Cognitive Computation (2019) 11:1–17
https://doi.org/10.1007/s12559-018-9620-7

Back to the Roots: Multi-*X* Evolutionary Computation

Abhishek Gupta[1,2] · Yew-Soon Ong[2]

Received: 3 October 2018 / Accepted: 11 December 2018 / Published online: 3 January 2019

Abstract
Over the years, evolutionary computation has come to be recognized as one of the leading algorithmic paradigms in the arena of global black box optimization. The distinguishing facets of evolutionary methods, inspired by Darwin's foundational principles of natural selection, stem mainly from their population-based search strategy—which gives rise to the phenomenon of *implicit parallelism*. Precisely, even as an evolutionary algorithm manipulates a population of a few candidate solutions (or: individuals), *it is able to simultaneously sample*, *evaluate*, *and process a vast number of regions of the search space*. This behavior is in effect analogous to our inherent cognitive ability of processing diverse information streams (such as sight and sound) with apparent simultaneity in different regions of our brain. For this reason, evolutionary algorithms have emerged as the method of choice for those search and optimization problems where a collection of multiple target solutions (that may be scattered throughout the search space) are to be found in a single run. With the above in mind, in this paper we return to the roots of evolutionary computation, with the aim of shedding light on a variety of problem settings that are uniquely suited for exploiting the implicit parallelism of evolutionary algorithms. Our discussions cover established concepts of multi-objective and multi-modal optimization, as well as new (schema) theories pertaining to emerging problem formulations that entail multiple searches to be carried out at once. We capture associated research activities under the umbrella term of *multi-X evolutionary computation*, where *X*, as of now, represents the following list: {"objective," "modal," "task," "level," "hard," "disciplinary," "form"}. With this, we hope that the present position paper will serve as a catalyst for effecting further research efforts into such areas of optimization problem-solving that are well-aligned with the fundamentals of evolutionary computation; in turn prompting the steady update of the list *X* with new applications in the future.

Keywords Multi-*X* evolutionary computation · Population-based search · Implicit parallelism · Schema theorem

Introduction

Whenever we are faced with some hypothetical set of alternatives to choose from, finding the best option among them often involves an optimization problem to be solved. To make this statement more concrete, consider the formulation of a general numerical optimization problem that can be stated as follows:

$$\text{maximize}_{\boldsymbol{y}\in Y}\ f(\boldsymbol{y}) \tag{1}$$

✉ Yew-Soon Ong
asysong@ntu.edu.sg

[1] Singapore Institute of Manufacturing Technology (SIMTech), Agency of Science, Technology and Research (A-STAR), Singapore 138634, Singapore

[2] School of Computer Science and Engineering, Nanyang Technological University, Singapore 639798, Singapore

Here, f is an objective function considered for maximization, and Y is the search space of all possible choices that are available to us. Accordingly, $\boldsymbol{y}$ is a *point* (denoted as a vector of decision variables) in the space Y, representing a specific choice. In most real-world optimization examples there are often additional restrictions placed on how one can choose $\boldsymbol{y}$. These constitute constraint functions that must be satisfied for a candidate solution to be considered feasible. The existence of such constraint functions has however been omitted from Eq. (1) for simplicity.

The ubiquity of the type of problem expressed by Eq. (1) has led to immense interest among scientists and engineers for decades, resulting in the development of a plethora of computational techniques for tackling them. While there exists a sea of associated algorithms to choose from, the interest of the present paper lies in a family of nature-inspired optimization methodologies that make up the field of *evolutionary computation* [1, 2]. As the name suggests, the algorithms belonging to this field—referred to as *genetic algorithms* or *evolutionary algorithms* (GAs/EAs for short)—draw inspiration from the foundational principles of evolutionary biology as laid out by Charles Darwin in his 1859 book "On the Origin of Species" [3]. The key distinguishing feature of EAs, relative to conventional (point-based) methods, is that they are able to avoid local optima by concurrently processing multiple regions of the search space through a *population-based search* strategy. To be precise, EAs employ a set of candidate solutions (i.e., a *population of individuals*) that are stochastically updated (*evolved*) from iteration to iteration (*generation* to *generation*) following principles loosely inspired by processes of biological evolution—namely, crossover, mutation, and natural selection—with the goal of guiding the population towards regions of the search space characterized by high objective function (*fitness*) values. As EAs are largely agnostic to the mathematical properties of the objective (and constraint) functions at hand, they can generally be applied with ease across a wide variety of black box search and optimization tasks—with the promise of returning near-optimal results. What is more, if prior information about a problem—in the form of analytical expressions for the objective and constraint functions, or heuristic information about the likely structure of optimal solutions [4, 5]—is available beforehand, then that can easily be incorporated within an EA to endow it with improved performance guarantees [6].

The motivation behind emulating nature (particularly evolutionary processes) for solving optimization problems stems from the observation that populations of living organisms demonstrate consummate problem-solving ability. In fact, the simple mechanism of natural selection has the power to eliminate one of the greatest hurdles in software development—i.e., the need to painstakingly specify all the features of a problem and the actions a program must take to deal with them [7]. In light of the above, it is recognized that one of the keys to the success of computational analogues of evolution, namely EAs, is the emergent phenomenon of *implicit parallelism* [8]—which is obtained as a consequence of its population-based search strategy. Simply put, the implicit parallelism of EAs allows them to concurrently sample, evaluate, and process a vast number of regions of the overall search space while manipulating a relatively small number of individuals—with natural selection kicking-in such that the fraction of individuals in a region grows at a rate proportional to the statistical estimate of its fitness (as shall be substantiated in "Holland's Schema Theorem: Implicit Parallelism in Canonical EAs"). Overall, EAs have the effect of simultaneously guiding multiple individuals through the search space, avoiding local optima to gradually converge towards those areas that tend to maximize the population's expected objective function value. Under this setting, the modified optimization problem statement can be stated as [9, 10],

$$\text{maximize}_{p(y)} \int_Y f(y) \cdot p(y) \cdot dy. \tag{2}$$

In the above, $p(\boldsymbol{y})$ represents the underlying population distribution. It can be seen that while Eq. (1) presents a conventional point-based formulation of optimization problems, Eq. (2) generalizes the same to the context of population-based search distributions.

Beyond standard problems of the type formulated in Eq. (1), the implicit parallelism of EAs is uniquely well-suited for addressing a range of practical scenarios in which the search is no longer limited to finding a single solution that (approximately) maximizes a scalar-valued objective function. Indeed, the ability of EAs to simultaneously evaluate multiple regions of the search space implies (a) that they are less likely to get trapped at a poor local optimum in a complex (rugged) objective function landscape, and, perhaps more interestingly (b) *that they can find multiple (possibly scattered) target solutions in a single run*. The most commonly cited class of problems that heavily exploit the latter property of EAs is *multi-objective* optimization [11]—where the goal is to find multiple trade-off solutions that offer an optimal compromise between conflicting objectives of interest. Thus, it comes as little surprise that the subject of multi-objective optimization has long been a stronghold of evolutionary computation research [12]. Further to this, the class of *multi-modal* optimization problems—where multiple global and local optima of a function are sought—provides another setting where the mechanisms of EAs have been known to thrive [13]. Apart from the usual suspects, there are also a handful of recent works demonstrating the efficacy of population-based search across a series of new (relatively under-explored) domains. Among them, the most instructive examples include *multi-task* [14], *multi-level* [15, 16], *multi-hard* [17, 18], *multi-disciplinary* [19], and *multi-form* [20, 21] optimization problems.

With the above in mind, the aim of this position paper is essentially to return to the roots of evolutionary computation—drawing the attention of upcoming researchers to the unique strengths of EAs, and population-based search in general, that makes them conceptually distinguished from other optimization procedures in the literature. To this end, we provide an overview of latest research activities in the field that can be encapsulated under the umbrella term of *multi-X evolutionary computation*, where *X*, as of now, represents the following list: {"objective," "modal," "task," "level," "hard," "disciplinary," "form"}. What is more, we present simple theoretical analyses that serve to substantiate that EAs are indeed compatible with problems of the multi-*X* type—*which are characterized by multiple distinct searches to be progressed jointly/simultaneously.* It is our hope that this paper will catalyze further research efforts into such areas of optimization problem-solving that are aligned with the fundamentals of evolutionary computation;

in turn prompting the steady expansion of the list *X* with new applications over time.

As an important aside, we venture that the implicit parallelism of EAs may in effect be analogous to our inherent cognitive ability of absorbing multiple distinct information streams with apparent simultaneity—which is achieved by distributing processes over different regions of the brain. For instance, while watching a video, we are able to process images (in the visual cortex of the brain) and sounds (in the auditory cortex of the brain) and synchronize them seamlessly with little effort. Thus, we conjecture that combining evolutionary concepts with cognitively-inspired algorithmic enhancements may have future implications for further realizing the potential of EAs. Some existing work in this direction can be found in [22–24]. In fact, it has been demonstrated that a cognitive multi-tasking interpretation of evolutionary computation—in the context of *multi-task evolutionary optimization*—greatly improves search efficiency when simultaneously dealing with multiple *related* optimization problems [25].

With this, the remainder of the paper is organized as follows. "Holland's Schema Theorem: Implicit Parallelism in Canonical EAs" establishes the basics of implicit parallelism in canonical EAs via a brief introduction to *Holland's schema theorem* [26]—which is widely regarded as the fundamental theorem of EAs. "An Overview of Early Members of the Multi-*X* Family" presents an overview of multi-objective and multi-modal evolutionary optimization—which are among the earliest beneficiaries of the implicit parallelism of evolutionary methods. Thereafter, "Multi-*task* Evolutionary Optimization: an Emerging Paradigm" is dedicated to a detailed summary on the cognitively rooted topic of multi-task evolutionary optimization—which serves as a key concept showcasing the modern utility of the implicit parallelism of a population. In particular, we theoretically study the efficacy of EAs with regard to simultaneously progressing multiple distinct searches—as arises in multi-tasking—via a generalization of the schema theorem to the case of multi-task optimization. Next, in "Other Recent Additions to the Multi-*X* Family," we position the spotlight on other emerging topics that further highlight the advantages of the distinctive facets of evolutionary computation. The paper is concluded in "Conclusion" with closing remarks and some preliminary thoughts on research directions for the future.

Holland's Schema Theorem: Implicit Parallelism in Canonical EAs

In this section, we start with a simple generational model of a canonical EA, as shown in Algorithm 1. The model incorporates standard steps of offspring creating (through genetic operators like crossover and/or mutation) followed by a computational analogue of natural selection in order to gradually guide an evolving population towards favorable regions of the search space.

Algorithm 1 Pseudocode of a canonical EA

```
1.  set $t = 0$
2.  *initialize* population $P(t)$
3.  *evaluate* individuals in $P(t)$
4.  **while termination condition not satisfied do**
5.          set $t = t + 1$
6.          *select* parent population $P'(t)$ from $P(t - 1)$
7.          apply *genetic operators* on $P'(t)$ to get offspring population $P(t)$
8.          *evaluate* individuals in $P(t)$
9.  **end**
10. return best individual(s) found
```

It is noteworthy that the simple steps outlined in Algorithm 1 give rise to the powerful phenomenon of implicit parallelism. To elaborate, even as an EA manipulates only a few individuals in its population, it manages to simultaneously sample, evaluate, and process a vast number of regions of the search space. This can be intuitively seen by considering a hypothetical individual in an EA population that is encoded as the bit string 10011011. Notably, this individual is simultaneously a member of the regions 10******, **011***, ******11, etc.; where the symbol * indicates that the bit is unspecified and can take any value. By convention, each of these regions is represented as a *schema* (or *genetic building-block*) [7], such that the schema 10****** describes the set of all possible candidate solutions with a 1 and a 0 in the first and second positions, respectively. The shorter the length of a schema (in turn representing the largest regions with many unspecified bits), the larger the fraction of the population that will contain it—at least during the initial stages of the search.

In addition to sampling diverse regions of the search space, the selection mechanism of an EA (step 6 of Algorithm 1) has the effect of exploiting the evaluated results in a manner that leads to the growth of the number of individuals in a region at a rate proportional to the statistical estimate of its fitness. This claim follows directly from Holland's schema theorem which has been widely studied in the past for explaining the performance of EAs [26]. For Algorithm 1, under assumptions of fitness proportionate selection, single-point crossover, and no mutation, the mathematical statement corresponding to the theorem may be written as [27],

$$E[m(H,t+1)] \geq m(H,t)\frac{f(H,t)}{\overline{f}(t)}\left[1-p_c\frac{\delta(H)}{l-1}\right]. \tag{3}$$

Here, $E[m(H,t+1)]$ is the *expected number* of individuals at generation $t+1$ containing schema H; $m(H,t)$ is the existing number of individuals in the population that contain schema H; $f(H,t)$ is the average fitness of individuals containing schema H at generation t; $\overline{f}(t)$ is the average fitness of the entire population at generation t; p_c is the crossover probability; $\delta(H)$ is the length of schema H; l is the length of the entire string encoding candidate solutions. *Note that we limit the fitness measure to be positive and strictly increasing with the objective function.*

Given the above, we may define the *growth rate* of schema H from generation t to generation $t+1$ as,

$$gr(H,t) = \frac{E[m(H,t+1)]}{m(H,t)}. \tag{4}$$

Thus, for the case of fitness proportionate selection, and under the simplifying condition that the probability of schema disruption / creation due to crossover can be ignored when $\delta(H) \ll l$ (which implies $p_c\frac{\delta(H)}{l-1}$ is small), the growth rate becomes,

$$gr(H,t) \approx \frac{f(H,t)}{f(t)}. \tag{5}$$

It is clear from Eq. (5) that as long as $f(H,t) > \overline{f}(t)$, the frequency of the corresponding schema will continue to grow in the population as $gr(H,t) > 1$. Further, for any pair of distinct schemata H_a and H_b (that may represent arbitrary regions of the search space) having average fitness $f(H_a, t)$ and $f(H_b, t)$, respectively, with $f(H_a, t) > f(H_b, t)$, the frequency of H_a tends to grow faster than that of H_b. Specifically, on discounting the possibility of H_a and H_b being created from scratch due to crossover, $f(H_a, t) > f(H_b, t) \Rightarrow gr(H_a, t) > gr(H_b, t)$. In [28], these results were further generalized to any admissible EA that may incorporate other kinds of popularly used selection schemes (such as rank-based selection); thereby supporting the general convergence behavior of EAs.

The key message to be drawn from the theoretical results above is the ability of EAs to simultaneously process multiple distinct regions of the search space (with the number of regions being significantly greater than the number of individuals in its population), while gradually focusing the search on those areas that are characterized by high fitness values. It is this property that has been termed as implicit parallelism, and serves as the main distinguishing advantage of EAs over other optimization techniques. What is more, it is this property that has laid the foundation for multi-*X* evolutionary computation, as shall be discussed hereafter.

An Overview of Early Members of the Multi-*X* Family

In this section, we briefly describe problem settings that were among the earliest to fully exploit the implicit parallelism of population-based search. Precisely, multi-objective and multi-modal optimization problems are classical examples that illustrate the efficacy of simultaneously progressing multiple searches via EAs. We note that much research on these topics continues to the present-day due to their immense practical relevance.

Multi-*Objective* Evolutionary Optimization

Over recent decades, the subject of multi-objective optimization problems (MOPs) has drawn considerable attention in the field of evolutionary computation. As a result, EAs have come to be deployed as practically useful tools in a number of real-world applications involving multiple objectives of interest [29–32]. In fact, such has been the success of evolutionary methods towards tackling MOPs that researchers have even considered *multi-objectivizing* standard single-objective problems, and then applying EAs for solving them [33]. The basic mathematical formulation of such MOPs can be stated as follows [34, 35],

$$\text{maximize}_{\boldsymbol{y}\in Y}\ \boldsymbol{f}(\boldsymbol{y}) = [f_1(\boldsymbol{y}), f_2(\boldsymbol{y}), \ldots, f_K(\boldsymbol{y})]. \tag{6}$$

Here, K is the total number of objective functions, and we consider maximization without loss of generality. Further, we recognize the likely existence of additional constraint functions that have been suppressed in Eq. (6).

One of the main reasons for the popularity of EAs in dealing with problems of the type expressed in Eq. (6) is that they offer an excellent platform for the power of implicit parallelism to come to the fore. This is because MOPs are generally characterized by multiple optimum solutions, each offering a distinct compromise between possibly conflicting objectives. As has previously been established, the population-based search strategy of EAs is naturally adept at simultaneously sampling, evaluating, and processing multiple regions of the search space. Thus, they are intrinsically well equipped for addressing MOPs [34].

In order to illustrate the existence of multiple optimum solutions in MOPs, we first clarify what it means for a solution to be considered optimum in the multi-objective sense. To this end, consider any pair of solutions $\boldsymbol{y}, \boldsymbol{y}' \in Y$. We say that $\boldsymbol{y}$ *dominates* $\boldsymbol{y}'$, denoted as $\boldsymbol{y} \succ \boldsymbol{y}'$, iff $\forall i \in \{1, 2, \ldots, K\}$: $f_i(\boldsymbol{y}) \geq f_i(\boldsymbol{y}')$, and $\exists j \in \{1, 2, \ldots, K\} : f_j(\boldsymbol{y}) > f_j(\boldsymbol{y}')$. Accordingly, we label a solution $\boldsymbol{y}^*$ as being optimal (or more precisely: *Pareto optimal*) for an MOP if $\nexists \boldsymbol{y} \in Y$ such that $\boldsymbol{y} \succ \boldsymbol{y}^*$ [36]. From this description it is clear that in any MOP with conflicting objectives, there will exist at least K Pareto optimal solutions corresponding to the maximum of each of the K objective functions. In addition, there will usually exist several other Pareto optimal solutions that offer a trade-off between the objectives. Given the set of all Pareto optimal solutions (or *Pareto set* for short), their mapping into the objective space constitutes what is known as the *Pareto front* [37].

Outline of Methodologies

A variety of multi-objective evolutionary algorithms (MOEAs) have been proposed in the literature. Broadly, these can be classified into three categories [11], viz., (a) *dominance-based MOEAs*, (b) *indicator-based MOEAs*, and (c) *decomposition-based MOEAs*.

Dominance-based MOEAs are among the most widely used in practice, with a popular example of this category being the classic NSGA-II algorithm [38]. The core mechanism of the procedure is to employ a rank-based selection scheme that gradually biases the population towards the Pareto set. To elaborate, at every generation t of NSGA-II, the current population $P(t)$ is first divided into *non-dominated fronts* NF_1, NF_2, …, NF_n such that: (a) $\cup_{i=1:n} NF_i = P(t)$ and $NF_i \cap NF_j = \phi \; \forall \, i, j$, given $i \neq j$; (b) if $\boldsymbol{y} \in NF_j$, then $\boldsymbol{y}$ is not dominated by any solution in $\cup_{i=j:n} NF_i$, and there exists at least one solution in NF_{j-1} that dominates it. Thereafter, the non-dominated front values serve as rankings (NF_1 being the best and NF_n being the worst) that are used for selection. By ensuring that there is sufficient population diversity within each non-dominated front (i.e., different regions of the objective space are covered), NSGA-II is capable of simultaneously converging to a good representation of the entire Pareto set.

An alternative to the above is for MOEAs to evaluate the contribution of a candidate solution in a population based on some quality indicator—such as the hypervolume measure [39]—instead of its dominance level. The advantage of this approach is that for a fixed number of solutions, maximization of the hypervolume not only yields a sub-set of the Pareto front, but also one that is well-distributed [40, 41]—thereby bypassing the need to account for population diversity in any ad-hoc manner. Indeed, indicators that are different from the hypervolume measure have also been successfully used in the past [42].

Aside from dominance- and indicator-based MOEAs, it is contended that the class of decomposition-based methods offer a clear visualization of the multiple searches that take place in MOEAs. It can be shown that every Pareto optimal solution of an MOP is in fact the optimal solution of a scalar-valued optimization problem in which the objective function is some aggregation of all the f_is [43–46]. We think of each scalar optimization problem as a *sub-problem* of the MOP, such that solving a large number of them leads to a good representation of the entire Pareto set. An illustration of the idea is provided in Fig. 1, where each arrow corresponds to a different sub-problem. In the special case of convex Pareto fronts, each sub-problem may be associated with a weight vector $[w_1, w_2, \ldots, w_K]$, leading to the following linear scalarization: $\sum_{i=1:K} w_i \cdot f_i$. Thus, given multiple well-distributed sub-problems (i.e., weight vectors), a decomposition-based MOEA is able to unleash the inherent parallelism of its evolving population to address them simultaneously [43].

Multi-*modal* Evolutionary Optimization

Single-objective optimization algorithms, including many EAs, are usually designed with the primary goal of finding a single optimal solution, despite the possible existence of several (local) optima in the search space—as depicted in Fig. 2. An algorithm is often preferred over others for its ability to escape locally optimal solutions and converge towards the true global optimum. However, there may be practical scenarios in which finding and maintaining a record of multiple optimum solutions is beneficial. For example, imagine a situation in

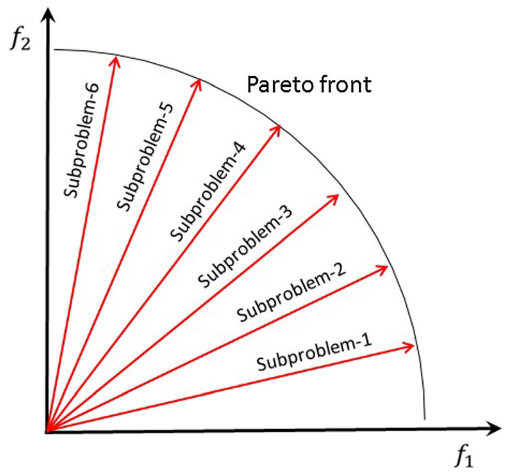

Fig. 1 An illustrative bi-objective convex MOP that has been decomposed into six sub-problems—indicating the mapping of six different search directions into the objective space

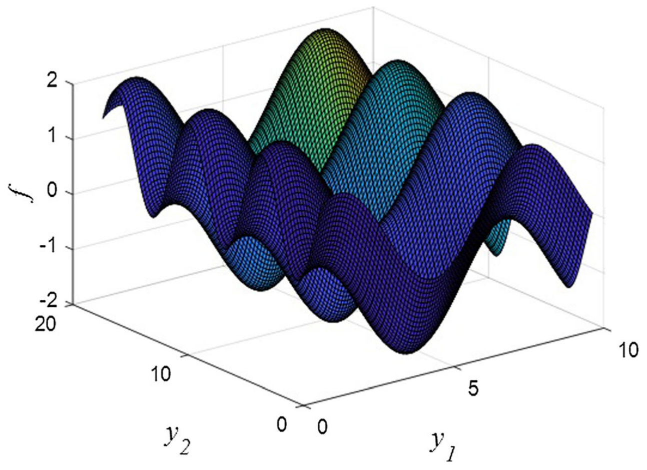

Fig. 2 An illustrative objective function landscape with multiple peaks of interest in multi-modal optimization

which a solution that is considered to be favorable at one point in time (due to the ready availability of necessary resources) suddenly becomes infeasible (due to the unforeseen dearth/exhaustion of some resource). This gives rise to the need for alternative solutions to be deployed within fast timescales that may be too short for a re-optimization process to be carried out. In such situations, maintaining multiple optimum solutions serves as a boon, as it becomes possible for a user to simply switch to a different solution whenever needed [47]. Hereafter, we present a brief overview of evolutionary methods that explicitly incorporate the afore-stated feature characterizing the family of multi-modal optimization problems.

Outline of Methodologies

Even if the objective function of interest can be treated mathematically, for a conventional method to find multiple optimum solutions implies that it must be run several times—with each restart executed with the aim of increasing chances of finding a different locally optimum solution. On the other hand, EAs are naturally at an advantage while dealing with multi-modal optimization problems as the implicit parallelism offered by an evolving population is inherently well suited for simultaneously exploring different regions of the search space. Nevertheless, explicit mechanisms need to be put in place in order to ensure that, over time, the population of an EA does not gradually collapse to only a single optimum. To this end, the majority of existing evolutionary methods use some kind of additional *niching* operation for maintaining a diverse population encompassing multiple optima [13, 48]. For example, the concept of *crowding* achieves the desired outcome by restricting the evolutionary selection (competition) to only apply between individuals belonging to the same neighborhood [49, 50]—drawing inspiration from the analogy that dissimilar individuals residing in different niches tend not to compete with each other. Another scheme that is based on a similar idea is that of *restricted tournament selection* [51]. From a different point of view, there exist methodologies that maintain population diversity by way of penalizing (reducing the fitness value) of individuals in crowded neighborhoods—as realized by the *sharing function* approach [52].

In addition to the above, there are a number of recent attempts at addressing multi-modal optimization problems by leveraging the ability of MOEAs to naturally maintain a diverse population (as discussed in the previous sub-section). Most of these methods apply some kind of transformation that recasts the multi-modal problem into an MOP that can then be handled by suitably modified MOEAs. Some examples of this are found in [53, 54].

Multi-*task* Evolutionary Optimization: an Emerging Paradigm

Multi-task evolutionary optimization, alternatively labeled as *evolutionary multi-tasking* or even *multi-factorial evolution*, puts forth the novel concept of *simultaneously solving multiple self-contained optimization problems/tasks* with the added scope of computationally encoded knowledge transfer between them [14]. If the problems happen to bear some commonality and/or complementarity in terms of their optimal solution(s) and/or function landscapes, then the scope for knowledge transfer often leads to significant performance improvements relative to solving each problem in isolation [55, 56]. One of the key inspirations behind the idea is drawn from the observation that real-world problems seldom exist in isolation. As such, humans possess the innate cognitive ability of recognizing and reusing recurring patterns from their problem-solving experiences to solve related new tasks more efficiently. Along similar lines, artificial intelligence systems of practical relevance (particularly those in industrial settings) are also expected to be faced with multiple related problems over their lifetime—with multi-tasking offering the algorithmic platform for the reuse of knowledge to take place autonomously without the need for any human intervention.

Prior to the proposal in [14], it is observed that despite possessing a population of evolving individuals at its disposal, the design of EAs had primarily been focused on solving only a single task at a time. In contrast, the notion of multi-tasking pushes the envelope of existing EAs, unleashing the power of implicit parallelism in order to simultaneously search for multiple optimal solutions corresponding to multiple distinct optimization tasks at once—with each task serving as a unique *factor* influencing the evolution of the population. In this regard, it is worth mentioning that in order to facilitate the *joint* evolution of multiple tasks, we make the explicit assumption that the search spaces of all tasks can be projected (*encoded*) into some common (*unified*) space—from which task-specific solutions can thereafter be efficiently *decoded*. Thus, the evolutionary processes are carried out in the unified space, mapping solutions back to the original (task-specific) space at the time of evaluation [14].

With the above in mind, and given a series of K distinct tasks $T_1, T_2, \ldots, T_K$ with objective functions $f_1, f_2, \ldots, f_K$, respectively, we can generalize Eq. (2) to get a mathematical formulation of multi-task optimization in terms of the population distributions of constitutive tasks, as follows

$$\underset{\{w_{jk}, p_j(z) \forall j,k\}}{\text{maximize}} \sum_{k=1}^{K} \int_Z f_k(z) \cdot \left[\sum_{j=1}^{K} w_{jk} \cdot p_j(z)\right] \cdot \mathrm{d}z,$$
$$\text{such that}, \sum_{j=1}^{K} w_{jk} = 1, \forall k, \text{and}\, w_{jk} \geq 0, \forall j, k. \tag{7}$$

Here, z indicates a point in the unified space Z, and, to avoid abuse of notation, objective functions $f_1, f_2, \ldots, f_K$ are also considered to be defined in that space. Further, $p_j(z)$ is the population distribution corresponding to the jth task, and the w_{jk}s are scalar weights of a mixture distribution.

It can be seen that optimally solving Eq. (7) will cause a multi-tasking population to simultaneously converge to the global optimums of all K tasks. What is more, during the course of solving Eq. (7), the term $\left[\sum_{j=1}^{K} w_{jk} \cdot p_j(z)\right]$ —indicating a mixture of population distributions—provides a bridge for knowledge to be transferred across them. To elaborate, if candidate solutions for the jth task (encoded in the unified space) turn out to be useful for the kth task as well, then it becomes possible for them to be copied across as a consequence of the mixture term—with the *extent* of transfer being *adapted* by the mixture weight w_{jk}. As a way to reduce the number of variables to be optimized in Eq. (7), we may further consider the weights to be tied by imposing a symmetry condition on them, i.e., $w_{jk} = w_{kj} \forall j, k$. Doing so reflects the practical intuition that if solutions that are good for T_j are useful for T_k as well, then the reverse is also likely to be true. Notably, if any pair of tasks does not complement each other, then setting the mutual mixture weight to zero can mitigate the threat of harmful (*negative*) transfer between them [57–59].

Generalizing Holland's Schema Theorem to Evolutionary Multi-tasking

In Algorithm 2, we provide an abstract workflow for a multi-tasking EA. The model is fundamentally similar to that of a canonical EA (as was presented in Algorithm 1) except for the added scope of knowledge transfer achieved by selecting parent individuals from a distribution over all tasks. Note that the distribution corresponds to the mixture weights in Eq. (7), which must either be carefully prespecified (through manual tuning), or be adaptively learned online based on the data collected for each of the tasks during the course of optimization [20, 57]. In the latter case, there is an additional learning step required that is not included in the algorithm pseudocode.

Algorithm 2 Pseudocode of an abstract multitasking EA applied to K tasks

```
1.  set $t = 0$
2.  *initialize* population $P(t)$ of $N$ individuals in $Z$
3.  Assign each task $N/K$ individuals uniformly at random, forming subpopulations $P_1(t), P_2(t), \ldots, P_K(t)$
4.  *evaluate* individuals in $P_k(t)$ with respect to $f_k$, $\forall k \in \{1,2,\ldots,K\}$
5.  **while termination condition not satisfied do**
6.      set $t = t + 1$
7.      **for** $k = 1{:}K$
8.          $P'_k(t) = \bigcup_{j=1}^{K} [selelct\ {}^{w_{jk} \cdot N}/_{K}\ individuals\ from\ P_j(t-1)]$
9.          apply *genetic operators* on parent population $P'_k(t)$ to get offspring population $P_k(t)$
10.         *evaluate* individuals in $P_k(t)$ with respect to $f_k$
11.     **end for**
12. **end while**
13. return best individual(s) found for each task
```

In the theoretical analyses below, we use the same symbols as in "Holland's Schema Theorem: Implicit Parallelism in Canonical EAs." In cases where we need to clearly distinguish between multi-task and standard single-task optimization, we use superscripts MT and ST, respectively.

Theorem 1 Under fitness proportionate selection, single-point crossover, and no mutation (in Algorithm 2), the expected number of individuals in population $P\left(=\bigcup_{i=1}^{K} P_i\right)$ containing schema H at generation $t+1$ is

$$E[m(H,t+1)] \geq \left[1-p_c\frac{\delta(H)}{l-1}\right]\sum_{k=1}^{K}\sum_{j=1}^{K} w_{jk}\cdot m_j(H,t)\cdot\frac{f_j(H,t)}{\overline{f}_j(t)}, \quad (8)$$

where $m_j(H,t)$ is the number of individuals in $P_j(t)$—i.e., at generation t—containing schema H, $f_j(H,t)$ is the average fitness (with respect to f_j) of individuals containing schema H in $P_j(t)$, and $\overline{f}_j(t)$ is the average fitness of the entire sub-population $P_j(t)$ with respect to f_j.

Proof Let N'_{jk} be the expected number of individuals containing schema H that are selected from $P_j(t)$ while constructing $P'_k(t+1)$ in step 8 of Algorithm 2. Following the mathematical statement of the basic schema theorem, as in Eq. (3), the effect of fitness proportionate selection (prior to any schema disruption due to crossover) implies

$$N'_{jk} = w_{jk}\cdot m_j(H,t)\cdot\frac{f_j(H,t)}{\overline{f}_j(t)}. \quad (9)$$

Thus, the total expected number of individuals containing schema H in $P'_k(t+1)$ is

$$E\left[m'_k(H,t+1)\right] = \sum_{j=1}^{K} N'_{jk} = \sum_{j=1}^{K} w_{jk}\cdot m_j(H,t)\cdot\frac{f_j(H,t)}{\overline{f}_j(t)}. \quad (10)$$

Next, accounting for the probability of crossover and resultant schema disruption / creation in step 9 of Algorithm 2, the total expected number of individuals containing schema H in sub-population $P_k(t+1)$ becomes

$$E[m_k(H,t+1)] \geq \left[1-p_c\frac{\delta(H)}{l-1}\right]\cdot\sum_{j=1}^{K} w_{jk}\cdot m_j(H,t)\cdot\frac{f_j(H,t)}{\overline{f}_j(t)}. \quad (11)$$

Since $E[m(H,t+1)] = \sum_{k=1}^{K} E[m_k(H,t+1)]$, summing Eq. (11) across all K sub-populations leads immediately to the statement of the theorem.

Corollary 1 Under the symmetry condition of the mixture weights, for any schema H (that may represent any region of the unified search space) satisfying the assumption $\delta(H) \ll l$, *if* $f_j(H,t) \geq \overline{f}_j(t) \forall j$ and $\exists k$ *s.t.* $f_k(H,t) > \overline{f}_k(t)$, then the frequency of H is guaranteed to grow in the population of the multi-tasking EA.

Proof Note that $m(H,t) = \sum_{j=1}^{K} m_j(H,t)$. Further, using the commutativity property of double finite sums, we can rewrite Eq. (8) as below. Herein, the assumption $\delta(H) \ll l$ allows crossover-based schema disruption / creation to be ignored.

$$E[m(H,t+1)] \approx \sum_{j=1}^{K} m_j(H,t)\cdot\frac{f_j(H,t)}{\overline{f}_j(t)}\cdot\sum_{k=1}^{K} w_{jk}. \quad (12)$$

Given $w_{jk} = w_{kj}\ \forall j,\ k$, we have $\sum_{j=1}^{K} w_{jk} = 1 \forall k \Longrightarrow \sum_{k=1}^{K} w_{jk} = 1 \forall j$. Thus, the right hand side of Eq. (12) reduces to $\sum_{j=1}^{K}\left[m_j(H,t)\cdot\frac{f_j(H,t)}{\overline{f}_j(t)}\right]$. Since $\frac{f_j(H,t)}{\overline{f}_j(t)} \geq 1 \forall j$ with at least one strict inequality, it follows that

$$\sum_{j=1}^{K}\left[m_j(H,t)\cdot\frac{f_j(H,t)}{\overline{f}_j(t)}\right] > m(H,t) \Longrightarrow gr^{MT}(H,t) = \frac{E[m(H,t+1)]}{m(H,t)} > 1. \quad (13)$$

In other words, the frequency of any such schema continues to grow in the population, highlighting the parallelism of EAs even in the case of simultaneously progressing multiple distinct searches.

Corollary 2 Within any sub-population P_k solving task T_k, and for any favorable (short) schema H satisfying $\delta(H) \ll l$ and $f_k(H,t) > \overline{f}_k(t)$, with an optimal combination of mixture weights we can guarantee $gr_k^{MT}(H,t) \geq gr_k^{ST}(H,t)$. Further, for any pair of short schemata H_a and H_b, if $f_k(H_a,t) > f_k(H_b,t)$, then $\exists \boldsymbol{w}_k = [w_{1k}, w_{2k}, \ldots, w_{Kk}]$ such that $gr_k^{MT}(H_a,t) > gr_k^{MT}(H_b,t)$.

Proof The possibility of crossover-based schema disruption/creation is ignored under $\delta(H) \ll l$. Then, setting $w_{jk} = 0\ \forall j \neq k$ in Eq. (11) (which induces zero knowledge transfer across tasks), and comparing with Eq. (5), we have

$$\left[\frac{E[m_k(H,t+1)]}{m_k(H,t)}\right]_{w_{jk}=0 \forall j \neq k} \approx \frac{f_k(H,t)}{\overline{f}_k(t)} \approx gr_k^{ST}(H,t). \quad (14)$$

However, clearly $\max_{w_{jk},\forall j}\frac{E[m_k(H,t+1)]}{m_k(H,t)} \geq \frac{f_k(H,t)}{\overline{f}_k(t)}$, implying that for an *optimal* configuration of mixture weights, it is guaranteed that $gr_k^{MT}(H,t) = \max_{w_{jk},\forall j}\frac{E[m_k(H,t+1)]}{m_k(H,t)} \geq gr_k^{ST}(H,t)$.

Similarly, note that $\left[\frac{E[m_k(H_a,t+1)]}{m_k(H_a,t)}\right]_{w_{jk}=0\forall j\neq k} - \left[\frac{E[m_k(H_b,t+1)]}{m_k(H_b,t)}\right]_{w_{jk}=0\forall j\neq k} \approx \frac{f_k(H_a,t)}{\bar{f}_k(t)} - \frac{f_k(H_b,t)}{\bar{f}_k(t)} > 0$. Therefore, $\max_{w_{jk},\forall j} \frac{E[m_k(H_a,t+1)]}{m_k(H_a,t)} - \frac{E[m_k(H_b,t+1)]}{m_k(H_b,t)} > 0$, which trivially ensures $gr_k^{MT}(H_a,t) > gr_k^{MT}(H_b,t)$ for some $\mathbf{w}_k$.

The first part of Corollary 2 sheds light on the potential for a multi-tasking EA to utilize knowledge transferred from other tasks in the multi-tasking environment to accelerate convergence towards high quality schema (in comparison to the conventional single-task case). However, for this outcome to be achieved consistently, the mixture weights must either be carefully tuned, or optimally learned online during the course of the multi-tasking search. As a counter to Corollary 2, it can also be shown that under a poor configuration of the mixture weights, the search towards favorable schema may in fact slow down compared to standard single-tasking, thereby highlighting the need to account for the threat of negative transfers in multi-task optimization.

Algorithmic Advances in Evolutionary Multi-tasking

A general schematic of evolutionary multi-tasking is depicted in Fig. 3. The knowledge base shown therein may either comprise of raw population datasets pertaining to each of the tasks (as in Algorithm 2) or learned models that capture recurring patterns from the data and can subsequently be transferred across tasks. For instance, probabilistic models that actualize the population distribution terms in Eq. (7) may be contained in the knowledge base.

Although the notion of evolutionary multi-tasking has been proposed relatively recently, there are a reasonable number of algorithmic advances that have already been reported. Nevertheless, we acknowledge that there is still much scope for future research in this topic. Among existing methods, perhaps the most common form of communication between tasks is through so-called *implicit genetic transfer* [14, 60, 61]. In this setting, the knowledge base (as shown in Fig. 3) consists of population datasets associated with each task in the multi-tasking EA, similar to Algorithm 2. However, differently from Algorithm 2 (where solutions are transferred through explicit sampling), knowledge exchange in implicit genetic transfer occurs when individuals belonging to different tasks crossover (mix/recombine) with one another. After crossover, the genetic material from one task can get copied into an offspring individual associated with a different task. The basic principle behind the approach is that if the transferred genetic material is beneficial, then the offspring survives in the population; else, if the transfer is harmful, then the corresponding offspring is gradually ejected by the selection mechanism.

As an alternative to the direct transfer of genetic material, the algorithms developed in [62, 63] propose to explicitly learn a mapping (*M*) between solution datasets of task pairs

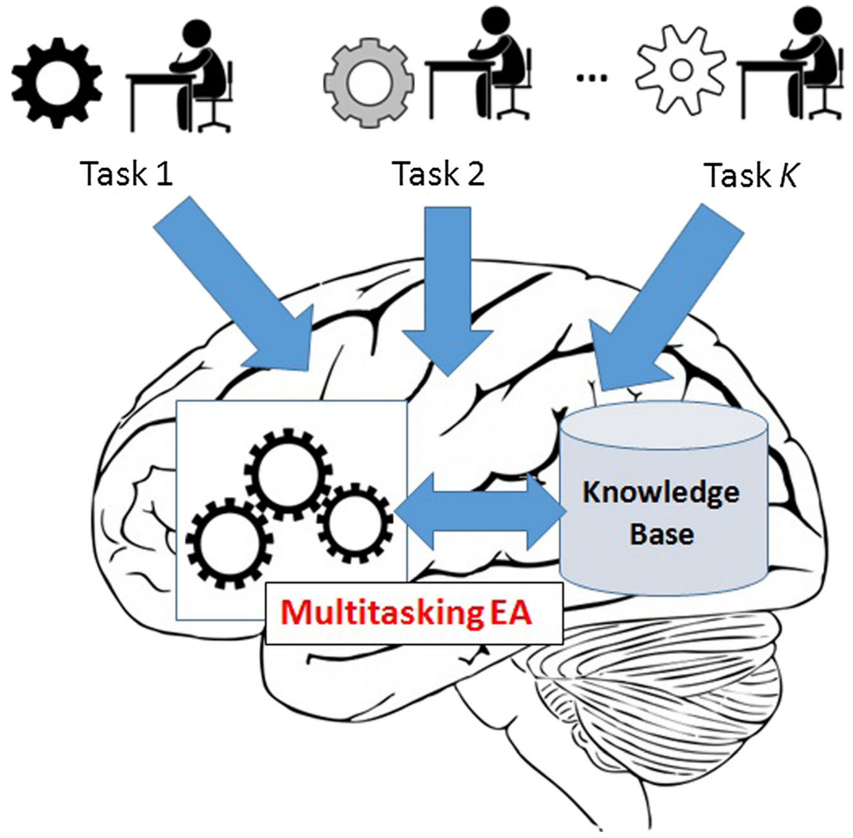

Fig. 3 A simplified illustration of the cognitively inspired evolutionary multi-tasking paradigm catering to multiple optimization tasks with the scope of computationally encoded knowledge transfer between them

so as to induce *high ordinal correlation* between their respective objective functions. Specifically, given any task pair T_j, T_k, the mapping is learned with the goal of satisfying: $f_j(z_1) < f_j(z_2) \Rightarrow f_k(M(z_1)) < f_k(M(z_2))$. The rationale behind incorporating such a mapping into a multi-tasking EA can be seen through the following simple result.

Proposition 1 Assuming the learned mapping M is a bijection (one-to-one and onto) that satisfies condition $f_j(z_1) < f_j(z_2) \Rightarrow f_k(M(z_1)) < f_k(M(z_2))$ for task pair T_j, T_k, then $[z^* = \text{argmax}\, f_j(z)] \Longrightarrow [M(z^*) = \text{argmax}\, f_k(z)]$.

Thus, instead of directly copying genetic material from one task to another, it is first transformed using the learned mapping—as a way of reducing the threat of negative transfers in those cases where the objective functions of distinct tasks exhibit negative ordinal correlation. It is worth highlighting that the additional cost of learning a mapping has the potential advantage of uncovering latent relationships between tasks, which might not be immediately achievable via simple genetic transfer.

Finally, we note that a task in a multi-tasking environment need not be restricted to having a scalar-valued objective function. Indeed, given a series of K tasks, some (or even all) of them may be MOPs. Algorithms capable of effective multi-tasking in such situations, employing either an implicit or an explicit mode of knowledge transfer, have been reported in [64–66].

Other Recent Additions to the Multi-*X* Family

In this section, we cover problem formulations that, at least within the field of evolutionary computation, have only recently begun to draw research attention. In much of what follows, the theories and algorithms presented in "Multi-*task* Evolutionary Optimization: an Emerging Paradigm," catering to the case of simultaneously progressing multiple searches in multi-task evolutionary optimization, will be found to have widespread utility.

Multi-*level* Evolutionary Optimization

Multi-level programming is a generalization of mathematical programming in which the constraint region is implicitly determined by a series of optimization problems which must be solved in a predetermined sequence [67]. For ease of exposition, in this paper, we limit our discussion to the special case where there are two hierarchical levels, which are generally referred to simply as *bi-level optimizations* [15, 16]. The formulation of a standard bi-level problem comprises of a *lower level* optimization task (also known as the *follower's problem*) that is nested within an *upper level* optimization task (also known as the *leader's* problem). The two levels jointly contribute a pair of objective functions, with the interaction between them captured according to Eq. (15) below—where we suppress the possible existence of additional constraint functions for simplicity.

$$\begin{aligned} &maximize_{y_u \in Y_u} f_u(y_u, y_l^*), \\ &\text{such } that, y_l^* = \text{argmax}_{y_l \in Y_l} f_l(y_u, y_l). \end{aligned} \tag{15}$$

In the above, f_u is the objective function of the leader, and f_l is the objective function of the follower. The problem statement specifies that the leader makes the first choice by selecting a candidate solution $\boldsymbol{y}_u$ from the upper level search space Y_u. Thereafter, the follower selects $\boldsymbol{y}_l^*$ from the lower level search space Y_l, such that the selected solution optimizes the lower level objective function f_l conditioned on the leader's preceding choice. In other words, the lower level task in Eq. (15) is parametrized by $\boldsymbol{y}_u$, and a solution $\boldsymbol{y}_l^*$ is considered feasible for the upper level problem only if it is optimal for the lower level.

An illustration of bi-level optimization is provided in Fig. 4. Generally speaking, the main purpose behind solving this problem is to ascertain the optimum solution/decision for the leader, given the hard constraint of optimally solving the follower's problem. It is worth mentioning that a variety of real-world scenarios of interest can be modeled as above, with some prominent examples occurring in the toll setting of highways [68], economic game theory (particularly the Stackelberg duopoly) [69], agricultural and environmental policy making [70], homeland security [71], cybersecurity [72], engineering design [73], etc.

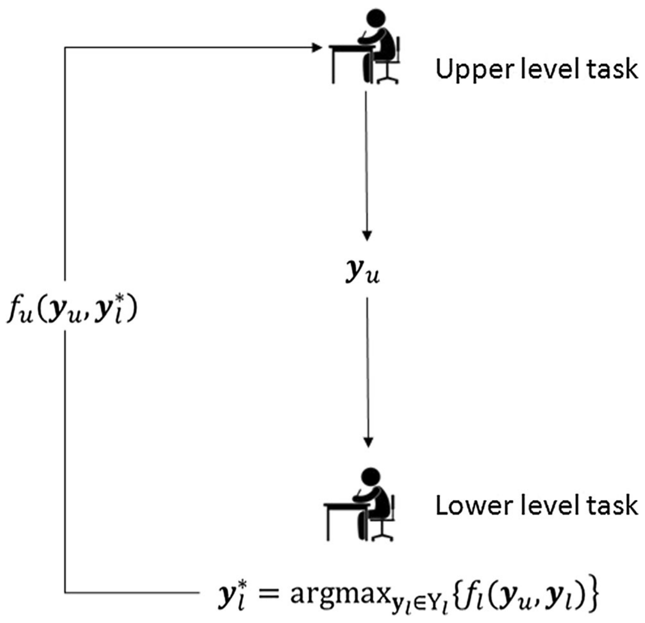

Fig. 4 In bi-level optimization, a solution $\boldsymbol{y}_l^*$, given $\boldsymbol{y}_u$, is considered feasible for the upper level problem only if it is optimal for the corresponding lower level problem

Outline of Methodologies

It has been shown that the hierarchical structure of bi-level optimization problems leads to difficulties, such as non-convexity and disconnectedness, that are difficult to handle using purely mathematical procedures [15], thereby making evolutionary methods a natural choice due to their general purpose global optimization capability. In this regard, a common approach is to use *nested algorithms*, wherein the lower level problem is solved to optimality for every candidate upper level solution [74–76]. To elaborate, if we assume that the upper level problem is tackled via an EA, then, for evaluating the fitness of any individual at that level, the corresponding lower level optimal response must be deduced—possibly using an EA as well.

While naïve nested algorithms employing EAs at both levels can often be effective, they are generally not very scalable. This is because with increasing dimensionality of the upper level search space Y_u, the number of lower level optimization tasks to be solved grows exponentially. What is more, if the lower level tasks are difficult, then solving them one by one becomes highly computationally intensive. As an alternative, in [77], the authors proposed to tackle multiple lower level optimization tasks simultaneously by extending the idea of evolutionary multi-tasking. The implicit parallelism of a population facilitates the joint evolution of multiple lower level tasks, with the continuous transfer of computationally encoded knowledge between them contributing towards significantly faster convergence (with more than 50% speedup reported in [77] relative to the naïve nested algorithm). In particular, assuming we have a total of K lower level tasks corresponding to a set of upper level candidate solutions $\{\boldsymbol{y}_{u,1}, \boldsymbol{y}_{u,2}, \ldots, \boldsymbol{y}_{u,K}\}$, the multi-task problem formulation can be expressed as

$$\underset{\{w_{jk}, p_j(y_l) \forall j,k\}}{\text{maximize}} \sum_{k=1}^{K} \int_{Y_l} f_l(y_{u,k}, y_l) \cdot \left[\sum_{j=1}^{K} w_{jk} \cdot p_j(y_l)\right] \cdot dy_l,$$
$$\text{such } that, \sum_{j=1}^{K} w_{jk} = 1, \forall k, \text{and } w_{jk} \geq 0, \forall j,k. \tag{16}$$

Equation (16) closely resembles Eq. (7), except that in the bi-level case: (a) we know that the K tasks are likely to be related as they are all associated with the same objective function f_l—implying that the threat of negative transfers is minimal—and (b) there is no separate unified search space as all lower level tasks are already defined in a common space Y_l.

Different from the above, there have been other attempts to speed up bi-level evolutionary optimization by training models to directly approximate lower level responses, instead of conducting full optimization runs [78, 79]. However, we do not examine these methods further herein as they place the onus more on the learning and exploitation of accurate approximation models [80], instead of harnessing the salient features of evolutionary search.

Finally, we note that the objective function at the lower (and even the upper) level of a bi-level optimization problem need not be scalar-valued. In such cases, corresponding to every upper level candidate solution $\boldsymbol{y}_u$, a lower level MOP must be solved to obtain a *set* of Pareto optimal trade-off solutions. Various perspectives have recently been put forward on how the multiple solutions returned by the lower level are to be handled at the upper level. This includes a so-called *optimistic* formulation in [81, 82], as well as a *pessimistic/adversarial/worst-case* formulation in [83, 84]. In either case, the overall multi-objective bi-level optimization problem is shown to offer an ideal setting for the implicit parallelism of EAs to be harnessed.

Multi-*hard* and Multi-*disciplinary* Evolutionary Optimization

Herein, we merge our discussions on multi-hard and multi-disciplinary evolutionary optimization as both these paradigms share a similar motivation of *solving problems comprised of a non-trivial combination of multiple complex sub-problems*, with the sub-problems/components interacting with each other in a manner such that solving them independently may either be infeasible or lead to subpar solutions to the overall problem [85]. In order to distinguish from multi-task optimization, we highlight that the components of a multi-hard or multi-disciplinary problem are closely interacting, whereas the constituents of multi-task optimization are distinct and self-contained (which may or may not have known commonalities).

The term "multi-hard" has recently been coined for practical optimization problems with multiple interacting *single-hard* sub-problems, commonly occurring in the operations research domain. Here, the term single-hard refers to a designed combinatorial optimization task of high computation complexity (that is NP-hard) [17]. As an illustrative example of the routine occurrence of multi-hard problems in the real world, one may consider the need to address intricate interactions across various echelons and silos while optimizing global supply chains [86]. As a specific case study, it has been shown that optimizing the supply chain operations of a mining company involves simultaneously dealing with mine planning and scheduling, stockpile management and blending, train scheduling, port operations, etc., with the overall objective of satisfying customer demands [17].

Similar to the above, the term "multi-disciplinary optimization"—commonly used in the engineering design community—refers to the accurate modeling of interactions between sub-systems that form part of a larger engineering system [87]. For example, consider the fact that the functionality of a generic turbofan engine that is used by most commercial airliners depends on a host of interdependent sub-systems, such as compressors, turbine, and combustion chamber [88]. An abstract illustration of the coupling for a simplified case comprising only two sub-systems is depicted in Fig. 5. Therein, $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$ represent the variable vectors corresponding to sub-system 1 and sub-system 2, respectively, while $\boldsymbol{y}_c$ stands for a common variable vector shared by both sub-systems. Further, $\boldsymbol{g}_{12}$ and $\boldsymbol{g}_{21}$ are referred to as coupling variables, which represent outputs of one sub-system that are required as input in the other sub-system. Note that in the mathematical formulation of an arbitrary K-sub-system multi-disciplinary optimization problem (as provided below), we ignore the coupling variables since they can frequently be subsumed into the common variable vector $\boldsymbol{y}_c$. The resultant *quasi-separable* multi-disciplinary optimization problem takes the form of the following MOP [89]

$$\text{maximize}_{\{\boldsymbol{y}_1,\boldsymbol{y}_2,\ldots,\boldsymbol{y}_K,\boldsymbol{y}_c\}} f_{\text{system}} = [f_1(\boldsymbol{y}_1,\boldsymbol{y}_c), f_2(\boldsymbol{y}_2,\boldsymbol{y}_c), \ldots, f_K(\boldsymbol{y}_K,\boldsymbol{y}_c)]. \quad (17)$$

Alternatively, a scalar-valued variant of Eq. (17) can be written as [19]

$$\text{maximize}_{\{\boldsymbol{y}_1,\boldsymbol{y}_2,\ldots,\boldsymbol{y}_K,\boldsymbol{y}_c\}} f_{\text{system}} = \sum_{i=1}^{K} f_i(\boldsymbol{y}_i,\boldsymbol{y}_c). \quad (18)$$

The discussions and formulations above demonstrate that large-scale multi-hard and multi-disciplinary optimizations give rise to settings wherein multiple searches corresponding to multiple interacting sub-problems/components/sub-systems are to be carried out simultaneously in order to achieve a satisfactory solution to the overall problem. Further, the interactions between components often lead to complexities that may be difficult to handle using exact mathematical procedures. Thus, following the general premise of the paper, we find that problems of this kind fall into a category (namely, multi-X) that lends well to the unique characteristics of EAs.

Outline of Methodologies

EAs of the *cooperative co-evolutionary* type are among the most promising for multi-hard and multi-disciplinary optimization [90, 91]. In particular, cooperative co-evolution solves the different sub-problems using separate co-evolving *sub-populations*, where each sub-population optimizes the variables associated with a particular component. In the process of doing so, the interaction between a particular sub-problem and the others is established through a collaboration step occurring in evaluation. Precisely, while evaluating an individual in a sub-population, we first select representative *collaborators* from the other sub-populations (such as the individuals with the best fitness values), which are subsequently *concatenated* to form a complete solution to the overall problem [92].

In addition to cooperative co-evolution, methods for multi-level optimization have also been applied to multi-disciplinary problems by imposing a hierarchical relationship on their components. For example, in [89], the authors proposed a nested decomposition of Eq. (17) into two levels, as follows

$$\begin{aligned} &maximize_{y_c \in Y_c} f_{system}(y_1^*, y_2^*, \ldots, y_K^*, y_c), \\ \text{such that}, &\; y_1^* = \text{argmax}_{y_1 \in Y_1} f_1(y_1, y_c), y_2^* = \text{argmax}_{y_2 \in Y_2} f_2(y_2, y_c), \ldots, \\ &y_K^* = \text{argmax}_{y_K \in Y_K} f_K(y_K, y_c). \end{aligned} \quad (19)$$

The leader's problem, which in this case is also referred to as the supervisor level, is to optimize the common variable vector $\boldsymbol{y}_c$. Given a particular choice of the common variable vector, the lower level then consists of K sub-systems that are treated as K distinct self-contained optimization tasks T_1, T_2, …, T_K. Notably, if the sub-systems share some commonality, then the entire lower level may ideally be tackled via

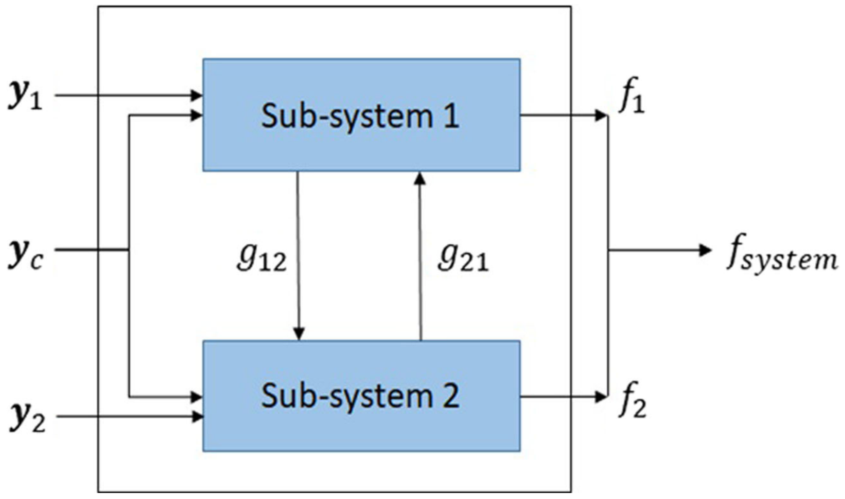

Fig. 5 A simplified two-disciplinary system

evolutionary multi-tasking (see "Multi-*task* Evolutionary Optimization: an Emerging Paradigm").

Similar to the above, a bi-level perspective on multi-hard combinatorial optimization was recently outlined in [93]. In particular, the approach proposed to decompose the so-called *profitable tour problem*—which combines the classical knapsack problem at the strategic level with a vehicle routing problem at the operational level—into its constituent parts by imposing a hierarchical relationship between them. In this example, the knapsack problem served as the upper level, the routing problem served as the corresponding lower level, and an efficient knowledge transfer-enhanced EA was designed for solving them simultaneously.

Multi-*form* Evolutionary Optimization

An illustration of the key idea behind multi-form optimization is presented in Fig. 6. It can be seen that at its core, the multi-form optimization paradigm utilizes a multi-tasking EA. However, while the notion of multi-tasking focuses on multiple self-contained tasks, multi-form optimization distinguishes itself in the sense that it proposes to reconcile multiple alternate formulations of a single target task of interest [20].

In many real-world scenarios, it may not be clear how best to formulate the optimization problem at hand. Alternate formulations induce different search behaviors, some of which may be more effective than others for a particular problem instance. For example, recent work in generating a low-dimensional formulation (via *random* embedding) of an otherwise high-dimensional optimization task can be found in [94]. In this case, it is practically impossible to know beforehand whether a particular random embedding is relevant or not, leading to the need to consider multiple (sequential) random embeddings [95]. Further examples appear in constrained optimization, where the best way to incorporate a penalty function, which converts a constrained problem into an unconstrained one, is not obvious [96]. Likewise, while multi-objectivizing standard single-objective problems has been proposed in the literature [33, 97], it is not necessary that the MOP formulation is always easier to solve [98]. In all such cases, assigning the various possible formulations to be simultaneously tackled in a multi-tasking EA relieves the user of the pains of ascertaining the most appropriate one. What is more, each formulation may serve as a related *helper* task [99, 100], enabling their unique advantages to be harnessed through continuous knowledge transfer.

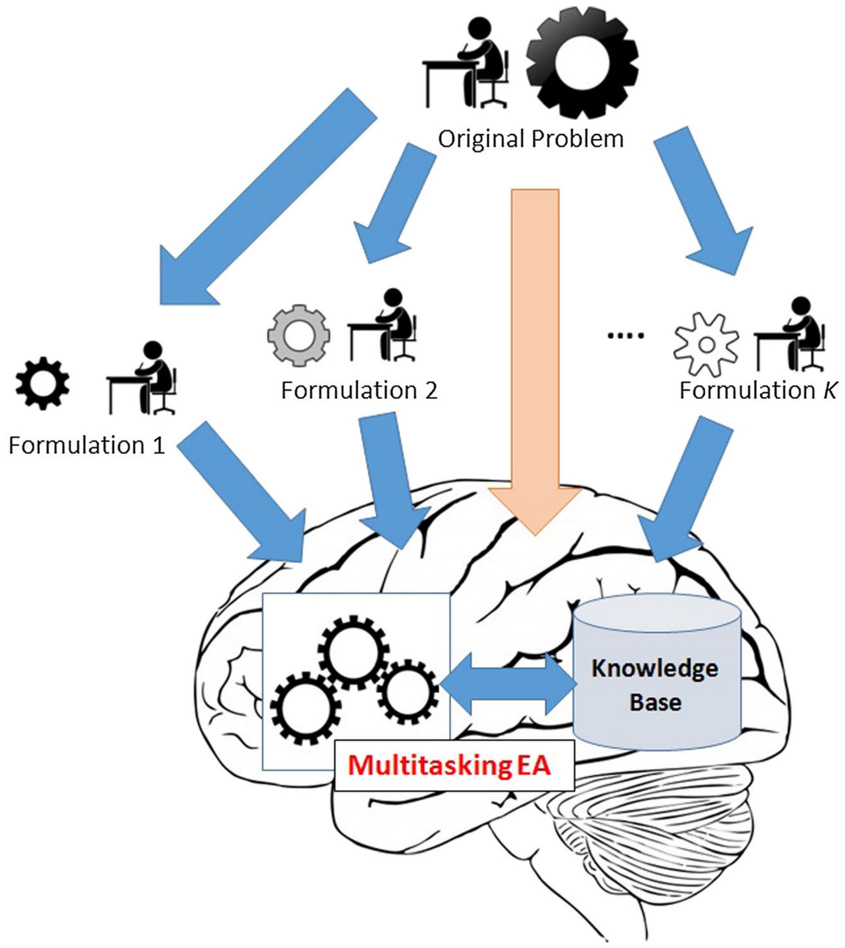

Fig. 6 Multi-form optimization is a novel paradigm that attempts to reconcile alternate formulations of a single optimization task through a multi-tasking approach

In addition to the examples above, where the best formulation of an optimization problem is hard to determine a priori, there are cases where although certain formulations are known to be less precise, they may be computationally much cheaper to handle. Such *multi-fidelity problems* provide another setting for multi-tasking EAs to transfer knowledge from low- to high-fidelity formulations with the goal of efficiently solving the target expensive task [61].

Outline of Methodologies

As of now, there is a lack of comprehensive realizations of the multi-form evolutionary optimization paradigm. One early example can however be found in [21], where single- and multi-objectivized formulations of the traveling salesman problem were treated jointly via an implicit genetic transfer-based multi-tasking EA. The results showed that the outcome of the multi-form optimization approach was either competitive with or significantly better than the most effective formulation considered in isolation.

Conclusions

In this paper, we have attempted to return to the roots of evolutionary computation—drawing the attention of upcoming researchers to the fundamentals of EAs, and population-based search in general, that set them apart from other optimization procedures in the literature. We identified a series of optimization problem settings that are deemed to be uniquely suited for exploiting the implicit parallelism inherent to EAs. We capture associated research activities under the umbrella term of multi-*X* evolutionary computation, where *X*, as of now, represents the following list: {"objective," "modal," "task," "level," "hard," "disciplinary," "form"}. The main purpose of this position paper is to place the spotlight on those areas of optimization problem-solving that are well aligned with the fundamentals of evolutionary computation. In turn, we hope to catalyze further research efforts in these directions, encouraging the steady expansion of the list *X* with new applications in the near future.

It is noted that the subject of multi-objective and multi-modal optimization have long been strongholds of evolutionary computation research. In contrast, however, multi-task, multi-level, multi-hard, multi-disciplinary, and multi-form optimization are newly emerging concepts (at least within the evolutionary computation research community) that offer tremendous scope for future research. For instance, we have only recently begun to unveil the potential of evolutionary multi-tasking, with much left to be done for the paradigm to consistently guarantee performance superiority over existing techniques. Likewise, while preliminary success has indeed been achieved in the application of EAs to multi-hard and multi-disciplinary optimization problems, major challenges remain in scaling the methods to complex real-world settings involving increasing number of interacting components.

A recurring facet of the problems we have examined herein is that they involve multiple searches to be carried out at the same time, such as finding multiple solutions to a single optimization task (as is the case in multi-objective and multi-modal optimization) and jointly solving distinct tasks (in multi-task optimization). In all cases, the seemingly cognitive ability of EAs to simultaneously sample, evaluate, and process a vast number of regions of the search space, while evolving a relatively small number of candidate individuals is deemed to offer distinctive advantages for problems of the multi-*X* type. The theoretical underpinnings behind our claims are also set forth in the paper by generalizing Holland's schema theorem (the fundamental theorem of EAs) to the case of distinct searches occurring in multi-tasking.

Funding This research is funded by the National Research Foundation Singapore under its AI Singapore Programme (Award No.: AISG-RP-2018-004).

Compliance with Ethical Standards

Conflict of Interest The authors declare that they have no conflict of interest.

Ethical Approval This article does not contain any studies with human participants or animals performed by any of the authors.

References

1. Siddique N, Adeli H. Nature-inspired chemical reaction optimisation algorithms. Cogn Comput. 2017;9(4):411–22.
2. Kendall GI. Evolutionary computation evolving fast enough? IEEE Comput Intell Mag. 2018;13(2):42–51.
3. Charles D. On the origin of species by means of natural selection. London: Murray; 1859.
4. Dantzig GB. Discrete-variable extremum problems. Oper Res. 1957;5(2):266–88.
5. Croes GA. A method for solving traveling-salesman problems. Oper Res. 1958;6(6):791–812.
6. Ong YS, Lim MH, Zhu N, Wong KW. Classification of adaptive memetic algorithms: a comparative study. IEEE Trans Syst Man Cybern B Cybern. 2006;36(1):141–52.
7. Holland JH. Genetic algorithms. Sci Am. 1992;267(1):66–73.
8. Bertoni A, Dorigo M. Implicit parallelism in genetic algorithms. Artif Intell. 1993;61(2):307–14.
9. Zhang Q, Muhlenbein H. On the convergence of a class of estimation of distribution algorithms. IEEE Trans Evol Comput. 2004;8(2):127–36.
10. Wierstra D, Schaul T, Glasmachers T, Sun Y, Peters J, Schmidhuber J. Natural evolution strategies. J Mach Learn Res. 2014;15(1):949–80.

11. Trivedi A, Srinivasan D, Sanyal K, Ghosh A. A survey of multiobjective evolutionary algorithms based on decomposition. IEEE Trans Evol Comput. 2017;21(3):440–62.
12. Coello CC. Evolutionary multi-objective optimization: a historical view of the field. IEEE Comput Intell Mag. 2006 Feb;1(1):28–36.
13. Das S, Maity S, Qu BY, Suganthan PN. Real-parameter evolutionary multimodal optimization—a survey of the state-of-the-art. Swarm Evol Comput. 2011;1(2):71–88.
14. Gupta A, Ong YS, Feng L. Multifactorial evolution: toward evolutionary multitasking. IEEE Trans Evol Comput. 2016;20(3):343–57.
15. Sinha A, Malo P, Deb K. A review on bilevel optimization: from classical to evolutionary approaches and applications. IEEE Trans Evol Comput 2018;22(2):276–295.
16. Sinha A, Malo P, Deb K. Evolutionary bilevel optimization: an introduction and recent advances. In: Recent advances in evolutionary multi-objective optimization. Cham: Springer; 2017. p. 71–103.
17. Przybylek MR, Wierzbicki A, Michalewicz Z. Decomposition algorithms for a multi-hard problem. Evol Comput. 2017;20(Early Access):1–27.
18. Friedrich T, Neumann F. What's hot in evolutionary computation. In AAAI 2017 (pp. 5064–5066).
19. Lu X, Menzel S, Tang K, Yao X. Cooperative co-evolution-based design optimization: a concurrent engineering perspective. IEEE Trans Evol Comput. 2018 Apr;22(2):173–88.
20. Gupta A, Ong YS, Feng L. Insights on transfer optimization: because experience is the best teacher. IEEE Trans Emerg Topics Comput Intell. 2018;2(1):51–64.
21. Da B, Gupta A, Ong YS, Feng L. Evolutionary multitasking across single and multi-objective formulations for improved problem solving. In Evolutionary Computation (CEC), 2016 IEEE Congress on 2016 Jul 24 (pp. 1695-1701). IEEE.
22. Tanweer MR, Sundaram S. Human cognition inspired particle swarm optimization algorithm. In Intelligent sensors, sensor networks and information processing (ISSNIP), 2014 IEEE Ninth International Conference on 2014 Apr 21 (pp. 1-6). IEEE.
23. Molina D, LaTorre A, Herrera F. An insight into bio-inspired and evolutionary algorithms for global optimization: review, analysis, and lessons learnt over a decade of competitions. Cogn Comput. 2018:1–28.
24. Ghanem WA, Jantan AA. Cognitively inspired hybridization of artificial bee colony and dragonfly algorithms for training multilayer perceptrons. Cogn Comput. 2018:1–39.
25. Ong YS, Gupta A. Evolutionary multitasking: a computer science view of cognitive multitasking. Cogn Comput. 2016;8(2):125–42.
26. Bridges CL, Goldberg DE. An analysis of reproduction and crossover in a binary-coded genetic algorithm. In: Proceedings of the Second International Conference on genetic algorithms and their application. Hillsdale: L. Erlbaum Associates Inc.; 1987. pp. 9–13.
27. Goldberg DE, Sastry K. A practical schema theorem for genetic algorithm design and tuning. In Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation 2001 Jul 7 (pp. 328–335). Morgan Kaufmann Publishers Inc.
28. Grefenstette JJ. Conditions for implicit parallelism. In Foundations of genetic algorithms 1991 Jan 1 (Vol. 1, pp. 252-261). Elsevier.
29. Gupta A, Heng CK, Ong YS, Tan PS, Zhang AN. A generic framework for multi-criteria decision support in eco-friendly urban logistics systems. Expert Syst Appl. 2017;71:288–300.
30. Ishibuchi H, Yoshida T, Murata T. Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling. IEEE Trans Evol Comput. 2003;7(2):204–23.
31. Jin Y, Sendhoff B. Pareto-based multiobjective machine learning: an overview and case studies. IEEE Trans Syst Man Cybern C Appl Rev. 2008;38(3):397–415.
32. Beniakar ME, Sarigiannidis AG, Kakosimos PE, Kladas AG. Multiobjective evolutionary optimization of a surface mounted PM actuator with fractional slot winding for aerospace applications. IEEE Trans Magn. 2014;50(2):665–8.
33. Ceberio J, Calvo B, Mendiburu A, Lozano JA. Multi-objectivising combinatorial optimisation problems by means of elementary landscape decompositions. Evol Comput. 2018;15(Early Access):1–21.
34. Kalyanmoy D. Multi objective optimization using evolutionary algorithms: Wiley; 2001.
35. Ehrgott M. Multicriteria optimization: Springer Science & Business Media; 2005.
36. Srinivas N, Deb K. Muiltiobjective optimization using nondominated sorting in genetic algorithms. Evol Comput. 1994;2(3):221–48.
37. Luo J, Gupta A, Ong YS, Wang Z. Evolutionary optimization of expensive multiobjective problems with co-sub-pareto front Gaussian process surrogates. IEEE Trans Cybern. 2018:1–14.
38. Deb K, Pratap A, Agarwal S, Meyarivan TA. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans Evol Comput. 2002;6(2):182–97.
39. Jiang S, Zhang J, Ong YS, Zhang AN, Tan PS. A simple and fast hypervolume indicator-based multiobjective evolutionary algorithm. IEEE Trans Cybern. 2015;45(10):2202–13.
40. Beume N, Naujoks B, Emmerich M. SMS-EMOA: multiobjective selection based on dominated hypervolume. Eur J Oper Res. 2007;181(3):1653–69.
41. Emmerich M, Beume N, Naujoks B. An EMO algorithm using the hypervolume measure as selection criterion. In International Conference on Evolutionary Multi-Criterion Optimization 2005 Mar 9 (pp. 62–76). Springer, Berlin, Heidelberg.
42. Tian Y, Cheng R, Zhang X, Cheng F, Jin Y. An indicator based multi-objective evolutionary algorithm with reference point adaptation for better versatility. IEEE Trans Evol Comput. 2017.
43. Zhang Q, Li H. MOEA/D: a multiobjective evolutionary algorithm based on decomposition. IEEE Trans Evol Comput. 2007;11(6):712–31.
44. Wu M, Li K, Kwong S, Zhang Q, Zhang J. Learning to decompose: a paradigm for decomposition-based multiobjective optimization. IEEE Trans Evol Comput. 2018.
45. He Z, Yen GG. Diversity improvement in decomposition-based multi-objective evolutionary algorithm for many-objective optimization problems. In Systems, man and cybernetics (SMC), 2014. IEEE International Conference on 2014 Oct 5 (pp. 2409-2414). IEEE.
46. Ishibuchi H, Setoguchi Y, Masuda H, Nojima Y. Performance of decomposition-based many-objective algorithms strongly depends on Pareto front shapes. IEEE Trans Evol Comput. 2017;21(2):169–90.
47. Deb K, Saha A. Finding multiple solutions for multimodal optimization problems using a multi-objective evolutionary approach. In Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation 2010 Jul 7 (pp. 447–454). ACM.
48. Li X, Epitropakis MG, Deb K, Engelbrecht A. Seeking multiple solutions: an updated survey on niching methods and their applications. IEEE Trans Evol Comput. 2017;21(4):518–38.
49. Mahfoud SW. Simple analytical models of genetic algorithms for multimodal function optimization. In ICGA 1993 Feb (p. 643).
50. Mengshoel OJ, Goldberg DE. Probabilistic crowding: deterministic crowding with probabilistic replacement. In Proc of the Genetic and Evolutionary Computation Conference (GECCO-99) 1999 (p. 409).

51. Harik GR. Finding multimodal solutions using restricted tournament selection. In: ICGA; 1995. p. 24–31.
52. Goldberg DE, Richardson J. Genetic algorithms with sharing for multimodal function optimization. In Genetic algorithms and their applications: Proceedings of the Second International Conference on Genetic Algorithms 1987 (pp. 41–49). Hillsdale: Lawrence Erlbaum.
53. Basak A, Das S, Tan KC. Multimodal optimization using a biobjective differential evolution algorithm enhanced with mean distance-based selection. IEEE Trans Evol Comput. 2013;17(5): 666–85.
54. Wang Y, Li HX, Yen GG, Song W. MOMMOP: multiobjective optimization for locating multiple optimal solutions of multimodal optimization problems. IEEE Trans Cybern. 2015;45(4):830–43.
55. Zhou L, Feng L, Zhong J, Zhu Z, Da B, Wu Z. A study of similarity measure between tasks for multifactorial evolutionary algorithm. In Proceedings of the Genetic and Evolutionary Computation Conference Companion 2018 Jul 6 (pp. 229-230). ACM.
56. Gupta A, Ong YS, Da B, Feng L, Handoko SD. Landscape synergy in evolutionary multitasking. In Evolutionary computation (CEC), 2016 IEEE Congress on 2016 Jul 24 (pp. 3076-3083). IEEE.
57. Da B, Gupta A, Ong YS. Curbing negative influences online for seamless transfer evolutionary optimization. IEEE Trans Cybern. 2018:1–14.
58. Wen YW, Ting CK. Parting ways and reallocating resources in evolutionary multitasking. In Evolutionary computation (CEC), 2017 IEEE Congress on 2017 Jun 5 (pp. 2404-2411). IEEE.
59. Lu J, Behbood V, Hao P, Zuo H, Xue S, Zhang G. Transfer learning using computational intelligence: a survey. Knowl-Based Syst. 2015;80:14–23.
60. Tang J, Chen Y, Deng Z, Xiang Y, Joy CPA. Group-based approach to improve multifactorial evolutionary algorithm. In: IJCAI; 2018. p. 3870–6.
61. Ding J, Yang C, Jin Y, Chai T. Generalized multi-tasking for evolutionary optimization of expensive problems. IEEE Trans Evol Comput. 2017.
62. Feng L, Zhou L, Zhong J, Gupta A, Ong YS, Tan KC, et al. Evolutionary multitasking via explicit autoencoding. IEEE Trans Cybern. 2018:1–14.
63. Bali KK, Gupta A, Feng L, Ong YS, Siew TP. Linearized domain adaptation in evolutionary multitasking. In Evolutionary computation (CEC), 2017 IEEE Congress on 2017 Jun 5 (pp. 1295-1302). IEEE.
64. Gupta A, Ong YS, Feng L, Tan KC. Multiobjective multifactorial optimization in evolutionary multitasking. IEEE Trans Cybern. 2017;47(7):1652–65.
65. Yang C, Ding J, Jin Y, Wang C, Chai T. Multitasking multiobjective evolutionary operational indices optimization of beneficiation processes. IEEE Trans Autom Sci Eng. 2018:1–12.
66. Mo J, Fan Z, Li W, Fang Y, You Y, Cai X. Multi-factorial evolutionary algorithm based on M2M decomposition. In: Asia-Pacific Conference on Simulated Evolution and Learning. Cham: Springer; 2017. p. 134–44.
67. Bard JF, Falk JE. An explicit solution to the multi-level programming problem. Comput Oper Res. 1982;9(1):77–100.
68. Wang JY, Ehrgott M, Dirks KN, Gupta A. A bilevel multi-objective road pricing model for economic, environmental and health sustainability. Transp Res Procedia 2014;3:393–402.
69. Von Stackelberg H. The theory of the market economy: Oxford University Press; 1952.
70. Whittaker G, Färe R, Grosskopf S, Barnhart B, Bostian M, Mueller-Warrant G, et al. Spatial targeting of agri-environmental policy using bilevel evolutionary optimization. Omega. 2017;66: 15–27.
71. Brown G, Carlyle M, Diehl D, Kline J, Wood K. A two-sided optimization for theater ballistic missile defense. Oper Res. 2005;53(5):745–63.
72. Eisenstadt E, Moshaiov A. Novel solution approach for multi-objective attack-defense cyber games with unknown utilities of the opponent. IEEE Trans Emerg Topics Comput Intell. 2017;1(1):16–26.
73. Gupta A, Kelly PA, Ehrgott M, Bickerton S. A surrogate model based evolutionary game-theoretic approach for optimizing non-isothermal compression RTM processes. Compos Sci Technol. 2013;84:92–100.
74. Sinha A, Malo P, Frantsev A, Deb K. Finding optimal strategies in a multi-period multi-leader–follower Stackelberg game using an evolutionary algorithm. Comput Oper Res. 2014;41:374–85.
75. Angelo JS, Krempser E, Barbosa HJ. Differential evolution for bilevel programming. In Evolutionary computation (CEC), 2013. IEEE Congress on 2013 Jun 20 (pp. 470-477). IEEE.
76. Angelo JS, Barbosa HJ. A study on the use of heuristics to solve a bilevel programming problem. Int Trans Oper Res. 2015;22(5): 861–82.
77. Gupta A, Mańdziuk J, Ong YS. Evolutionary multitasking in bi-level optimization. Compl Intell Syst. 2015;1(1–4):83–95.
78. Sinha A, Malo P, Deb K. An improved bilevel evolutionary algorithm based on quadratic approximations. In Evolutionary computation (CEC), 2014. IEEE Congress on 2014 Jul 6 (pp. 1870-1877). IEEE.
79. Sinha A, Malo P, Deb K. Evolutionary algorithm for bilevel optimization using approximations of the lower level optimal solution mapping. Eur J Oper Res. 2017 Mar 1;257(2):395–411.
80. Islam MM, Singh HK, Ray T. A surrogate assisted approach for single-objective bilevel optimization. IEEE Trans Evol Comput. 2017;21(5):681–96.
81. Deb K, Sinha A. An efficient and accurate solution methodology for bilevel multi-objective programming problems using a hybrid evolutionary-local-search algorithm. Evol Comput. 2010;18(3): 403–49.
82. Gupta A, Ong YS. An evolutionary algorithm with adaptive scalarization for multiobjective bilevel programs. In Evolutionary computation (CEC), 2015. IEEE Congress on 2015 May 25 (pp. 1636-1642). IEEE.
83. Eisenstadt E, Moshaiov A, Avigad G. Co-evolution of strategies for multi-objective games under postponed objective preferences. In Computational intelligence and games (CIG), 2015. IEEE Conference on 2015 Aug 31 (pp. 461-468). IEEE.
84. Żychowski A, Gupta A, Mańdziuk J, Ong YS. Addressing expensive multi-objective games with postponed preference articulation via memetic co-evolution. Knowl-Based Syst. 2018;154:17–31.
85. Bonyadi MR, Michalewicz Z, Neumann F, Wagner M. Evolutionary computation for multicomponent problems: opportunities and future directions. arXiv preprint arXiv:1606.06818. 2016 Jun 22.
86. Ibrahimov M. Evolutionary algorithms for supply chain optimisation (doctoral dissertation). The University of Adelaide 2012.
87. Cramer EJ, Dennis JE Jr, Frank PD, Lewis RM, Shubin GR. Problem formulation for multidisciplinary optimization. SIAM J Optim. 1994;4(4):754–76.
88. Min AT, Sagarna R, Gupta A, Ong YS, Goh CK. Knowledge transfer through machine learning in aircraft design. IEEE Comput Intell Mag. 2017;12(4):48–60.
89. Rabeau S, Dépincé P, Bennis F. Collaborative optimization of complex systems: a multidisciplinary approach. Int J Interact Des Manuf. 2007;1(4):209–18.
90. Omidvar MN, Li X, Mei Y, Yao X. Cooperative co-evolution with differential grouping for large scale optimization. IEEE Trans Evol Comput. 2014;18(3):378–93.

91. Zhao W, Alam S, Abbass HA. MOCCA-II: a multi-objective co-operative co-evolutionary algorithm. Appl Soft Comput. 2014;23: 407–16.
92. Mei Y, Li X, Yao X. On investigation of interdependence between sub-problems of the travelling thief problem. Soft Comput. 2016;20(1):157–72.
93. Handoko SD, Chuin LH, Gupta A, Soon OY, Kim HC, Siew TP. Solving multi-vehicle profitable tour problem via knowledge adoption in evolutionary bi-level programming. In Evolutionary computation (CEC), 2015 IEEE Congress on 2015 May 25 (pp. 2713-2720). IEEE.
94. Wang Z, Hutter F, Zoghi M, Matheson D, de Feitas N. Bayesian optimization in a billion dimensions via random embeddings. J Artif Intell Res. 2016;55:361–87.
95. Qian H, Hu YQ, Yu Y. Derivative-free optimization of high-dimensional non-convex functions by sequential random embeddings. In IJCAI 2016 Jul 9 (pp. 1946–1952).
96. Coello CA. Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. Comput Methods Appl Mech Eng. 2002;191(11–12): 1245–87.
97. Abbass HA, Deb K. Searching under multi-evolutionary pressures. In: International Conference on Evolutionary Multi-Criterion Optimization. Berlin: Springer; 2003. p. 391–404.
98. Handl J, Lovell SC, Knowles J. Multiobjectivization by decomposition of scalar cost functions. In: International Conference on Parallel Problem Solving from Nature. Berlin: Springer; 2008. p. 31–40.
99. Caruana R. A dozen tricks with multitask learning. In Neural networks: tricks of the trade. Berlin: Springer; 1998. p. 165–91.
100. Chandra R, Gupta A, Ong YS, Goh CK. Evolutionary multi-task learning for modular knowledge representation in neural networks. Neural Process Lett. 2018;47(3):993–1009.