial problem is denoted as (20S, 20A). Similarly, the final function values obtained by the algorithm are also presented in the form $(\cdot, \cdot)$. On the other hand, if only a single problem, say 20R, is being solved in the form of SOO, it is denoted as (20R, none).

While organizing the experiments, it is considered important to showcase how the optimization of complex functions, such as 30R or 30A, can be aided by the presence of other optimization tasks in a multitasking environment. To this end, we assemble two families of problem sets as follows, *F1:* {(30R, none), (30R, 30S), (30R, 30A), (30R, 20S)} and *F2:* {(30A, none), (30A, 30S), (30A, 30R), (30A, 20S), (30A, 20R)}. With regard to the MFEA, popular genetic operators, namely, Simulated Binary Crossover (SBX) [29] and Gaussian mutation [30], are used. All results presented in this section are summaries of 30 independent runs of the MFEA for each problem set, under a consistent experimental setup. During each run, a population of 100 individuals is evolved for a maximum of 500 generations or until the solutions converge to the global optimums of the constitutive tasks. To facilitate high quality solutions, every individual in the MFEA undergoes a learning step. For continuous problems, individual learning is carried out via the BFGS quasi-Newton method and follows the principle of Lamarckism [31], [32]. Note that a generated offspring undergoes local improvement only with respect to the skill factor that it imitates. Finally, the value of *rmp* is set to 0.3 in all experiments in order to allow sufficient cross-cultural communication.

From Fig. 2(a), it can be seen that most instances with multitasking lead to significantly improved convergence rate.

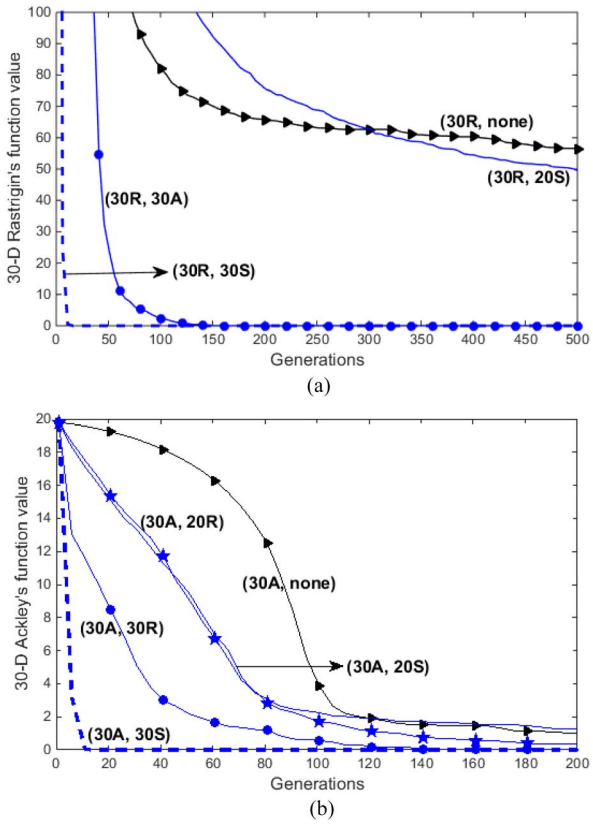


Fig. 2. Convergence trends (averaged over 30 independent runs) for (a) 30-D Rastrigin's function in problem-set $F1$ and (b) 30-D Ackley's function in problem-set $F2$.

Not surprisingly, the curve corresponding to (30R, 30S) converges within the first few generations. This happens because the sphere function gets optimized quasi-instantaneously during the search process, thereby creating refined genetic material that is useful for optimizing Rastrigin's function. A similar explanation can be given for the curve (30R, 30A), which also converges rapidly. Even though Ackley's function is multimodal, it is often considered less challenging to optimize due to its shallow local optima. Accordingly, in the experiment (30R, 30A), Ackley's function tends to converge comparatively faster, which then makes high quality genetic material available for transfer to Rastrigin's function. Finally, in the problem (30R, 20S), the genetic material created while optimizing the sphere function only constitutes two-thirds of what is necessary for Rastrigin's function. Consequently, the convergence rate is restricted. Nevertheless, the convergence curve of (30R, 20S), as depicted in Fig. 2(a), reveals that the partial information provided by the sphere function assists in surpassing the overall performance of (30R, none). Further evidence of improvement in wall-clock time is contained in Table I.

The convergence trends in Fig. 2(b) have similar qualitative characteristics as those presented in Fig. 2(a). A noteworthy observation is that pairing Ackley's function with Rastrigin's function in fact leads to accelerated convergence for Ackley's function as well, as is evidenced by comparing curves (30A, none) and (30A, 30R). This is interesting as Rastrigin's function is known to be more challenging to

TABLE I
AVERAGED FUNCTION VALUES ACHIEVED BY THE MFEA AND NORMALIZED WALL-CLOCK TIME WHEN THE STOPPING CRITERION IS REACHED. NORMALIZATION IS CARRIED OUT WITH RESPECT TO THE WALL-CLOCK TIME FOR (30R, NONE)

Problem Label	Averaged function values achieved	Normalized wall-clock time	Convergence to global optimum
(30R, none)	(54.465, ·)	1	No
(30R, 30S)	**(0, 0)**	**0.0168**	**Yes**
(30R, 30A)	**(0, 0)**	**0.2535**	**Yes**
(30R, 20S)	(49.548, 0)	0.9457	No
(30A, none)	(0.300, ·)	0.8523	No
(30A, 30S)	**(0, 0)**	**0.0184**	**Yes**
(30A, 20S)	(0.551, 0)	0.7656	No
(30A, 20R)	(0.054, 0.099)	0.4756	No

optimize, and is therefore expected to impede the convergence rate. On the contrary, the provision for genetic exchange in the MFEA aids convergence on both tasks. It is postulated that the evolving population successfully exploits the landscape of both functions simultaneously, thereby efficiently bypassing obstacles to converge faster.

Table I reports the averaged function values achieved and the normalized wall-clock time for all numerical experiments carried out in this subsection. In addition to the improved convergence characteristics of MFO, Table I also illustrates the potential benefits of multitasking in terms of minimizing makespan for a batch of optimization tasks. For instance, on comparing the wall-clock time of (30R, none) and (30A, none) to that of (30R, 30A), we find that multitasking leads to a saving of at least 70% relative to SOO. This is primarily because rapid convergence to the intersecting global optima is achieved when the tasks 30R and 30A are combined in MFO. In contrast, convergence is seldom achieved within the computational budget of 500 generations when the tasks are solved solitarily in the form of SOO. A second contributing feature is the selective imitation strategy of the MFEA, according to which the individuals in a multitasking environment select only a single task for evaluation. Thus, for a two-factorial problem like (30R, 30A), the total number of function evaluations is substantially reduced as compared to serially solving (30R, none) and (30A, none). Therefore, on comparing the outcome for (30A, none) to that for (30A, 30S) or (30A, 20S) or (30A, 20R), we find that the time taken to multitask through a batch of optimization tasks is potentially less than the time consumed to solitarily tackle only the single most expensive task in that batch.

B. Multitasking Across Functions With Separated Optima

In the next set of examples, we study cases in which the constitutive tasks in MFO have varying degrees of separation between their respective function optima. Accordingly, we consider the (30R, 30A) composite test problem with $\boldsymbol{O}_R = 0$ (origin) and $\boldsymbol{O}_A$ shifted as follows: 1) *small separation* (*SS*): each element of $\boldsymbol{O}_A$ is randomly shifted between 0 and 2; 2) *medium separation* (*MS*): each element of $\boldsymbol{O}_A$ is randomly shifted between 2 and 5; and 3) *large separation* (*LS*): each element of $\boldsymbol{O}_A$ is randomly shifted between 5 and 10. Moreover, the search spaces corresponding to the two multimodal functions are considered to be heterogeneous. This is achieved by setting the extent of the search space for Ackley's function variables to $[-32, 32]$, and that for Rastrigin's function variables to $[-5, 5]$. Notice that for the large separation case (*LS*), the optimum of the shifted Ackley's function lies outside the bounds of Rastrigin's function variables. A 1-D visualization of the setup is presented in Fig. 3.

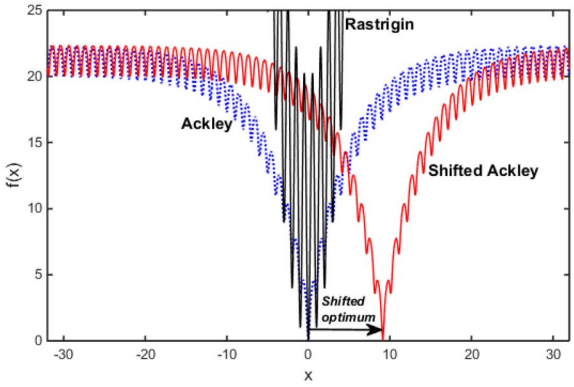


Fig. 3. 1-D illustration of the shifted Ackley's function. Notice that the optimum of the shifted Ackley's function lies outside the bounds of Rastrigin's function variables $\rightarrow [-5, 5]$.

While carrying out the experiments, the algorithmic specifications of the MFEA are kept identical to those stated in the previous section. From the convergence trends depicted in Fig. 4, it can be seen that regardless of the degree of separation between the constitutive function optima, evolutionary multitasking consistently leads to significant performance improvement in comparison to solving Ackley's (or Rastrigin's) function in isolation (in the form of SOO). On close inspection, it can, however, be seen that the effectiveness of multitasking slowly reduces with increasing optima separation [see convergence trends corresponding to *SS*, *MS*, and *LS* in Fig. 4(a) and (b)]. In order to explain this phenomenon, we claim that increasing optima separation weakens the genetic complementarity between the two tasks. Nevertheless, the MFEA exploits underlying synergies that continue to exist between the shifted functions, with its performance gradually approaching that of SOO for larger separations [see *LS* in Fig. 4(a)]. This observation provides indirect evidence of the claim made in Section IV-E (see italicized) that the MFEA effectively splits the population into different skill groups. This feature is particularly crucial when the optima of the constitutive tasks are distantly located from one another. The increased exploration that transpires from controlled cross-cultural mating is found to assist optimization even during cases when there exists no apparent source of genetic complementarity between distinct tasks.

C. Brief Summary

The results presented in this section highlight three major theoretical incentives of incorporating the

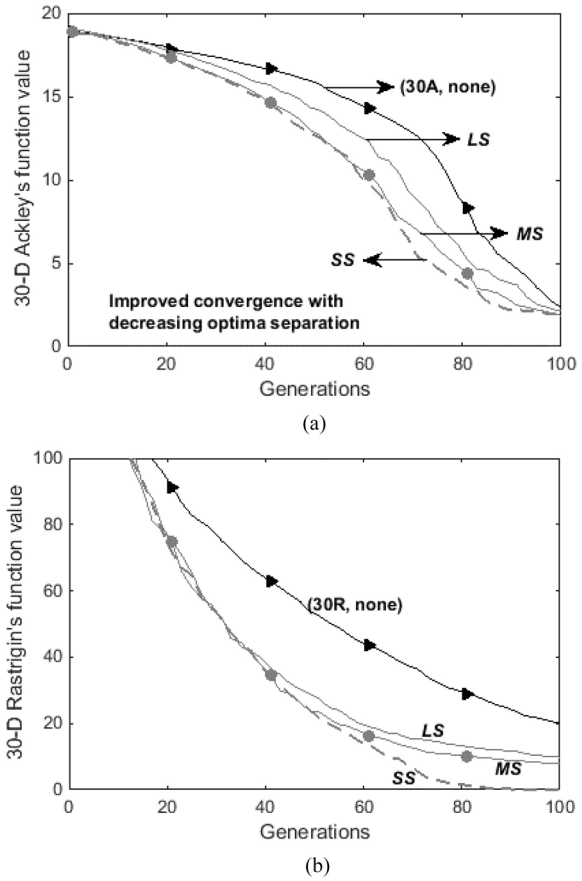


Fig. 4. Averaged convergence trends of (a) 30-D Ackley's function and (b) 30-D Rastrigin's function, for varying degree of optima separation in MFO; *SS*: small separation, *MS*: medium separation, and *LS*: large separation.

concept of multitasking into the field of evolutionary optimization.

1) Implicit genetic transfer from a simple to complex task, in the presence of genetic complementarity, can lead to rapid convergence for complex optimization tasks.
2) Genetic exchange between two complex tasks facilitates simultaneous exploitation of both function landscapes, thereby efficiently evading obstacles to converge faster.
3) The natural phenomenon of vertical cultural transmission is harnessed in a multifactorial setting to potentially reduce wall-clock time for a batch of optimization problems.

VI. Incorporating Discrete Optimization Problems

In the previous sections, we have developed the MFEA and demonstrated its efficacy in the context of continuous optimization. With the ultimate aim of achieving cross-domain multitasking, in this section, we introduce techniques that generalize the MFEA to the case of discrete optimization problems as well. For the sake of brevity, the focus of our discussions shall be placed on combinatorial optimization. It is well known that such problems can occur in numerous flavors that generally necessitate genetic representations that are distinct from one another (as illustrated in Fig. 5), thereby posing a stiff challenge to the generality of the MFEA.

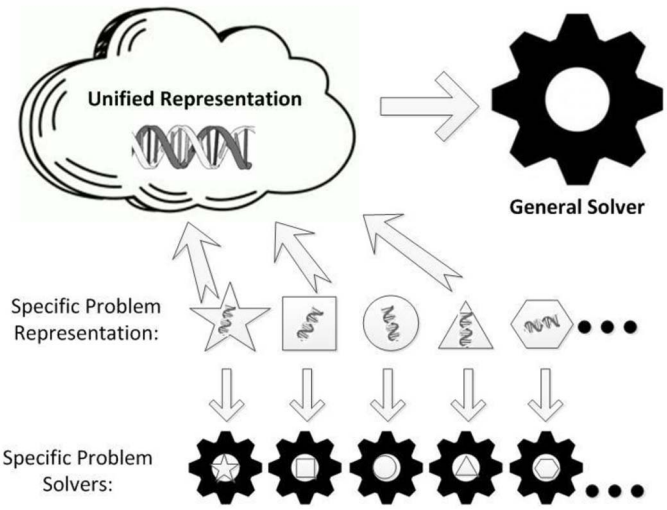


Fig. 5. For cross-domain MFO, domain-specific representations (depicted by different shapes) must be encoded within a unified representation scheme. MFEA, which operates directly on the unified representation, acts as a general solver that replaces domain-specific solvers across continuous and discrete problems.

Typically, a combinatorial problem can be defined by a finite ground state $S = \{1, 2, 3, \ldots, n\}$, which gives rise to a discrete search space consisting of a set of feasible solutions $\boldsymbol{X} \subseteq 2^S$, where each solution maps to a real function value, as $f : 2^S \rightarrow \mathbb{R}$ [33]. This seemingly simple description can lead to a rich variety of problem statements. As the margins of this paper preclude a comprehensive exposition, we select the following well-known combinatorial optimization problems for our subsequent demonstrations: 1) the 0/1 single knapsack problem (KP) and multiple knapsack problem (MKP); 2) the quadratic assignment problem (QAP); and 3) capacitated vehicle routing problem (CVRP).

In the sections to follow, we introduce techniques by which the random keys can be decoded to domain-specific representations of the aforementioned problems.

A. Single and Multiple Knapsack Problems

The KP states that a subset of n items, each with a weight w_i and profit q_i, for $i \in \{1, 2, \ldots, n\}$, are to be filled in a knapsack of capacity W such that it yields the maximum profit. The more general problem with MKP consists of m knapsacks with capacities $W_1, W_2, \ldots, W_m$, respectively, and n items. In this case, the weight of the ith item is given by w_{ik} when it is considered for inclusion in the kth knapsack. In addition, the MKP states that an item is either placed in all knapsacks or in none at all.

In classical EAs for the KP and MKP, an individual solution is typically represented by a vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ of binary variables [34], [35]; where $x_i = 1$ indicates that the ith item is placed in the knapsack(s), and 0 indicates otherwise. One simplistic way of deducing binary variables from random keys is to consider the following: $x_i = 1$ if the value of the random key $y_i \geq 0.5$, else $x_i = 0$ [33].

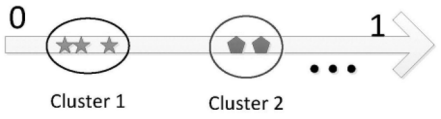


Fig. 6. Random keys are considered to behave as a collection of mobile genes that are clustered into two groups representing items that are either placed in the knapsack(s) or not.

However, through experiments we find that using such a technique generally leads to poor performance of the MFEA. Instead, in this paper, we consider each individual, defined by a vector of random keys $(y_1, y_2, \ldots, y_n)$, to represent a collection of n genes that can move freely on a segment of the real line, i.e., between 0 and 1. Each gene corresponds to a unique item. The free mobility afforded to the genes implies that their positions can be manipulated through real-coded operators, as is desired for the MFEA. The binary variables are deduced from this picture by considering that the gene collection forms two clusters, as depicted in Fig. 6. For the purpose of identifying the clusters, the single-linkage algorithm is used in this paper. Depending upon the size of the knapsack(s), one of the clusters is selected such that all its items are provisionally included in the knapsack(s). All items in the other cluster are excluded.

It may so happen that the total weight of items in the chosen cluster violates the capacity constraint of the knapsack(s). In such cases, we consider Dantzig's greedy approximation algorithm [34] in which items are considered for deletion in the order of increasing efficiency, i.e., their profit to weight ratio. Note that the repair is performed in the spirit of Baldwinian learning [18].

B. Quadratic Assignment Problem

The QAP [36] aims to assign n facilities to n locations, one to each location, such that a quadratic objective function is minimized. The random keys can be decoded for the QAP in a straightforward manner, as has been proposed in [13]. Note that the ith random key of an individual is viewed as a label tagged to the ith facility. In order to assign facilities to locations, the random keys are simply sorted in ascending order. Thereafter, facilities are assigned to locations according to their position in the sorted list. For example, let us consider a set of three locations, labeled {1, 2, 3}, and a set of three facilities represented by random keys: (0.7, 0.1, 0.3). According to the prescribed assignment technique, Facility 2 is assigned to Location 1, Facility 3 to Location 2, and Facility 1 to Location 3.

With regard to local search, the two-exchange heuristic [37] is used to improve a candidate assignment. As the changes introduced can be reincorporated into the chromosome by simply interchanging the random key labels of facilities, learning in the QAP occurs in the spirit of Lamarckism.

C. Capacitated Vehicle Routing Problem

The CVRP is defined on a graph $G(V, E)$, where V is a set of nodes and E is a set of arcs. One node on the graph represents a vehicle depot, while all others represent customers with certain demands. The purpose of solving the CVRP is to find trip schedules for a fleet of vehicles such that the cost of servicing all the customers is minimized. The cost is generally given by the total distance traveled by the vehicles. In the trip schedule, it must be ensured that the total customer demand serviced by a particular vehicle does not exceed its capacity. Moreover, there often exists an additional constraint in which the customers prescribe a hard time-window during which they prefer to be serviced [38].

Several genetic representation schemes have been proposed by EA researchers for the CVRP. These include: 1) the variable-length chromosome scheme [39] and 2) the giant-tour scheme [40], [41], etc.; the latter being a permutation of all customers in the form of a travelling salesman (TSP) route with no trip delimiters. Similarly, several random key decoding procedures can also be conceived for the CVRP. In this paper, we consider the ith random key to label the ith customer. Thereafter, a TSP route is deduced by simply sorting the random keys in ascending order (like for the case of QAP). Next, the TSP route is partitioned into individual vehicle trips based on a *split* procedure proposed in [40] (details are omitted for the sake of brevity). Finally, individuals undergo a set of simple local search moves [42], in the spirit of Lamarckian learning, to improve solution quality.

D. Illustration of Cross-Domain Random Key Decoding

In order to summarize the cross-domain decoding procedures described heretofore, we present an illustrative example in which three distinct optimization tasks are to be performed simultaneously. These are: 1) a continuous problem with five decision variables, each bounded within the range [0, 5]; 2) a KP with 10 items; and 3) a QAP with 10 facilities (and 10 locations).

For the aforementioned three-factorial problem, $D_{\text{multitask}} = \max\{5, 10, 10\} = 10$. Therefore, we randomly generate a ten-dimensional chromosome $\boldsymbol{y}$ with the following random key representation: (0.79, 0.31, 0.53, 0.17, 0.60, 0.26, 0.65, 0.69, 0.75, 0.45). Fig. 7 depicts the manner in which the same chromosome can be decoded for all three optimization tasks. For the continuous problem, the process is relatively straightforward. The first five random keys of $\boldsymbol{y}$ are simple extracted and linearly mapped from [0, 1] to [0, 5].

For the KP, we refer to the real line shown in Fig. 7. The random keys have been plotted along the line, together with the index of the corresponding gene. The single-linkage algorithm is used to divide the genes into two clusters, with the first cluster consisting of genes {4, 6, 2} and the second cluster consisting of genes {10, 3, 5, 7, 8, 9, 1}. According to the recommended decoding procedure for KPs, we can consider the items {4, 6, 2} to be provisionally included in the knapsack, while all items in the other cluster are excluded. In other words, the domain-specific chromosome representation for the KP works out to (0 1 0 1 0 1 0 0 0 0). Thus, we hereafter refer to the decoding procedure as *binary clustering*.

Finally, for the QAP, the sequence of gene indices (as appears along the real line in Fig. 7) prescribes the order in which facilities are assigned to locations. For the given $\boldsymbol{y}$, the domain-specific representation thereby equates

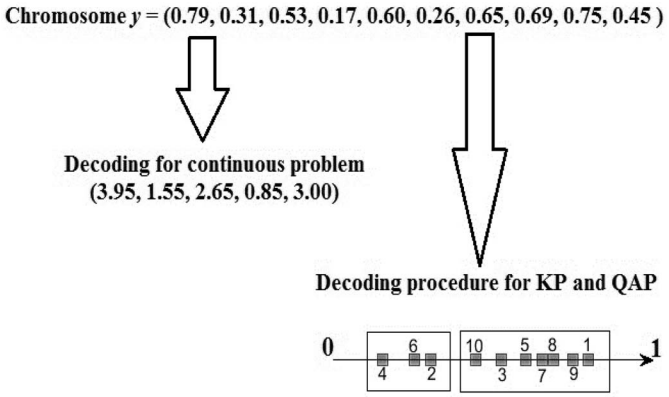


Fig. 7. Illustrative example of decoding a random key chromosome into different domain-specific representations.

to (4, 6, 2, 10, 3, 5, 7, 8, 9, 1). An interesting feature of the random key representation scheme, particularly in the case of *sequencing problems* such as the QAP and the CVRP, is that it ensures offspring feasibility under real-coded crossover and mutation operations. This is in contrast to domain-specific representations of such problems, wherein specially designed genetic operators are needed to ensure offspring feasibility (see [13] and [23] for more details). The aforementioned observation was first made in [10], and has since led to notable interest among EA practitioners in the field of OR [33].

VII. Numerical Experiments and Discussions

The aim of this section is twofold. In Section VII-A, we consider four sets of MFO examples with diverse properties. These provide a more comprehensive understanding of the performance of the MFEA, particularly in the context of a potential cloud-based software service receiving diverse jobs from multiple customers at the same time. The experiments have been designed by combining popular benchmarks instances from discrete and continuous optimization. It is stated that there exists no *a priori* knowledge of discernible genetic complementarity between the constituent tasks. This simulates the realistic situation of a cloud server faced with multiple jobs (or tasks) with unknown characteristics.

In Section VII-B, we also briefly discuss the scenario in which all tasks to be solved via MFO are not presented at exactly the same time. This leads to dynamism in the sense that a new task can be presented to a solver in the intermediate stages of tackling a different task. The motivation behind this hypothetical scenario is derived from the observation that cloud customers are naturally expected to be distributed in space and in time. It shall be demonstrated that a notable implication of MFO in such situations is that genetic material from the current task can conceivably be utilized to jump-start the optimization process of the intermediately presented task.

A. MFO for Discrete and Cross-Domain Optimization

The MFEA is tested on a variety of composite problem sets, each comprising three tasks. Broadly speaking, these problem sets can be classified into three categories. Set 1 is composed of sequencing problems (namely, QAP and CVRP) for which the random keys must be sorted while decoding. Tasks in Set 2 are similar in the sense that their respective random key decoding procedures call for binary clustering of the genes. Finally, Sets 3 and 4 are true examples of cross-domain multitasking as they combine instances from sequencing, binary clustering, and continuous optimization. Results obtained by the MFEA on the aforementioned problem sets are reported herein. Each set is a three-factorial optimization environment with the three constitutive tasks drawn from popular benchmark instances of the respective domains (see Table II for details).

TABLE II
Background Details of Composite Problem Sets Considered

	Task 1 (T_1)		Task 2 (T_2)		Task 3 (T_3)	
	Label	Optimum	Label	Optimum	Label	Optimum
Set 1	QAP Chr22b [43]	6.1940e+3	CVRP A-n65-k9 [44]	1.1740e+3	QAP Chr25a [43]	3.7960e+3
Set 2	MKP (n=1000)	n.a.	MKP (n=100)	n.a.	KP (n=100)	5.9100e+2
Set 3	Rastrigin's 30-D $-5 \le x_i \le 5$	0	QAP Chr22a [43]	6.1560e+3	KP (n=250)	1.5290e+3
Set 4	KP (n=1000)	5.9860e+3	CVRP B-n63-k10 [44]	1.4960e+3	Ackley's 30-D $-32 \le x_i \le 32$	0

n.a. = not available.

It has been stated previously that the KP and MKP are naturally maximization problems. Accordingly, in the MFEA, which is a global minimizer, we consider the negative of the corresponding objective function values. Also note that the KP and MKP instances are randomly generated. While optimum solutions to the KP instances have been obtained via dynamic programming, optimum solutions to MKP are not available.

The procedure for generating a KP instance is as follows:

$$\text{Let } w_i = \text{uniformly random } [1, 10] \text{ and } q_i = w_i + 5.$$

Accordingly, the capacity of an average knapsack is given by: $W = (1/2)\sum_{i=1}^{n} w_i$.

For the case of MKP, a similar procedure is used, except that five knapsacks are considered. Thus

$$w_{ik} = \text{uniformly random } [1, 10] \text{ and } q_i = \text{mean}_k(w_{ik}) + 5.$$

The capacity of the kth knapsack is given by: $W_k = (1/2)\sum_{i=1}^{n} w_{ik}$.

In order to judge the output of the MFEA in terms of quality and robustness, the following parameters are considered.

1) *Average (Avg):* The averaged function value across all runs of the MFEA.
2) *Coefficient of Variation (CV):* Ratio of standard deviation to the averaged function value.
3) *Average Gap (AG):* Difference between the Average and the optimum value. Presented as a percentage of the optimum.
4) *Best-Found (BF):* Best function value achieved across all runs of the MFEA.
5) *Gap:* Difference between the Best-found and the optimum. Presented as a percentage of the optimum.

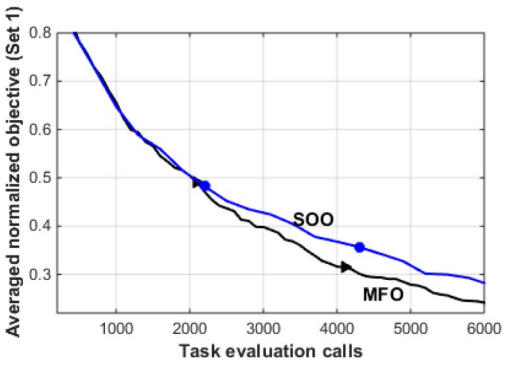


Fig. 8. Convergence trends of $\tilde{f}$ in MFO and serial SOO for Set 1 (QAP, CVRP, and QAP) during initial stages of evolution.

In all experiments that follow, a population of 100 individuals is evolved over 500 generations, i.e., the stopping criterion is configured to the completion of approximately 50 000 task evaluations, where a single "task evaluation" call includes the Lamarckian/Baldwinian learning steps that the individual undergoes. The same chromosome representation scheme and genetic operators are used for MFO as well as SOO simulations. Specifically, SBX (crossover) and Gaussian mutation operators are employed for navigating the continued search space, with *rmp* set to 0.3 in the MFEA.

In the convergence trends presented hereafter, the objective function values for task T_j, where $j \in \{1, 2, 3\}$, are first normalized as $\tilde{f}_j = (f_j - f_j^{\min})/(f_j^{\max} - f_j^{\min})$. Here, $f_j^{\min}$ and $f_j^{\max}$ are the minimum and maximum function values discovered for T_j across all test runs (i.e., including MFO and SOO). Next, the normalized values are averaged across the three tasks in the set; $\tilde{f} = (\tilde{f}_1 + \tilde{f}_2 + \tilde{f}_3)/3$. Finally, $\tilde{f}$ is further averaged over 30 independent runs of the solver.

1) Performance on Sample Set 1 (Sequencing Problems): Fig. 8 depicts the averaged convergence trends during the initial stages of the MFEA for Set 1. The curve is compared against that obtained from an SOO solver. It is clear that the curve corresponding to the MFO simulation surpasses the averaged SOO performance, showing better overall convergence characteristics. As the same encoding scheme and genetic operators are used in SOO and MFO, the improvement can be attributed entirely to the exploitation of multiples function landscapes via implicit genetic transfer, as is afforded by the evolutionary multitasking paradigm.

In terms of the quality of solutions produced during multitasking, the results presented in Table III show that the MFEA performs consistently well. We find that the algorithm reliably outputs optimum solutions to the two QAP instances (T_1, T_3). For the CVRP instance, a small gap of 1.1783% to the optimum (or best-known solution) is attained on average.

2) Performance on Sample Set 2 (Binary Clustering Problems): Set 2 comprises a large MKP instance (T_1: consisting of 1000 items) and two other relatively small instances (T_2 and T_3: with 100 items each). The convergence trends in Fig. 9 once again demonstrate the effectiveness of evolutionary multitasking in comparison to serial SOO. Moreover, from the final results presented in Table IV, it can be seen that the optimum solution to the small KP instance (T_3) is consistently achieved on every run of the MFEA.

TABLE III
AVERAGED PERFORMANCE OF THE MFEA ON SET 1 PROBLEMS AT THE END OF 500 GENERATIONS

Set 1	T_1	T_2	T_3
Avg (CV)	6.2331e+3 (0.4919%)	1.1878e+3 (0.4321%)	3.7960e+3 (0)
AG	0.6318%	1.1783%	0
BF	6.1940e+3	1.1770e+3	3.7960e+3
Gap	0	0.2555%	0

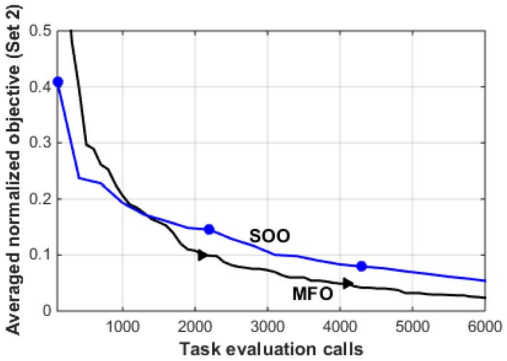


Fig. 9. Convergence trends of $\tilde{f}$ in MFO and serial SOO for Set 2 (MKP, MKP, and KP) during initial stages of evolution.

TABLE IV
AVERAGED PERFORMANCE OF THE MFEA ON SET 2 PROBLEMS AT THE END OF 500 GENERATIONS

Set 2	T_1	T_2	T_3
Avg (CV)	4.1280e+3 (0)	3.9900e+2 (0)	5.9100e+2 (0)
AG	n.a.	n.a.	0
BF	4.1280e+3	3.9900e+2	5.9100e+2
Gap	n.a.	n.a.	0

3) Performance on Sets 3 and 4 (Cross-Domain Multitasking): Sets 3 and 4 comprise instances from sequencing, binary clustering, and continuous optimization, thereby creating a truly cross-domain multitasking environment. The efficacy of MFO in such settings is highlighted by the superior convergence trends in comparison to serial SOO, as revealed in Figs. 10 and 11. The final results for Set 3, reported in Table V, show that the MFEA successfully converges to the optimum of each of the three problem variants (Rastrigin's function, QAP, and KP). This observation provides strong evidence on the robustness of the MFEA under diverse multitasking environments, thereby encouraging its potential application as a cloud-based computation intelligence tool.

Analyzing the results for Set 4, as presented in Table VI, it can be seen that the MFEA is capable of finding the optimum solution to the large KP instance (T_1: 1000 items), while simultaneously ensuring high-quality solutions for the CVRP and the continuous optimization task. Convergence to the global

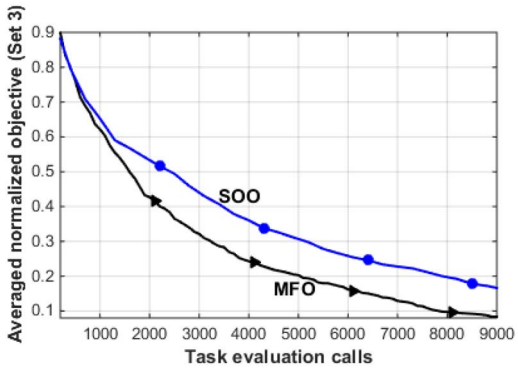


Fig. 10. Convergence trends of $\tilde{f}$ in MFO and serial SOO for Set 3 (Rastrigin's, QAP, and KP) during initial stages of evolution.

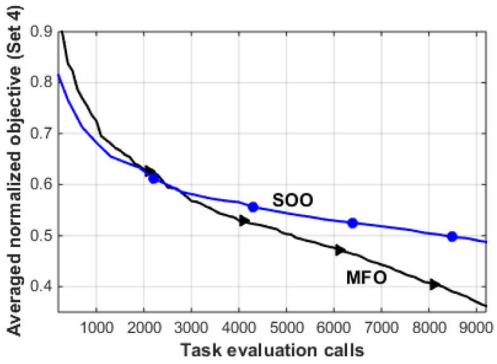


Fig. 11. Convergence trends of $\tilde{f}$ in MFO and serial SOO for Set 4 (KP, CVRP, and Ackley's) during initial stages of evolution.

TABLE V
AVERAGED PERFORMANCE OF THE MFEA ON SET 3 PROBLEMS AT THE END OF 500 GENERATIONS

Set 3	T_1	T_2	T_3
Avg *(CV)*	0 (0)	6.1613e+3 (0.2273%)	1.5289e+3 (0.0477%)
AG	0	0.0866%	0.0087%
BF	0	6.1560e+3	1.5290e+3
Gap	0	0	0

TABLE VI
AVERAGED PERFORMANCE OF THE MFEA ON SET 4 PROBLEMS AT THE END OF 500 GENERATIONS

Set 4	T_1	T_2	T_3
Avg *(CV)*	5.9812e+3 (0.0786%)	1.5237e+3 (0.7718%)	0.4543 (1.4800e+2%)
AG	0.0807%	1.8516%	0.4543
BF	5.9860e+3	1.5040e+3	0
Gap	0	0.5348%	0

optimum of Ackley's function (T_3) is achieved on a regular basis. Moreover, notice that the average gap for the CVRP instance (T_2) is also small.

In order to better understand the improved performance as a consequence of MFO, we refer to Fig. 12. The figure depicts the convergence trends corresponding to each individual task in the cross-domain multitasking examples (Sets 3 and 4).

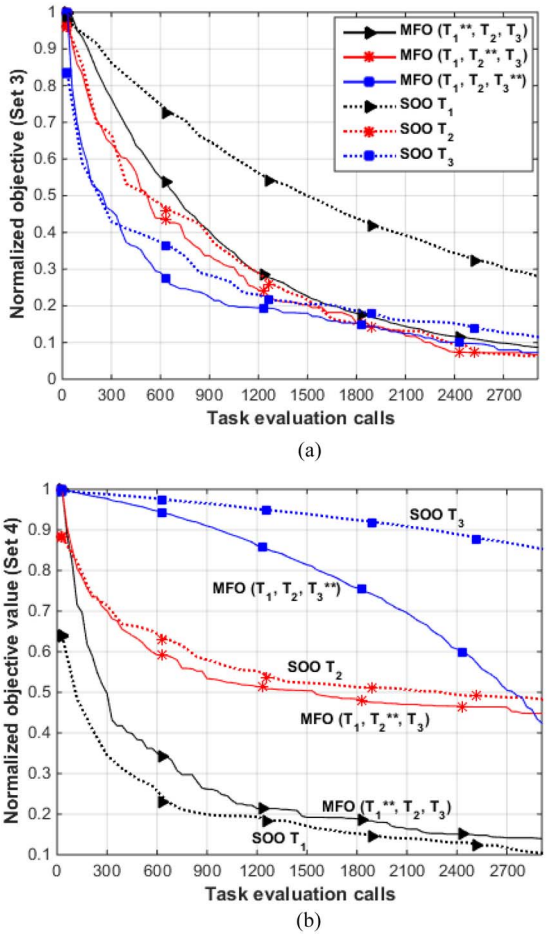


Fig. 12. Comparing convergence trends of $\tilde{f}_1$, $\tilde{f}_2$, and $\tilde{f}_3$ in MFO and serial SOO for (a) Set 3 and (b) Set 4. Note that the curve corresponding to a particular task in MFO is labeled by appending asterisks (**) to the task label.

Note that the curve corresponding to a particular task in MFO is labeled by appending asterisks (**) to the task label. For example, the curve for T_2 is labeled as MFO (T_1, T_2^{**}, and T_3). From Fig. 12(a) (for Set 3), we find that the convergence rate is accelerated for each of the constitutive tasks in MFO. While the acceleration is less conspicuous for the combinatorial instances (namely, T_2 and T_3), it is particularly pronounced for the continuous optimization problem (T_1: Rastrigin's function). Similar inferences can be drawn from Fig. 12(b) (for Set 4), where the optimization of Ackley's function (T_3) is found to be most benefited. On the other hand, it is also seen in Fig. 12(b) that, in comparison to SOO, convergence for the large KP instance (T_1) is slightly impeded. This observation leads to the conclusion that evolutionary multitasking may not necessarily guarantee performance improvement for every task, since not all genetic transfer is always useful. In other words, while certain tasks are positively impacted by the implicit genetic transfer available during multitasking, certain other tasks may be negatively impacted [45], [46]. Nevertheless, in practice, since there is expected to be little *a priori* knowledge about the genetic complementarity between tasks, it makes good sense to allow evolution to autonomously harness the latent complementarity when available, without the need to explicitly identify and

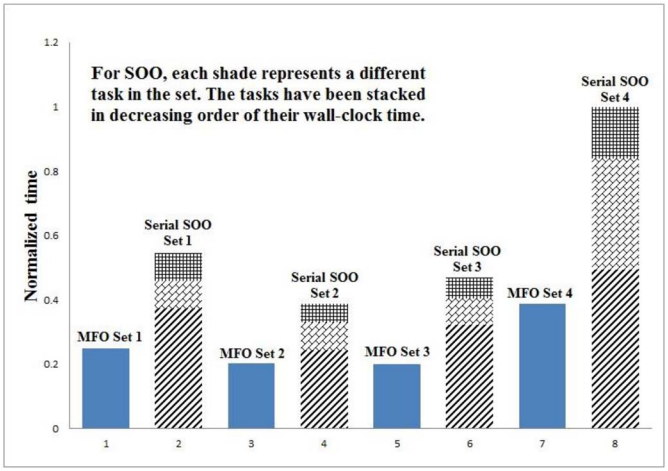


Fig. 13. Comparing the wall-clock time of the MFEA and serial SOO for a batch of optimization tasks (Sets 1–4).

inject domain knowledge into the MFEA. Notably, our results suggest that positive transfer considerably outweighs negative transfer in many cross-domain optimization problems in MFO.

4) Makespan Minimization (Byproduct of MFO): An added advantage of the MFO paradigm is the minimization of makespan (the time required for completing all jobs) for a batch of optimization problems. In particular, our experiments on Sets 1–4 reveal that the makespan achieved by the MFEA on a single machine can often be less than the makespan achieved when tasks are distributed over multiple machines executing SOO solvers. Notably, this performance enhancement is attained by simply unlocking the inherent parallelism of evolutionary search.

In Fig. 13, we compare the total wall-clock time of the MFEA (given a stopping criterion of 50 000 task evaluations) against serial SOO (given the same stopping criterion per task) for Sets 1–4. Notice that the total number of task evaluations performed in the MFEA is not scaled in proportion to the number of tasks being handled. As a consequence of the selective imitation strategy (which prescribes that individuals in a multitasking environment must select only a single task for evaluation), we find that the total time required by the MFEA to concurrently optimize a batch of tasks is measurably less than the time spent by the SOO solver on only the single most expensive task. Accordingly, if we assume that the SOO solvers are deployed in parallel on multiple machines, the most expensive task creates a bottleneck impeding the makespan for the entire batch. Thus, it is contended that the makespan achievable via MFO is potentially less than that achieved by multiple SOO solvers running in parallel (with each solver executing one task).

B. MFO for Asynchronous Optimization Tasks

Optimum scheduling of jobs is a serious challenge for cloud service providers. Within a short duration of time, multiple jobs can be received from multiple users. In this section, we briefly consider the case in which a new optimization task is presented to an MFO solver in the intermediate stages of tackling a different task. In particular, we wish to show that

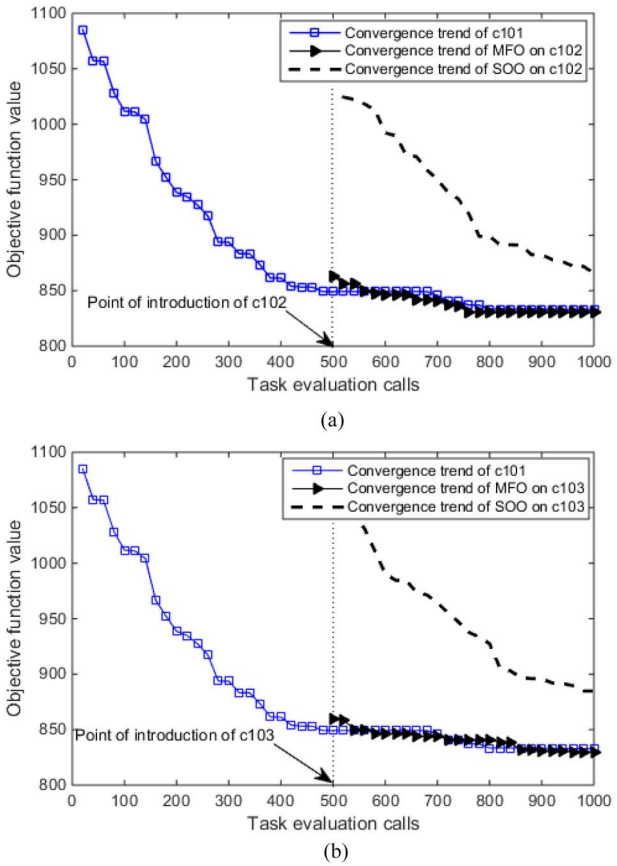


Fig. 14. Boon of MFO in a dynamic setting. (a) Improvement in convergence trend of CVRP instance c102 and (b) improvement in convergence trend of CVRP instance c103.

the provision for implicit genetic transfer can be invaluable for improving performance in such dynamic settings.

To this end, we theorize an example in capacitated vehicle routing. Imagine a scenario where different logistics service providers within the same city have similar geographical distribution of customers. In such cases, it is expected that there exist structural similarities in the optimum vehicle routes that can be positively transferred from one problem to the other. For the sake of exposition, it is assumed that the logistic companies present their respective vehicle routing problems to the MFO solver in reasonably quick succession.

In order to design a computational analogue of the aforementioned situation, we consider three time-window constrained CVRP instances (c101, c102, and c103) from the popular Solomon's benchmark set [38]. The interesting feature of these sample instances is that while they possess similar distribution of customers in 2-D Euclidean space, the time-window constraints specified by the customers are different. A dynamic environment is artificially created by assuming the evolutionary process to initiate with instance c101 in isolation. The subsequent instance, c102 (or c103), is then presented to the solver at an intermediate stage. Note that since all the instances belong to the CVRP domain, the MFO solver we use in this case employs a domain-specific representation scheme [41]. The advantage of implicit genetic transfer, leading to the exploitation of solution similarities, is clearly demonstrated in Fig. 14. The gene pool created

at the intermediate stages of optimizing c101 gets positively transferred to the subsequent task. A significant impetus is thus provided to the evolutionary search process for instances c102 and c103, as compared to an independent SOO solver.

VIII. Conclusion

In this paper, we have introduced MFO as a new paradigm in evolutionary computation. It is contended that while traditional population-based EAs focus on solving a single problem at a time, suitably harnessing the implicit parallelism of these methods can lead to effective multitasking engines with the ability to solve multiple problems simultaneously. The term "multifactorial" is in fact inspired from the observation that every additional task contributes an extra factor influencing evolution.

With the aim of efficient evolutionary multitasking, a new MFEA has been proposed based on the biocultural concept of multifactorial inheritance. The algorithm is endowed with a unified chromosome representation scheme that is shown to be the key ingredient for handling cross-domain multitasking problems. Moreover, the unified search space implies that the building blocks corresponding to different tasks are contained within a unified pool of genetic material. In particular, this allows the MFEA to harness latent genetic complementarities between tasks, leading to the discovery and implicit transfer of useful genetic material from one task to another in an efficient manner.

Several numerical experiments were carried out to test the efficacy of the proposed algorithm. The experiments can be categorized into continuous problems, discrete optimization, and a mixture of continuous and discrete optimization. The results show that evolutionary multitasking provides some promising outcomes in terms of accelerating convergence for complex optimization functions and minimizing makespan for a batch of optimization tasks. A deeper analysis to understand the convergence characteristics of individual tasks in cross-domain multitasking showed that while certain tasks are positively impacted by the implicit genetic transfer, there may exist certain other tasks that are negatively impacted. However, in most cases, we find positive transfer to outweigh the negative, thereby leading to improved convergence trends when the performance is averaged over all tasks.

Although the results of this paper have been encouraging, it must be kept in mind that this paper is only a first step in a so far unexplored research direction. Rigorous investigation of several practical and theoretical aspects of the paradigm is needed in the future. For instance, there exists a vast body of literature in the field of discrete optimization that remains to be addressed. Furthermore, the current methods of the MFEA are yet to be thoroughly tested for scalability. The latter is expected to be a critical issue in evolutionary multitasking, especially from the perspective of cloud computing, as the number of tasks to be tackled is likely to grow very rapidly. The population of individuals searching the unified search space must therefore be intelligently manipulated to accommodate large volumes of diverse incoming tasks in the most efficient manner possible.

Finally, we draw the attention of the reader to the assumption made thus far that there exists no prior knowledge of any relationship between the tasks being solved in tandem. The efficacy of the MFEA has been showcased while multitasking across distinct problems that have traditionally been viewed independently. However, it is worth highlighting here that the applicability of MFO is not restricted to such scenarios. It can often happen that a single complex problem is composed of several interdependent subproblems or components [47]–[49]. MFO can conceivably be adapted to solve these components simultaneously. As another example, imagine the problems are precedence constrained, which implies that certain tasks must utilize the results from certain other tasks [50]. Clearly, an improved MFO solver capable of handling such challenges will be of significant value in numerous practical applications.

References

[1] T. Back, U. Hammel, and H. P. Schwefel, "Evolutionary computation: Comments on the history and current state," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 3–17, Apr. 1997.

[2] S. M. Guo and C. C. Yang, "Enhancing differential evolution using eigenvector-based crossover operator," *IEEE Trans. Evol. Comput.*, vol. 19, no. 1, pp. 31–49, Feb. 2014.

[3] C. A. C. Coello, "Evolutionary multi-objective optimization: A historical view of the field," *IEEE Comput. Intell. Mag*., vol. 1, no. 1, pp. 28–36, Feb. 2006.

[4] C. M. Fonseca and P. J. Fleming, "An overview of evolutionary algorithms in multi-objective optimization," *Evol. Comput*., vol. 3, no. 1, pp. 1–6, Dec. 2007.

[5] L. Tang and X. Wang, "A hybrid multiobjective evolutionary algorithm for multiobjective optimization problems," *IEEE Trans. Evol. Comput*., vol. 17, no. 1, pp. 20–45, Feb. 2012.

[6] K. Li, A. Fialho, S. Kwong, and Q. Zhang, "Adaptive operator selection for a multiobjective evolutionary algorithm based on decomposition," *IEEE Trans. Evol. Comput*., vol. 18, no. 1, pp. 114–130, Feb. 2013.

[7] M. Asafuddoula, T. Ray, and R. Sarker, "A decomposition-based evolutionary algorithm for many objective optimization," *IEEE Trans. Evol. Comput*., vol. 19, no. 3, pp. 445–460, Jun. 2014.

[8] J. Rice, C. R. Cloninger, and T. Reich, "Multifactorial inheritance with cultural transmission and assortative mating. I. Description and basic properties of the unitary models," *Amer. J. Hum. Genet*., vol. 30, pp. 618–643, Nov. 1978.

[9] C. R. Cloninger, J. Rice, and T. Reich, "Multifactorial inheritance with cultural transmission and assortative mating. II. A general model of combined polygenic and cultural inheritance," *Amer. J. Hum. Genet*., vol. 31, pp. 176–198, Mar. 1979.

[10] L. L. Cavalli-Sforza and M. W. Feldman, "Cultural versus biological inheritance: Phenotypic transmission from parents to children (A theory of the effect of parental phenotypes on children's phenotypes)," *Amer. J. Hum. Genet*., vol. 25, pp. 618–637, Nov. 1973.

[11] M. W. Feldman and K. N. Laland, "Gene-culture coevolutionary theory," *Trends Ecol. Evol*., vol. 11, pp. 453–457, Nov. 1996.

[12] R. Dawkins, *The Selfish Gene*. Oxford, U.K.: Oxford Univ. Press, 1976.

[13] J. C. Bean, "Genetic algorithms and random keys for sequencing and optimization," *ORSA J. Comput*., vol. 6, no. 2, pp. 154–160, 1994.

[14] M. Ehrgott, *Multicriteria Optimization*. Berlin, Germany: Springer, 2005.

[15] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*. New York, NY, USA: Wiley, 2001.

[16] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multi-objective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput*., vol. 6, no. 2, pp. 182–197, Apr. 2002.

[17] X. Zhang, Y. Tian, R. Cheng, and Y. Jin, "An efficient approach to nondominated sorting for evolutionary multiobjective optimization," *IEEE Trans. Evol. Comput*., vol. 19, no. 2, pp. 201–213, Apr. 2014.

[18] X. Chen, Y. S. Ong, M. H. Lim, and K. C. Tan, "A multi-facet survey on memetic computation," *IEEE Trans. Evol. Comput*., vol. 15, no. 5, pp. 591–606, Oct. 2011.

[19] Y. S. Ong, M. H. Lim, X. S. Chen, "Research frontier: Memetic computation—Past, present & future," *IEEE Comput. Intell. Mag*., vol. 5, no. 2, pp. 24–36, May 2010.

[20] M. Iqbal, W. N. Browne, and M. Zhang, "Reusing building blocks of extracted knowledge to solve complex, large-scale Boolean problems," *IEEE Trans. Evol. Comput*., vol. 18, no. 4, pp. 465–480, Aug. 2013.
[21] R. Mills, T. Jansen, and R. A. Watson, "Transforming evolutionary search into higher-order evolutionary search by capturing problem structure," *IEEE Trans. Evol. Comput*., vol. 18, no. 5, pp. 628–642, Oct. 2014.
[22] A. H. Wright, M. D. Vose, and J. E. Rowe, *Implicit Parallelism* (LNCS 2724), Berlin, Germany: Springer, 2003, pp. 1505–1517.
[23] D. E. Goldberg, *Genetic Algorithms in Search Optimization and Machine Learning*, Reading, MA, USA: Addison Wesley, 1989.
[24] M. Srinivas and L. M. Patnaik, "Genetic algorithms: A survey," *Computer*, vol. 27, no. 6, pp. 17–26, Jun. 1994.
[25] Z. Wu, Z. Ni, and X. Liu, "A revised discrete particle swarm optimization for cloud workflow scheduling," in *Proc. Int. Conf. Comput. Intell. Security (CIS)*, Nanning, China, 2010, pp. 184–188.
[26] C. Lin and S. Lu, "Scheduling scientific workflows elastically for cloud computing," in *Proc. IEEE Int. Conf. Cloud Comput. (CLOUD)*, Washington, DC, USA, 2011, pp. 746–747.
[27] Y. S. Ong, Z. Z. Zong, and D. Lim, "Curse and blessing of uncertainty in evolutionary algorithms using approximation," *IEEE CEC*, Vancouver, BC, Canada, 2006, pp. 2928–2935.
[28] P. Chauhan, K. Deep, and M. Pant, "Novel inertia weight strategies for particle swarm optimization," *Memetic Comput.*, vol. 5, no. 3, pp. 229–251, 2013.
[29] K. Deb and R. B. Agrawal, "Simulated binary crossover for continuous search space," *Complex Syst.*, vol. 9, no. 2, pp. 115–148, 1995.
[30] R. Hinterding, "Gaussian mutation and self-adaption for numeric genetic algorithms," *IEEE CEC*, vol. 1. Perth, WA, Australia, 1995, pp. 384–389.
[31] R. Meuth, M. H. Lim, Y. S. Ong, and D. C. Wunsch, "A proposition on memes and meta-memes in computing for higher-order learning," *Memetic Comput.*, vol. 1, no. 2, pp. 85–100, 2009.
[32] Y. S. Ong and A. J. Keane, "Meta-Lamarckian learning in memetic algorithms," *IEEE Trans. Evol. Comput*., vol. 8, no. 2, pp. 99–110, Apr. 2004.
[33] J. F. Goncalves and M. G. C. Resende, "Biased random-key genetic algorithms for combinatorial optimization," *J. Heuristics*, vol. 17, no. 5, pp. 487–525, 2011.
[34] S. Bansal, C. Patvardhan, and A. Srivastav, "Quantum-inspired evolutionary algorithm for difficult knapsack problems," *Memetic Comput.*, vol. 7, no. 2, pp. 135–155, 2015.
[35] S. Khuri, T. Back, and J. Heitkotter, "The zero/one multiple knapsack problem and genetic algorithms," in *Proc. ACM Symp. Appl. Comput*., Phoenix, AZ, USA, 1994, pp. 188–193.
[36] M. H. Tayarani-N and A. Prugel-Bennett, "On the landscape of combinatorial optimization problems," *IEEE Trans. Evol. Comput*., vol. 18, no. 3, pp. 420–434, Jun. 2013.
[37] R. K. Ahuja, J. B. Orlin, and A. Tiwari, "A greedy genetic algorithm for the quadratic assignment problem," *Comput. Oper. Res.*, vol. 27, no. 10, pp. 917–934, 2000.
[38] M. M. Solomon, "Algorithms for the vehicle routing and scheduling problems with time window constraints," *Oper. Res.*, vol. 35, no. 2, pp. 254–265, 1987.
[39] K. C. Tan, Y. H. Chew, and L. H. Lee, "A hybrid multi-objective evolutionary algorithm for solving vehicle routing problems with time windows," *Comput. Optim. App*., vol. 34, no. 1, pp. 115–151, 2006.
[40] C. Prins, "A simple and effective evolutionary algorithm for the vehicle routing problem," *Comput. Oper. Res.*, vol. 31, no. 12, pp. 1985–2002, 2004.
[41] A. Gupta, Y. S. Ong, A. N. Zhang, and P. S. Tan, "A bi-level evolutionary algorithm for multi-objective vehicle routing with time window constraints," in *Proc. 18th Asia Pac. Symp. Intell. Evol. Syst.*, Singapore, vol. 2. 2014, pp. 27–38.
[42] C. Prins, "Two memetic algorithms for heterogeneous fleet vehicle routing problems," *Eng. App. Artif. Intell*., vol. 22, no. 6, pp. 916–928, 2009.
[43] N. Christofides and E. Benavent, "An exact algorithm for the quadratic assignment problem," *Oper. Res.*, vol. 35, no. 9, pp. 760–768, 1989.
[44] P. Augerat *et al.*, "Computational results with a branch and cut code for the capacitated vehicle routing problem," Universite Joseph Fourier, Grenoble, France, Tech. Rep. 949-M, 1998.
[45] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng*., vol. 22, no. 10, pp. 1345–1359, Oct. 2009.
[46] L. Feng, Y. S. Ong, M. H. Lim, and I. W. Tsang, "Memetic search with inter-domain learning: A realization between CVRP and CARP," *IEEE Trans. Evol. Comput.*, to be published.
[47] M. R. Bonyadi, Z. Michalewicz, F. Neumann, and M. Wagner, *Evolutionary Computation for Multi-Component Problems: Opportunities and Future Directions*. [Online]. Available: https://cs.adelaide.edu.au/~zbyszek/Papers/Vis.pdf, accessed Mar. 15, 2015.
[48] M. R. Bonyadi, Z. Michalewicz, and L. Barone, "The travelling thief problem: The first step in the transition from theoretical problems to realistic problems," in *Proc. IEEE CEC*, Cancun, Mexico, 2013, pp. 1037–1044.
[49] S. D. Handako *et al.*, "Solving multi-vehicle profitable tour problem via knowledge adoption in evolutionary bilevel programming," in *Proc. IEEE CEC*, Sendai, Japan, 2015, pp. 2713–2720.
[50] M. R. Garey, "Optimal task sequencing with precedence constraints," *Discrete Math.*, vol. 4, no. 1, pp. 37–56, 1973.

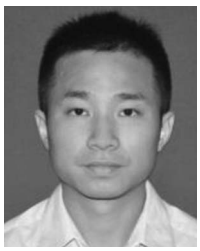
Abhishek Gupta received the Ph.D. degree in engineering science from University of Auckland, Auckland, New Zealand, in 2014.

He is currently a Research Fellow with the Computational Intelligence Graduate Laboratory, School of Computer Engineering, Nanyang Technological University, Singapore. His research interests include continuum mechanics as well as computational intelligence, with recent emphasis on evolutionary computation, multiobjective optimization, and bi-level programming.

Yew-Soon Ong received the Ph.D. degree on artificial intelligence in complex design from the Computational Engineering and Design Center, University of Southampton, Southampton, U.K., in 2003.

He is a Professor with the School of Computer Engineering, Nanyang Technological University, Singapore. His research interests include computational intelligence spans across memetic computing, evolutionary design, machine learning, and big data.

Dr. Ong is the Founding Technical Editor-in-Chief of *Memetic Computing Journal*, the Founding Chief Editor of the Springer book series *Studies in Adaptation, Learning, and Optimization*, and an Associate Editor of IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, *IEEE Computational Intelligence Magazine*, IEEE TRANSACTIONS ON CYBERNETICS, *Soft Computing*, and *International Journal of System Sciences*.



Liang Feng received the Ph.D. degree in computational intelligence and machine learning from the School of Computer Engineering, Nanyang Technological University, Singapore, in 2014.

He was a Post-Doctoral Research Fellow with the Computational Intelligence Graduate Laboratory, Nanyang Technological University. He is currently an Assistant Professor with the College of Computer Science, Chongqing University, Chongqing, China. His research interests include computational and artificial intelligence, memetic computing, big data optimization, and learning, as well as transfer learning.