

# *Evolutionary multitasking in bi-level optimization*

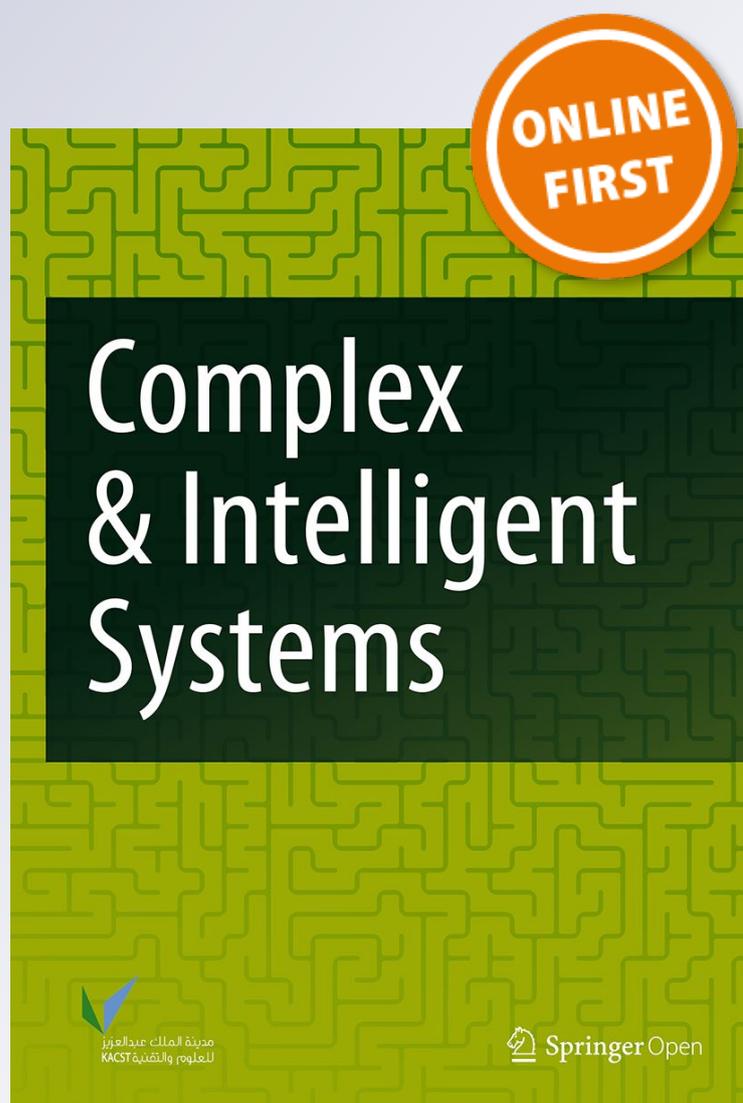
**Abhishek Gupta, Jacek Mańdziuk & Yew-Soon Ong**

**Complex & Intelligent Systems**

ISSN 2199-4536

Complex Intell. Syst.

DOI 10.1007/s40747-016-0011-y



**Your article is published under the Creative Commons Attribution license which allows users to read, copy, distribute and make derivative works, as long as the author of the original work is cited. You may self-archive this article on your own website, an institutional repository or funder's repository and make it publicly available immediately.**

# Evolutionary multitasking in bi-level optimization

Abhishek Gupta<sup>1</sup> · Jacek Mańdziuk<sup>2,3</sup> · Yew-Soon Ong<sup>1</sup>

Received: 1 December 2015 / Accepted: 11 February 2016  
© The Author(s) 2016. This article is published with open access at Springerlink.com

**Abstract** Evolutionary multitasking has recently emerged as an effective means of facilitating implicit genetic transfer across different optimization tasks, thereby potentially accelerating convergence characteristics for multiple tasks at once. A natural application of the paradigm is found to arise in the area of bi-level programming wherein an upper level optimization problem must take into consideration a nested lower level problem. Thus, while tackling instances of bi-level optimization, a significant challenge surfaces from the fact that multiple upper level candidate solutions are to be analyzed at the same time by inferring the corresponding optimum response from the lower level. Thus, the process of bi-level optimization often becomes exorbitantly time consuming, especially in the case of real-world instances involving expensive objective function evaluations. Accordingly, the significance of this paper lies in showcasing that the practicality of population-based bi-level optimization can be considerably enhanced by simply incorporating the novel concept of evolutionary multitasking into the search process. As a result, it becomes possible to process multiple lower level optimization tasks concurrently, thereby facilitating the exploitation of underlying commonalities among them. To demonstrate the implications of our proposal, we present computational experiments on some synthetic benchmark

functions, as well as a real-world case study in complex engineering design from the composites manufacturing industry.

**Keywords** Evolutionary multitasking · Bi-level optimization · Evolutionary algorithms · Complex engineering design

## 1 Introduction

A recent study has shown that the concept of multitasking in optimization provides the scope for potentially fruitful implicit knowledge transfer across different optimization tasks, thereby facilitating accelerated convergence characteristics among them [19]. A natural beneficiary of this novel paradigm is contended to be the field of evolutionary bi-level optimization that has begun to gain much research attention in recent years. The formulation of a single-objective bi-level optimization problem (BLOP) is special in the sense that one optimization task (often referred to as the lower level problem or the follower's problem) is nested within another (which is labeled as the upper level problem or the leader's problem). The two levels of this single-act hierarchical optimization setting together comprise a pair of objective functions, namely,  $F : \mathbb{R}^u \times \mathbb{R}^l \rightarrow \mathbb{R}$  and  $f : \mathbb{R}^u \times \mathbb{R}^l \rightarrow \mathbb{R}$  [7, 26, 28, 29]. The relationship between the two functions can be stated according to Eqs. (1) and (2) below. Note that the likely occurrence of additional equality and/or inequality constraints has been suppressed in the formulation for simplicity.

$$\text{Minimize}_{\mathbf{x}_u \in X_u, \mathbf{x}_l \in X_l} F(\mathbf{x}_u, \mathbf{x}_l), \quad (1)$$

$$\text{subject to, } \mathbf{x}_l \in \text{argmin}\{f(\mathbf{x}_u, \mathbf{x}_l)\}. \quad (2)$$

In the above,  $F$  is the objective function of the so-called leader, and  $f$  represents the objective function of the fol-

✉ Yew-Soon Ong  
asysong@ntu.edu.sg

<sup>1</sup> Rolls-Royce@NTU Corporate Lab c/o, School of Computer Engineering, Nanyang Technological University, Singapore, Singapore

<sup>2</sup> School of Computer Engineering, Nanyang Technological University, Singapore, Singapore

<sup>3</sup> Faculty of Mathematics and Information Science, Warsaw University of Technology, Warsaw, Poland

lower. By being single-act, it is implied that each decision maker (i.e., the leader and the follower) must make a one-time decision. The leader makes the (chronologically) first choice by selecting a specific design vector (or decision vector)  $\mathbf{x}_u$  from the upper level design space  $X_u \subset \mathbb{R}^u$ . Thereafter, the follower simply selects a preferred  $\mathbf{x}_l$  from the lower level design space  $X_l \subset \mathbb{R}^n$ , such that its objective is optimized given the leader's preceding action. In other words, a rational follower is expected to arrive at a preferred strategy by solving a lower level optimization task (see Eq. 2) that is parametrized by the leader's prior decision  $\mathbf{x}_u$ .

A fundamental assumption of the BLOP is that the leader possesses complete knowledge about the characteristics of the design space available to the follower and is aware of the fact that the follower can observe and appropriately respond to the leader's prior action. Accordingly, the main purpose of a BLOP is to determine the optimum decision for the leader in such a setting. Clearly, a mathematical treatment of this scenario must embed the optimization problem of the follower (as a lower level task) within that of the leader (which forms the upper level problem), as has been indicated by Eqs. (1) and (2). It is interesting to note that a variety of scenarios in operations research, logistics, transportation research, science, economics, security games, complex engineering design, etc., can in fact be posed (or naturally occur) in the form of BLOPs [1, 2, 17, 18, 20, 21, 30, 31, 34].

Not surprisingly, due to its apparent real-world relevance, the field of BLOPs has attracted much research attention over several decades [4, 10, 35]. More recently, the interest has spread to researchers in the field of computational intelligence for developing more widely applicable techniques for tackling complex real-world BLOPs, i.e., via the use of evolutionary algorithms (EAs) [27, 29, 31]. As is well known, EAs provide several advantages over conventional mathematical tools due to their considerable flexibility in dealing with a wide variety of optimization problems [3]. However, EAs are not exempt from their own share of algorithmic difficulties. With regard to BLOPs, a significant challenge surfaces from the fact that multiple upper level candidate solutions are to be analyzed at the same time by inferring the corresponding optimum response from the lower level. To elaborate, distinct lower level optimization tasks emerge with respect to each upper level candidate solution. Failure to accurately resolve each lower level task may often cause a misleading representation of the leader's estimated objective function value corresponding to a given upper level design vector  $\mathbf{x}_u$ . As a result, the process of optimization generally becomes exorbitantly time consuming, especially when dealing with real-world problems involving several expensive objective function evaluations and conflicts between the two levels.

With the above in mind, in this paper, we demonstrate that the practicality of population-based bi-level optimization can

be significantly enhanced simply by incorporating the novel concept of *evolutionary multitasking* into the search process. As has been demonstrated in [19, 22], evolutionary multitasking provides the scope for accelerating convergence to near optimal solutions of multiple optimization tasks at once, simply by harnessing the latent complementarities among them (when available). In particular, the true power of implicit parallelism of population-based search is unleashed by combining the design spaces corresponding to different tasks into a *unified pool of genetic material*, thereby facilitating implicit information exchange among them in the form of encoded genetic material. While encapsulating the major implication of this phenomenon, it is most interesting to note that the notion of multitasking emerges naturally in the realm of evolutionary bi-level optimization wherein multiple lower level optimization tasks (one corresponding to every upper level population member) are to be solved at once. Accordingly, the significance of the present paper lies in revealing the efficacy of evolutionary multitasking as a novel means to augment bi-level optimization. Traditional evolutionary search processes are shown to be considerably enriched by appropriately exploiting the unique benefits of multitasking in optimization.

To provide an in-depth exposition of the notions presented heretofore, the remainder of this paper has been structured as follows. Section 2 introduces the basic concepts of multitasking in optimization and presents an associated multitasking EA. Thereafter, the incorporation of multitasking into BLOPs is described in detail in Sect. 3. Computational tests on some synthetic benchmark problems shall then be carried out in Sect. 4, followed by a real-world case study from the composites manufacturing industry in Sect. 5. Finally, a summary of the research, conclusions, and future directions shall be presented in Sect. 6.

## 2 An overview of evolutionary multitasking

We have conceived of the evolutionary multitasking paradigm as a means to leverage upon the true power of implicit parallelism of population-based search. To this end, consider a hypothetical scenario in which  $K$  different optimization tasks are to be solved at the same time. Without loss of generality, all tasks are considered to be minimization problems. The  $k$ th task, denoted  $T_k$ , has a design space  $X_k$  on which the objective function is defined as  $f_k: X_k \rightarrow \mathbb{R}$ . Then, the aim of an evolutionary multitasking engine is to efficiently navigate the design space of all tasks concurrently, aided by the scope for implicit genetic exchange, so as to rapidly obtain  $\text{argmin}\{f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_K(\mathbf{x})\}$ . Since each  $f_k$  is treated as an additional *factor* influencing the evolution of a single population of individuals, the composite problem is also referred to as a  $K$ -factorial environment [19].

While developing an EA for the purpose of multitasking, it is necessary to first devise a technique for comparing candidate solutions in a multitasking environment. To this end, we define a set of properties for every individual  $p_j$ , where  $j \in \{1, 2, \dots, |P|\}$ , in a population  $P$ . Note that every individual is encoded into a unified search space  $Y$  encompassing  $X_1, X_2, \dots, X_K$ , and can thereafter be decoded into a task-specific solution representation with respect to each of the  $K$  optimization tasks.

**Definition 1** (*Factorial rank*) The factorial rank  $r_k^j$  of  $p_j$  for task  $T_k$  is the index of  $p_j$  in the list of population members sorted in ascending order of  $f_k$ .

**Definition 2** (*Skill factor*) The skill factor  $\tau_j$  of  $p_j$  is the one task, amongst all other tasks in a  $K$ -factorial environment, with which the individual is associated. If  $p_j$  is evaluated for all tasks then  $\tau_j = \operatorname{argmin}_k \{r_k^j\}$ , where  $k \in \{1, 2, \dots, K\}$ .

**Definition 3** (*Scalar fitness*) The scalar fitness of  $p_j$  in a multitasking environment is given by  $\varphi_j = 1/r_{\tau_j}^j$ .

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, individual  $p_1$  will be considered to dominate individual  $p_2$  during evolutionary multitasking simply if  $\varphi_1 > \varphi_2$ .

It should be noted that the procedure described heretofore for comparing individuals is not absolute. As the factorial rank of an individual (and implicitly its scalar fitness) 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,  $\varphi^* \geq \varphi_j$  for all  $j \in \{1, 2, \dots, |P|\}$ . Therefore, it can be said that the proposed technique is indeed consistent with the ensuing definition of optimality in evolutionary multitasking.

**Definition 4** (*Optimality in evolutionary multitasking*) An individual  $p^*$ , with a list of objective values  $\{f_1^*, f_2^*, \dots, f_K^*\}$ , is considered to be optimum during evolutionary multitasking iff  $\exists k \in \{1, 2, \dots, K\}$  such that  $f_k^* \leq f_k(x_k)$ , for all feasible  $x_k \in X_k$ .

## 2.1 The multifactorial evolutionary algorithm: an evolutionary multitasking engine

In [19], the *Multifactorial Evolutionary Algorithm* (MFEA) was developed as a computational analog of the bio-cultural models of *multifactorial inheritance* [8,9,25]. The unique feature of the MFEA is that in a multitasking environment it effectively combines the transmission of biological as well as cultural building blocks from parents to their offspring. Thus, the algorithm is ascribed to the field of memetic computation [7,23] which has recently emerged as a successful

computational paradigm inspired by Darwinian principles of natural selection as well as Dawkins' notion of a *meme* as the basic unit of cultural transmission [11].

---

### Algorithm 1: Pseudocode of the MFEA

---

1. Randomly generate  $n$  individuals in  $Y$  to form initial population  $P_0$
  2. **for every**  $p_j$  in  $P_0$  **do**  
     Assign skill factor  $\tau_j = \operatorname{mod}(j, K)+1$ , for the case of  $K$  tasks  
     Evaluate  $p_j$  for task  $\tau_j$  only
  3. **end for**
  4. Compute scalar fitness  $\varphi_j$  for every  $p_j$
  5. Set  $t = 0$
  6. **while** (stopping conditions are not satisfied) **do**  
      $C_t = \text{Crossover+Mutate}(P_t)$   
     **for every**  $c_j$  in  $C_t$  **do**  
         Determine skill factor  $\tau_j \rightarrow$  Refer Algorithm 2  
         Evaluate  $c_j$  for task  $\tau_j$  only  
     **end for**  
      $R_t = C_t \cup P_t$   
     Update scalar fitness of all individuals in  $R_t$ .  
     Select  $N$  fittest members from  $R_t$  to form  $P_{t+1}$ .  
     Set  $t = t + 1$
  7. **end while**
- 

As summarized in Algorithm 1, the MFEA is initiated by randomly generating a population of  $n$  individuals in a unified search space  $Y$ . Moreover, each individual in the initial population is pre-assigned a specific skill factor (see Definition 2) in a manner that guarantees every task to have uniform number of representatives. We would like to state that the skill factor of an individual is viewed as the computational representation of its pre-assigned cultural trait. The significance of this step is to ensure that an individual is only evaluated with respect to a single task (i.e., only its skill factor) among all other tasks in the multitasking environment. Doing so is considered practical since evaluating every individual exhaustively for every task will generally be computationally demanding, especially when  $K$  (the number of tasks in the multitasking environment) becomes large. The remainder of the algorithm proceeds similarly to standard evolutionary procedures, with the only major deviation occurring in terms of offspring evaluation, as is described next.

### 2.1.1 Offspring evaluation in the MFEA

Following the memetic phenomenon of *vertical cultural transmission* [6], offspring in the MFEA experience strong cultural influences from their parents, in addition to inheriting their genetic material. In the bio-cultural models of multifactorial inheritance, vertical cultural transmission is viewed as a mode of inheritance that operates in tandem with genetics and leads to the phenotype of an offspring being directly influenced by the phenotype of its parents.

**Algorithm 2:** Vertical cultural transmission via selective imitation

Consider offspring  $c \in C_i$  where  $c = \text{Crossover} + \text{Mutate}(p_1, p_2)$

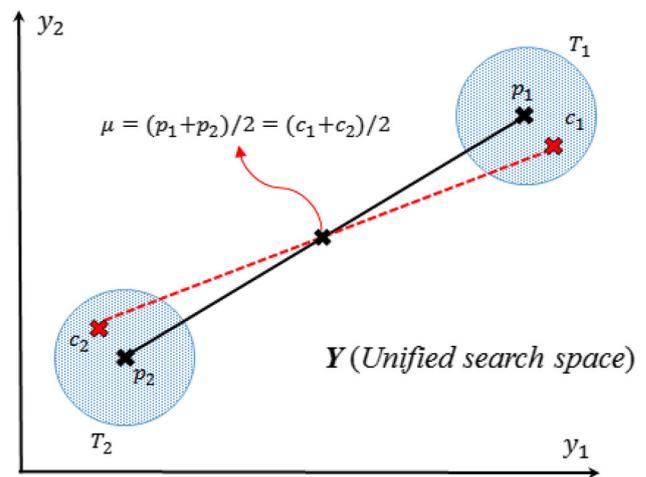
1. Generate a random number *rand* between 0 and 1.
2. **if**  $\text{rand} \leq 0.5$  **then**  
 $c$  imitates skill factor of  $p_1$
3. **else**  
 $c$  imitates skill factor of  $p_2$
4. **end if**

The algorithmic realization of the aforementioned notion is achieved in the MFEA via a *selective imitation strategy*. In particular, selective imitation is used to mimic the commonly observed phenomenon that offspring tend to imitate the cultural traits (i.e., skill factors) of their parents. Accordingly, in the MFEA, an offspring is only decoded (from the unified genotype space  $Y$  to a task-specific phenotype space) and evaluated with respect to a single task with which at least one of its parents is associated. As has been mentioned earlier, selective evaluation plays a role in managing the computation expense of the MFEA. A summary of the steps involved is provided in Algorithm 2.

### 2.1.2 Implicit genetic transfer in the unified search space

*Information*, in the MFEA, exists in the form of encoded genetic material in the unified search space  $Y$ . Thus, the transfer of relevant information from one task to another occurs in the form of *implicit genetic transfer* when parents belonging to different cultural backgrounds (i.e., having different skill factors) undergo crossover. With this in mind, consider the situation in Fig. 1 where two parents  $p_1$  and  $p_2$ , with different skill factors, undergo crossover in a hypothetical 2-D search space. To be precise,  $p_1$  has skill factor  $T_1$  and  $p_2$  has skill factor  $T_2$ . A pair of offspring,  $c_1$  and  $c_2$ , is created by the simulated binary crossover (SBX) operator [12]. A noteworthy feature of SBX is that it creates offspring that are located close to the parents with high probability [14]. As a result,  $c_1$  is found to inherit much of its genetic material from  $p_1$ , while  $c_2$  is found to inherit much of its genetic material from  $p_2$ . In such a scenario, if  $c_1$  imitates the skill factor of  $p_2$  (i.e., if  $c_1$  is evaluated for  $T_2$ ) and/or if  $c_2$  imitates the skill factor of  $p_1$  (i.e.,  $c_2$  is evaluated for  $T_1$ ), then implicit genetic transfer occurs between the two tasks. In particular, if the genetic material corresponding to  $T_1$  (carried by  $c_1$ ) happens to also be useful for  $T_2$ , or vice versa, then the transfer is fruitful (or positive). Otherwise, there is a high chance for the transfer to be negative [15, 24]. However, the nice property of evolution is that when the latter occurs, the negatively transferred genes get gradually eliminated from the population by the process of natural selection.

Note that for implicit genetic transfer to take place efficiently, it is crucial to have an effective unified search space encompassing all the constitutive tasks. Accordingly, assum-



**Fig. 1** Parent candidates  $p_1$  and  $p_2$  undergo standard SBX crossover to produce offspring  $c_1$  and  $c_2$  that are located close to their parents with high probability. Parent  $p_1$  possesses skill factor  $T_1$  while  $p_2$  possesses skill factor  $T_2$ . If  $c_1$  is evaluated for  $T_2$  and/or if  $c_2$  is evaluated for  $T_1$ , then implicit genetic transfer is said to occur between the two tasks

ing the design space dimensionality of task  $T_k$  to be  $D_k$ , we define a unified search space  $Y$  such that  $D_{\text{multitask}} = \max_k \{D_k\}$  for  $k \in \{1, 2, \dots, K\}$ . Thus, the chromosome of an individual in the MFEA is defined by a vector of  $D_{\text{multitask}}$  random-keys [5], with each random-key bounded within the fixed range  $[0, 1]$ . While addressing task  $T_k$ , we simply refer to  $D_k$  random-keys contained in the chromosome.

With the above in mind, we now describe the decoding procedure of a chromosome  $y \in Y$  into a meaningful task-specific solution representation. In the case of continuous optimization, this can be achieved in a straightforward manner by linearly mapping each random-key from the genotype space to the design space of the optimization task. For instance, consider a task  $T_k$  in which the  $i$ th variable ( $x_i$ ) is bounded in the range  $[L_i, U_i]$ . If the  $i$ th random-key of the chromosome  $y$  takes a value  $y_i \in [0, 1]$ , then the decoding procedure is simply given by,

$$x_i = L_i + (U_i - L_i) \cdot y_i. \quad (3)$$

### 3 Multitasking in bi-level optimization

In this section, we describe the means by which the multitasking paradigm can be incorporated into the evolutionary search process to enhance a bi-level optimization procedure. In particular, we shall incorporate the MFEA into a basic Nested Bi-Level Evolutionary Algorithm (N-BLEA). The main feature of the N-BLEA, as is imposed by the nature of the BLOP, is that a lower level optimization task must be appropriately solved with respect to each population member in the upper level EA. In particular, with respect to the  $j$ th

upper level candidate solution  $\mathbf{x}_{u,j}$ , the corresponding lower level objective function takes the form  $f(\mathbf{x}_{u,j}, \mathbf{x}_l)$ . The setup of the solution procedure is illustrated in Fig. 2. It is to be noted that the optimization of the lower level task is also undertaken by an equivalent EA, so as to overcome the challenges associated with multi-modality, non-differentiability of the lower level objective function, etc. For a summary of the N-BLEA, the reader is referred to Algorithm 3.

It must be kept in mind that a *standard* EA is used at both levels of the N-BLEA, with no non-traditional algorithmic artifacts as have been incorporated in some other previously proposed methodologies for evolutionary bi-level optimization (such as the use of biased lower level population initialization in [31] or the use of a quadratic approximation model in [28]). This prevents any adulteration of the observed results by factors that are not of interest to the present study. Thereby allowing us to clearly identify and focus on the performance improvements achieved by simply introducing the concept of multitasking into bi-level optimization.

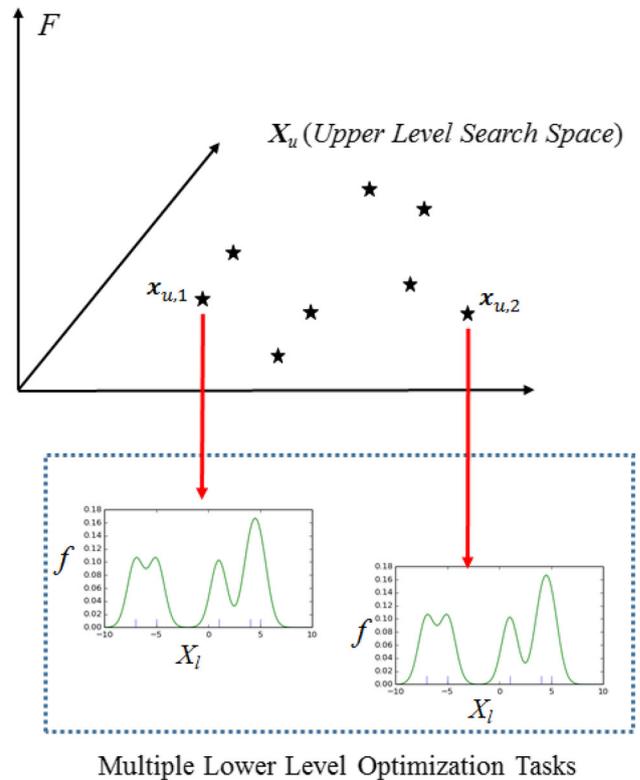
---

**Algorithm 3:** The Nested-BLEA (N-BLEA)

---

1. Randomly generate  $N$  individuals to form initial population  $P_{u,0}$
  2. **for every**  $p_j$  in  $P_{u,0}$  **do**  
     Get  $\mathbf{x}_{l,j}^*$  by evolutionary optimization of lower level task  
     Evaluate  $p_j$  as  $F(\mathbf{x}_{u,j}, \mathbf{x}_{l,j}^*)$
  3. **end for**
  4. Set  $T = 0$
  5. **while** (stopping conditions are not satisfied) **do**  
      $C_{u,T} = \text{Crossover+Mutate}(P_{u,T})$   
     **for every**  $c_j$  in  $C_{u,T}$  **do**  
         Get  $\mathbf{x}_{l,j}^*$  by evolutionary optimization of lower level task  
         Evaluate  $c_j$  as  $F(\mathbf{x}_{u,j}, \mathbf{x}_{l,j}^*)$   
     **End**  
      $R_{u,T} = C_{u,T} \cup P_{u,T}$   
     Select  $N$  fittest members from  $R_{u,T}$  to form  $P_{u,T+1}$   
     Set  $T = T + 1$
  6. **end while**
- 

As highlighted by Fig. 2, multiple lower level optimization tasks are to be solved during every generation of the upper level EA. Thus, it can clearly be seen that the scope for evolutionary multitasking naturally emerges within the context of BLOPs. However, to increase the probability of fruitful information exchange during the process of multitasking, some important steps are taken into consideration. To elaborate, we begin with the assumption that the landscape of the lower level objective function  $f(\mathbf{x}_u, \mathbf{x}_l)$  changes slowly with respect to small changes in  $\mathbf{x}_u$ ; this is contended to be a reasonable assumption in many cases of practical interest. Accordingly, it can be claimed that lower level optimization tasks corresponding to neighboring upper level population members are likely to possess useful underly-

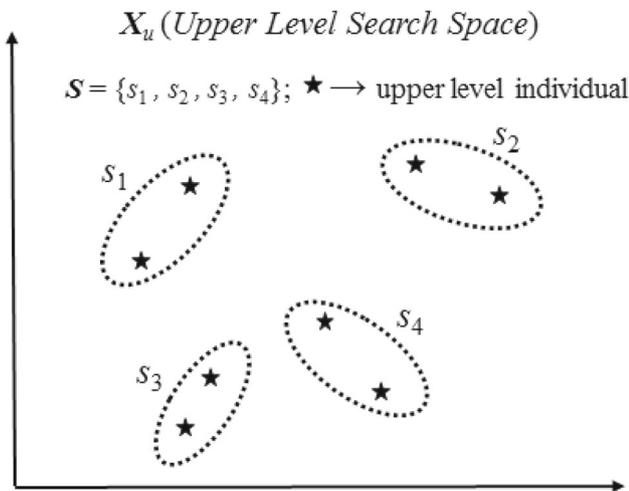


**Fig. 2** An illustration of a typical BLOP where multiple lower level optimization tasks are to be solved at once corresponding to different upper level candidate solutions

ing commonalities. Based on this contention, the genetically modified individuals in every generation of the upper level EA are first clustered into a set of groups denoted as  $S = \{s_1, s_2, \dots, s_G\}$ ; an instantiation is depicted in Fig. 3. Note that the number of clusters or groups  $G$  is determined by the maximum allowable size of each cluster, which in turn depends on the scalability of the evolutionary multitasking engine used to solve the lower level optimization tasks. So for instance, if the multitasking engine is known to perform up to 3 tasks effectively, then  $G$  may be chosen to be roughly  $N/3$ , where  $N$  is the size of the upper level population. Thereafter, lower level optimization tasks corresponding to individuals within each cluster  $s_g \in S$  are performed simultaneously using the MFEA. By doing so, it becomes possible to exploit the underlying commonalities among them, thereby potentially accelerating convergence to high quality lower level solutions. A summary of the steps involved in the Multitasking Bi-Level Evolutionary Algorithm (M-BLEA) is provided in Algorithm 4. Notice that the only distinction between the N-BLEA and the M-BLEA is with regard to the presence of lower level evolutionary multitasking. Thus, any significant variation in the performance of the two algorithms can be entirely attributed to this particular aspect of the respective approaches.

**Algorithm 4:** The Multitasking-BLEA (M-BLEA)

1. Randomly generate  $N$  individuals to form initial population  $P_{u,0}$
2. Cluster  $P_{u,0}$  into a set of groups  $\mathcal{S}$
3. **for every** group  $s_g$  in  $\mathcal{S}$  **do**  
     Solve lower level optimization tasks of all  $p_j \in s_g$  together via MFEA  
     Evaluate every  $p_j$  in  $s_g$  as  $F(\mathbf{x}_{u,j}, \mathbf{x}_{l,j}^*)$
4. **end for**
5. Set  $T = 0$
6. **while** (stopping conditions are not satisfied) **do**  
      $C_{u,T} = \text{Crossover+Mutate}(P_{u,T})$   
     Cluster  $C_{u,T}$  into a set of groups  $\mathcal{S}$   
     **for every** group  $s_g$  in  $\mathcal{S}$  **do**  
         Solve lower level optimization tasks of all  $c_j \in s_g$  together via MFEA  
         Evaluate every  $c_j$  in  $s_g$  as  $F(\mathbf{x}_{u,j}, \mathbf{x}_{l,j}^*)$   
     **end**  
      $R_{u,T} = C_{u,T} \cup P_{u,T}$   
     Select  $N$  fittest members from  $R_{u,T}$  to form  $P_{u,T+1}$   
     Set  $T = T + 1$
7. **end while**



**Fig. 3** An illustrative example of how upper level population members in the M-BLEA may be clustered into a set of four groups, i.e.,  $G = 4$

## 4 Benchmark computational studies

In this section, we present results for a series of computational experiments that demonstrate the efficacy of our proposed method on synthetic benchmark problems. The unconstrained single-objective bi-level test problems considered herein have recently been proposed in [26,29]. The salient feature of the test problems, which are labeled as the SMD benchmarks, is that they are scalable and showcase various levels of difficulty that are commonly encountered in practical bi-level optimization problems. In the present study, only ten-dimensional variants of the benchmarks are considered. For a complete description of their mathematical formulation, the reader is referred to [26,29]. The details have not been reproduced in this paper for the sake of brevity.

### 4.1 Algorithmic specifications

The purpose of this experimental study is to highlight the performance benefits achievable purely by the scope of evolutionary multitasking when applied to BLOPs. Thus, all peripheral aspects of the N-BLEA and the M-BLEA, including the encoding scheme, crossover operator, mutation operator, and parameter settings, are kept identical. Accordingly, any performance improvements that may be achieved during multitasking can be entirely attributed to the exploitation of underlying commonalities between tasks by the process of implicit genetic transfer. At both levels of the bi-level optimizers we employ the random-key encoding scheme, the SBX operator for crossover, and polynomial mutation [13]. Crossover probability of  $p_c = 1$  with 50 % chance of variable exchange between offspring, and mutation probability of  $p_m = 1/5$ , are used. Finally, we set the crossover index and the mutation index as  $\eta_c = 20$  and  $\eta_m = 20$ , respectively.

With regard to population size, the N-BLEA employs 20 individuals at both levels. This implies that at every generation of the upper level EA,  $N = 20$  genetically modified individuals are created. Subsequently, corresponding to each upper level individual, the lower level optimization task is comprehensively solved by evolving another population of  $n = 20$  individuals. Thus, an exorbitantly large number of lower level function evaluations will typically be required to appropriately resolve a BLOP using the N-BLEA.

To overcome the challenge mentioned above, the MFEA has been incorporated into the proposed M-BLEA. Interestingly, the M-BLEA also employs 20 individuals at both levels. Without loss of generality, before executing the lower level optimization tasks, the upper level population is clustered into ten groups (i.e.,  $G = |\mathcal{S}| = 10$ ) via Euclidean distance-based single-linkage clustering, with each group comprising 2 individuals (i.e.,  $|s_g| = 2$ , for all  $g \in \{1, 2, \dots, G\}$ ). Note that any alternate clustering mechanism and/or larger cluster sizes may also be readily tested without any changes to the overall algorithm. The present settings have been fixed throughout the paper as a simple proof-of-concept, and also because the scalability of the current implementation of the MFEA is yet to be thoroughly verified for handling larger number of tasks at a time. Our main aim here is simply to showcase the potential advantages brought to the table by multitasking in bi-level optimization. Under the current settings, it is implied that the MFEA concurrently tackles two lower level optimization tasks that correspond to the two upper level individuals existing within the same cluster. Thus, the  $n = 20$  evolving individuals in the MFEA are evenly split between two lower level tasks which are likely to have underlying commonalities/similarities between them due to their proximity in the upper level search space (as was illustrated in Fig. 3). It is expected that the MFEA will

automatically harness the available synergies by the process of implicit genetic transfer, thereby providing a significant impetus to the evolutionary search.

#### 4.1.1 Termination criteria

The algorithms follow a variance-based termination criteria at both levels [27]. In particular, when the numeric value of the evolving quantity  $\alpha_u$  (described below) becomes less than a pre-defined value  $\alpha_u^{stop}$ , sufficient convergence is assumed to be achieved and the upper level EA terminates.

$$\alpha_u = \sum_{i=1}^{D_u} \frac{\sigma^2(x_{u,i,T})}{\sigma^2(x_{u,i,0})}. \tag{4}$$

Here,  $x_{u,i,T}$  represents the  $i$ th upper level design variable at generation  $T$ . Note that the upper level termination criterion is identical in the N-BLEA and the M-BLEA, with  $\alpha_u^{stop}$  set to  $1e-3$ .

At the lower level, however, the termination criterion is slightly different for the M-BLEA and the N-BLEA due to the presence (or absence) of multiple lower level optimization tasks in a single multitasking environment. In the traditional N-BLEA, the evolving quantity  $\alpha_l^{N-BLEA}$  is described as follows,

$$\alpha_l^{N-BLEA} = \sum_{i=1}^{D_l} \frac{\sigma^2(x_{l,i,t})}{\sigma^2(x_{l,i,0})}, \tag{5}$$

where  $x_{l,i,t}$  indicates the  $i$ th lower level design variable at generation  $t$  of the lower level EA. In contrast, in the M-BLEA, the evolving quantity  $\alpha_l^{M-BLEA}$  is described as.

$$\alpha_l^{M-BLEA} = \max_{\forall k} \left\{ \sum_{i=1}^{D_l} \frac{\sigma^2(x_{l,i,k,t})}{\sigma^2(x_{l,i,k,0})} \right\}, \tag{6}$$

where  $x_{l,i,k,t}$  indicates the  $i$ th lower level design variable of the  $k$ th task at generation  $t$  of the MFEA. Recall that under the current settings  $k \in \{1, 2\}$ . The respective algorithms are terminated when  $\alpha_l^{N-BLEA}$ ,  $\alpha_l^{M-BLEA}$  become less than  $\alpha_l^{stop}$ , which is fixed at  $1e-5$ . Superior convergence is demanded at the lower level since inaccurate lower level solutions are likely to mislead the upper level EA whenever there is a conflict between the two levels.

#### 4.2 Numerical results and discussions

The efficacy of the evolutionary multitasking paradigm is illustrated by the comparison study presented in Tables 1 and 2 for examples without and with upper level objective function multi-modality, respectively. The results have been deduced from 31 independent runs of each solver. The performance of the optimizers has been quantified in terms of the total number of lower level function evaluations required for convergence (i.e., for the upper level termination criterion to be satisfied).

**Table 1** N-BLEA vs. M-BLEA in terms of lower level function evaluations (LL-FE) for SMD1 to SMD5

Label	N-BLEA					M-BLEA				
	Best LL-FE	Median LL-FE	Mean LL-FE	Worst LL-FE	UL-A	Best LL-FE	Median LL-FE	Mean LL-FE	Worst LL-FE	UL-A
SMD1	1,591,720	2,248,240	2,274,605.16	2,912,200	0.00245	<b>670,000</b>	<b>984,340</b>	<b>976,641.94</b>	<b>1,243,960</b>	0.00213
SMD2	1,742,000	2,470,360	2,540,604.52	4,017,360	0.00103	<b>624,440</b>	<b>927,940</b>	<b>953,762.58</b>	<b>1,558,580</b>	0.00286
SMD3	1,780,120	2,756,640	2,725,602.58	3,297,260	0.00253	<b>902,060</b>	<b>1,137,960</b>	<b>1,133,180.65</b>	<b>1,352,800</b>	0.00355
SMD4	2,368,540	3,694,120	3,584,107.74	4,613,840	0.00129	<b>809,340</b>	<b>1,282,560</b>	<b>1,280,561.29</b>	<b>1,758,000</b>	0.00007
SMD5	2,358,080	3,633,300	3,678,349.03	5,194,180	0.00036	<b>697,360</b>	<b>1,264,460</b>	<b>1,244,645.81</b>	<b>1,741,320</b>	0.00017

Best, median, mean, and worst, outcomes achieved across 31 independent runs of the solvers are presented. Superior LL-FEs for each test problem are depicted in bold. The upper level accuracy achieved is denoted as UL-A

**Table 2** N-BLEA vs. M-BLEA in terms of lower level function evaluations (LL-FE) for SMD7 and SMD8

Label	N-BLEA					M-BLEA				
	Best LL-FE	Median LL-FE	Mean LL-FE	Worst LL-FE	UL-A	Best LL-FE	Median LL-FE	Mean LL-FE	Worst LL-FE	UL-A
SMD7	1,680,560	2,977,080	3,069,575.48	5,038,120	0.00567	<b>920,400</b>	<b>1,347,840</b>	<b>1,326,115.48</b>	<b>1,868,960</b>	0.00396
SMD8	2,125,840	4855,880	4,752,300.00	6,909,280	0.01022	<b>1,404,920</b>	<b>2,025,660</b>	<b>2,045,438.06</b>	<b>2,707,560</b>	0.00053

Best, median, mean, and worst, outcomes achieved across 31 independent runs of the solvers are presented. Superior LL-FEs for each test problem are depicted in bold. The upper level accuracy achieved is denoted as UL-A

#### 4.2.1 Examples with convex upper level objective functions

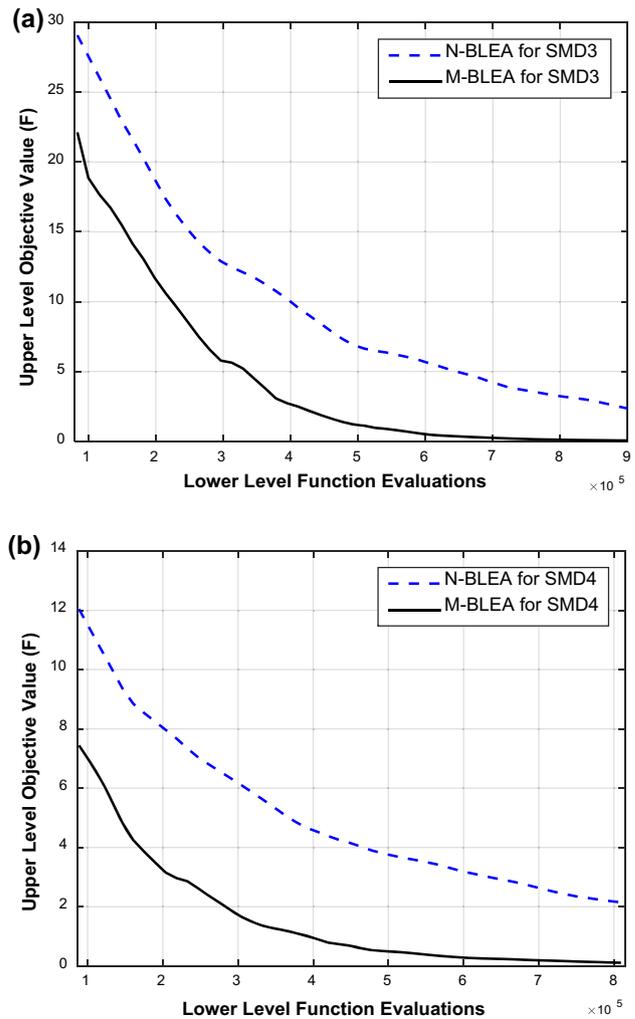
As can clearly be seen in Table 1, the performance of the M-BLEA is significantly superior in comparison to N-BLEA with respect to the number of lower level function evaluations required for convergence. This is true for best, median, mean, and worst-case outcomes (over 31 runs) for benchmark problems SMD1 to SMD5 that are characterized by convex upper level objective functions. In fact, in all cases, the worst-case performance of the M-BLEA is found to be better than the best performance of the N-BLEA. Note that the example SMD6 has been left out in the experiments as it is characterized by infinitely many optimums at the lower level for any given upper level design vector, of which only one corresponds to the best upper level value [29]. From a practical standpoint, such a scenario will often cause uncertainty at the upper level as the optimum response from the lower level optimization task is indecisive. The M-BLEA is thus not currently equipped to handle related uncertain and/or multimodal optimization cases in bi-level optimization.

Further, Table 1 presents the accuracy achieved by the two algorithms for upper level objective function values, the theoretical optimum being equal to zero in all cases. Focusing on the averaged (mean) performance of either algorithm, we find that the M-BLEA saves at least 57.06 % of lower level function evaluations in comparison to N-BLEA. The observed saving is in consonance with the fact that in the current implementation of the M-BLEA, the MFEA at the lower level handles two (similar or related) optimization tasks (that are associated with the same cluster) simultaneously in multitasking. Thus, the total number of lower level function evaluations can be expected to be *at least* halved as compared to N-BLEA, which is indeed found to be the case.

To further emphasize on the improvements brought to the table by multitasking, we refer to Fig. 4 which depicts the averaged convergence trends attained by the N-BLEA and M-BLEA for test problems SMD3 and SMD4. These two problems especially pose notable challenges to the process of bi-level optimization due to the presence of complex multimodality in the lower level objective function landscape [29]. As can be seen in Fig. 4a, b, multitasking successfully provides a significant impetus to the evolutionary search by appropriately harnessing the complementarity between related lower level optimization tasks via implicit genetic transfer, and thereby leading to considerably accelerated convergence towards high quality solutions.

#### 4.2.2 Examples with multimodal upper level objective functions

The complexity of instances SMD7 and SMD8 emerges from the fact that there exist multimodalities in the upper level



**Fig. 4** Comparison of the averaged convergence trends achieved by the N-BLEA and the M-BLEA for **a** SMD3 and **b** SMD4

objective function landscape. Nevertheless, the outcome achieved is very much similar to the previous case where M-BLEA is found to considerably outperform N-BLEA by saving more than 50 % of the lower level function evaluations required for convergence. The results are reported in Table 2.

## 5 A real-world case study in composites manufacturing

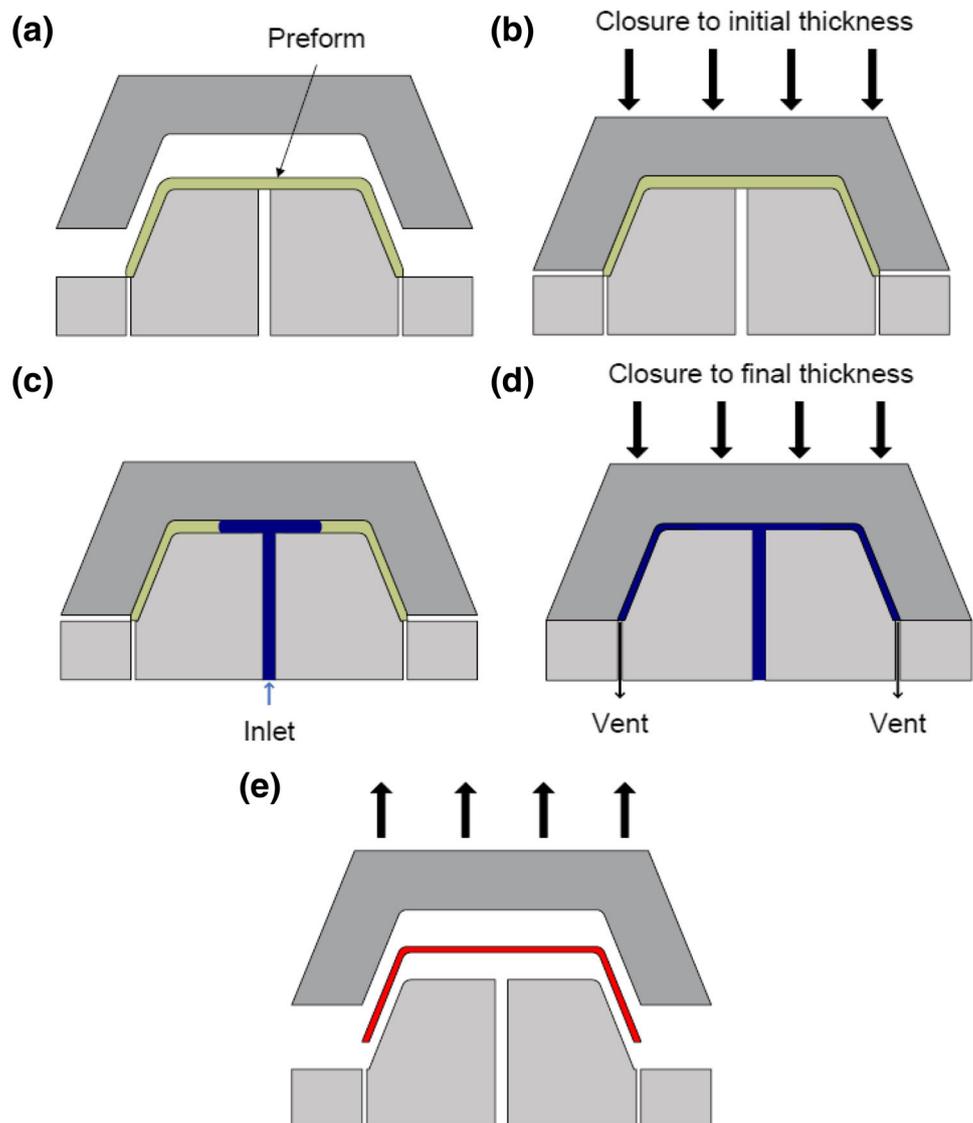
In [17], the comprehensive optimization of rigid tool Liquid composite molding (LCM) processes was established as a BLOP. Accordingly, in this case study, we consider the simulation-based optimization of a kind of LCM process, namely, compression resin transfer molding (CRTM), for the manufacture of a synthetic fiber-reinforced polymer (FRP) composite part. CRTM is a popular technique for high vol-

ume production of FRP parts and is characterized by the use of high stiffness molds. Under the assumption of rigidity, the molds are expected to undergo negligible deflection in response to the large internal forces that originate from the cumulative effect of high fluid (polymer resin) pressure and the compression of the fibrous reinforcement. As a result, CRTM processes find widespread application in areas where tight geometrical tolerances are to be met, such as in the automobile industry. However, it turns out that sophisticated peripheral equipment (such as a hydraulic press) is generally needed to equilibrate the large internal forces. Thus, an important aim in the optimal design of the CRTM process is to maximize throughput (by minimizing manufacturing time) and quality of the finished product, while simultaneously satisfying the constraints placed by the availability and/or running costs of peripheral equipment.

### 5.1 A brief overview of the CRTM process

Before presenting the real-world BLOP associated with composites manufacturing, we present a brief overview of the CRTM cycle. The setup of the cycle, as illustrated in Fig. 5, comprises a metal mold machined according to the geometry of the composite part to be manufactured. First, a preform of the fibrous reinforcement is placed into the mold cavity (step [a] in Fig. 5). The mold is then *partially* closed, mildly compressing the fibrous reinforcement, thereby leaving sufficient gap for the liquid resin to flow through without much hindrance (step [b] in Fig. 5). Prior to injecting the resin, the mold is preheated to a preferred operation temperature. Thereafter, a measured volume of liquid thermosetting resin (which is also preheated) is injected into the mold at high pressure (step [c] in Fig. 5). Finally, the mold is *completely* closed in situ to the final thickness of the part using a force-controlled mech-

**Fig. 5** Steps of the CRTM cycle: **a** preform placement, **b** partial mold closure, **c** resin injection, **d** in situ mold closure to final part thickness, **e** resin solidification and part extraction [33]



anism, thereby fully compressing the fibrous reinforcement (step [d] in Fig. 5). Note that steps [a] to [d] constitute the first phase of the CRTM cycle, which we shall refer to hereafter as the *mold filling phase*. In the second phase, the filled mold is allowed to rest (under controlled thermal conditions) until the resin solidifies, followed by part extraction (step [e] in Fig. 5). Accordingly, step [e] constitutes the *curing phase* of the manufacturing cycle.

## 5.2 The associated BLOP

While researchers have traditionally attempted to optimize the two phases of the manufacturing cycle separately, in [17, 18], it was established that more comprehensive results may be obtained by combining the two phases as a single BLOP. In particular, the curing phase of the CRTM cycle was viewed as the upper level of the BLOP, while the mold filling phase constituted the lower level. For a detailed discussion on the arguments corroborating the bi-level formulation, the reader is referred to [17]. Herein, we simply state the mathematical formulation of the composites manufacturing optimization problem as has been solved in this paper.

$$\text{Minimize } F = 0.7 \left( \frac{\text{TEMP}_{\text{diff}}}{20} \right) + 0.3 \left( \frac{\text{Curing time} - 1000}{3000} \right), \quad (7)$$

$$\text{subject to minimization of } f = \text{Mold filling time,} \\ \text{such that } F_{\text{fluid}} + F_{\text{fibre}} \leq F_{\text{capacity}}. \quad (8)$$

As mentioned earlier, the upper level of the BLOP comprises primarily the curing phase, while the lower level problem is formed by the mold filling phase. To be precise, in Eq. (7),  $\text{TEMP}_{\text{diff}}$  equates to the maximum difference in temperature between the surface and the core of the part during the entire curing cycle and represents a heuristic prediction of the final quality of the composite part. The higher the temperature difference, the greater the chance of residual thermal stresses that diminish the quality of the finished part. As is self-explanatory, the *Curing Time* (also appearing in Eq. 7) represents the time needed for the resin to solidify (step [e] in Fig. 5), while the *Mold Filling Time* (appearing in Eq. 8) represents the time needed for the mold to be filled with resin (i.e., the time needed to perform step [a] to step [d] in Fig. 5). Clearly, the *Curing Time* and the *Mold Filling Time* are quantities that must preferably be minimized to maximize throughput. Finally, the lower level constraint (in Eq. 8) accommodates for the restrictions placed by the peripheral equipment. In particular,  $F_{\text{fluid}}$  represents the internal force originating from the high pressure resin injection,  $F_{\text{fibre}}$  is the reaction force from the compressed fibrous reinforce-

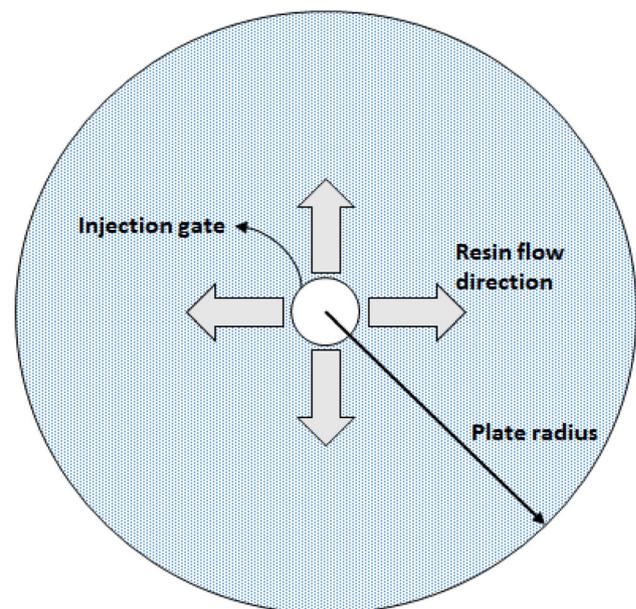
ment, and  $F_{\text{capacity}}$  is the prescribed capacity of the available hydraulic press.

Next, we specify the design variables corresponding to each phase. At the upper level, the temperature profile of the mold during the entire manufacturing cycle (denoted as  $\text{TEMP}_{\text{mold}}^{\text{profile}}$ ) constitutes  $x_u$ . For the lower level optimization task, which is parametrized by  $\text{TEMP}_{\text{mold}}^{\text{profile}}$ , the design space  $X_l$  comprises (a) the mold cavity thickness during resin injection ( $H_{\text{inj}}$ ), (b) resin injection pressure ( $P_{\text{inj}}$ ), and (c) preheated resin temperature ( $\text{TEMP}_{\text{resin}}$ ).

To conclude the problem formulation, it is stated that the computation of upper and lower level objectives is carried out via sophisticated process simulation algorithms which numerically evaluate a series of partial differential equations governing the complex non-isothermal and reactive resin behavior in the fibrous media. Details of the algorithms are not presented here for the sake of brevity. The interested reader is referred to [16, 32]. It is important to note that these simulations are typically computationally expensive, consuming several minutes for a single function evaluation of acceptable fidelity for a complex composite part.

## 5.3 Experimental setup and numerical results

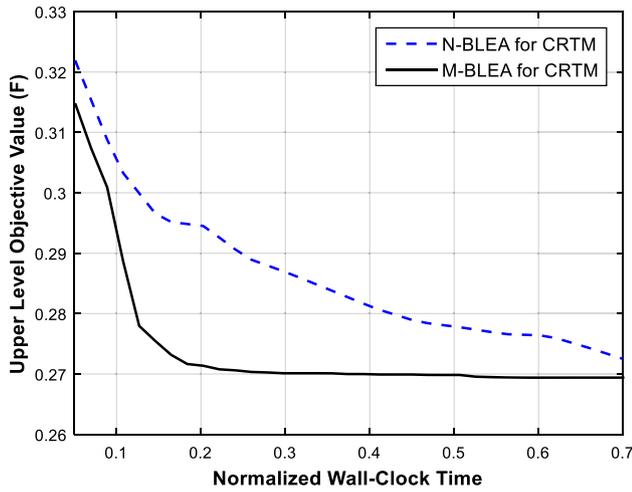
We undertake the simulation-based optimization of the manufacturing processes for a circular FRP composite plate. The diameter of the plate is 1 m, with a central injection hole of 2 cm (see Fig. 6). The final thickness of the part is 0.75 cm. A glass-fiber chopped strand mat forms the reinforcing material with final fiber volume fraction of 35 %, while an epoxy resin



**Fig. 6** A schematic of the circular FRP composite plate with central injection hole

**Table 3** Extent of the design space for composites manufacturing design variables

Description	Lower bound	Upper bound
Lower level design variables ( $x_l$ )		
Resin injection pressure ( $P_{inj}$ )	200 kPa	750 kPa
Cavity thickness during injection ( $H_{inj}$ )	0.8 cm	1 cm
Resin temperature ( $TEMP_{resin}$ )	293 K (20 °C)	348 K (75 °C)
Upper level design variables ( $x_u$ )		
Mold temperature during filling ( $TEMP_{mold}^{filling}$ )	293 K (20 °C)	333 K (60 °C)
Mold heat rate ( $Q_{mold}$ )	2 K/min	30 K/min
Final mold temperature ( $TEMP_{mold}^{final}$ )	373 K (100 °C)	433 K (160 °C)



**Fig. 7** Comparison of averaged convergence trends achieved by the N-BLEA and the M-BLEA for the composites manufacturing problem

system forms the polymer matrix. For complete details of the material properties needed for the simulations, the reader is once again referred to [16]. Finally, it is stated that a highly restrictive  $F_{capacity} = 2$  tons ( $2E+04$  N) is dictated by the available peripheral equipment.

The extent of the design space for all design variables is specified in Table 3 (the table includes variables corresponding to the lower and upper levels). Note that  $TEMP_{mold}^{profile} = \{TEMP_{mold}^{filling}, Q_{mold}, TEMP_{mold}^{final}\}$ . To account for the computational expense of the simulations, the upper level EA is

executed for a maximum of 50 generations, while  $\alpha_l^{stop}$  is set to  $1e-3$ . The remaining algorithmic specifications are kept the same as in Sect. 4.

As is depicted in Fig. 7, the convergence trends (averaged over three independent runs) of the M-BLEA are considerably improved as compared to that of N-BLEA. The impetus provided to the evolutionary search, as a by-product of implicit genetic transfer in evolutionary multitasking, is found to significantly cut down on the design time for this complex engineering design problem. To elaborate, we find that in achieving the upper level objective function target of 0.28, the M-BLEA saves approximately 65 % in wall-clock time. This outcome highlights the utility of the proposed methodology in the fast-paced practical settings of today where one is faced with tight deadlines and stringent time constraints.

Further, to confirm the robustness of the M-BLEA, we present the final (optimized) upper level design variables attained by both algorithms at termination, in each of the three independent runs. The results are reported in Table 4. As can be seen, both algorithms converge to more-or-less the same design vector on each run, thereby verifying the accuracy and repeatability of the proposed approach.

## 6 Conclusions

The significance of this paper lies in revealing the effectiveness of the novel evolutionary multitasking paradigm as a promising means to enhance the performance of bi-level

**Table 4** Testing the robustness of the M-BLEA

Run number	N-BLEA			M-BLEA		
	$TEMP_{mold}^{filling}$ (K)	$Q_{mold}$ (K/min)	$TEMP_{mold}^{final}$ (K)	$TEMP_{mold}^{filling}$ (K)	$Q_{mold}$ (K/min)	$TEMP_{mold}^{final}$ (K)
1	298.87	5.08	432.74	301.07	4.99	432.99
2	297.60	5.31	429.92	301.73	5.17	432.94
3	301.73	4.97	432.63	298.40	5.23	432.79

The table presents the optimum upper level design variables obtained from each of the three independent runs for the composites manufacturing problem

optimization. It is found that BLOPs provide a natural environment wherein multiple (related) optimization tasks are to be solved at the same time, thereby presenting the perfect platform for evolutionary multitasking to autonomously harness the relationships between tasks through the process of implicit genetic transfer. To this end, a recently proposed evolutionary multitasking engine, labeled as a multifactorial evolutionary algorithm (MFEA), has been merged into an evolutionary bi-level optimizer. Several benchmark test functions, and a real-world case study in complex engineering design, demonstrate the efficiency of the multitasking paradigm in exploiting the underlying commonalities between optimization tasks, thereby providing a significant impetus to the evolutionary search. The numerical results indicate that the proposed Multitasking bi-level evolutionary algorithm (M-BLEA) successfully converges to high quality solutions in significantly lesser number of lower level function evaluations (and implicitly in significantly lesser amount of wall-clock time) as compared to the basic nested bi-level evolutionary algorithm (N-BLEA) upon which it has been built.

The present study has investigated the utility of the M-BLEA in a real-world setting involving expensive simulation-based function evaluations. With regard to future works, we will focus on further improving the algorithm to tackle real-time bi-level optimization problems, such as those occurring in security games. One possible means of achieving further speed up of the M-BLEA is the incorporation of quadratic approximation models (or other surrogate modeling techniques) at the lower level, instead of repeatedly having to solve the lower level optimization task in an evolutionary manner, as has very recently been proposed in [28].

**Acknowledgments** This work was conducted within the Rolls-Royce@NTU Corporate Lab with support from the National Research Foundation (NRF) Singapore under the Corp Lab@University Scheme.

**Open Access** This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

## References

- Avigad G, Eisenstadt E, Cohen MW (2011) Optimal strategies for multi objective games and their search by evolutionary multi objective optimization. In: 2011 IEEE conference on computational intelligence and games (CIG). IEEE, pp 166–173
- Avigad G, Eisenstadt E, Glizer VY (2013) Evolving a Pareto front for an optimal bi-objective robust interception problem with imperfect information. In: EVOLVE—a bridge between probability, set oriented numerics, and evolutionary computation II. Springer, Berlin, Heidelberg pp 121–135
- Bäck T (1996) Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms. Oxford University Press, Oxford
- Bard JF, Falk JE (1982) An explicit solution to the multi-level programming problem. *Comput Oper Res* 9(1):77–100
- Bean JC (1994) Genetic algorithms and random keys for sequencing and optimization. *ORSA J Comput* 6(2):154–160
- Cavalli-Sforza LL, Feldman MW (1973) Cultural versus biological inheritance: phenotypic transmission from parents to children (a theory of the effect of parental phenotypes on children's phenotypes). *Am J Hum Genet* 25(6):618
- Chen X, Ong YS, Lim MH, Tan KC (2011) A multi-facet survey on memetic computation. *IEEE Trans Evolut Comput* 15(5):591–607
- Cloninger CR, Rice J, Reich T (1979) Multifactorial inheritance with cultural transmission and assortative mating. II. A general model of combined polygenic and cultural inheritance. *Am J Hum Genet* 31(2):176
- Cloninger CR, Rice J, Reich T (1979) Multifactorial inheritance with cultural transmission and assortative mating. III. Family structure and the analysis of separation experiments. *Am J Hum Genet* 31(3):366
- Colson B, Marcotte P, Savard G (2007) An overview of bilevel optimization. *Ann Oper Res* 153(1):235–256
- Dawkin R (1976) The selfish gene. Oxford University Press, Oxford
- Deb K, Agrawal RB (1994) Simulated binary crossover for continuous search space. *Complex Syst* 9(3):1–15
- Deb K, Deb D (2014) Analysing mutation schemes for real-parameter genetic algorithms. *Int J Artif Intell Soft Comput* 4(1):1–28
- Deb K, Sindhya K, Okabe T (2007) Self-adaptive simulated binary crossover for real-parameter optimization. In: Proceedings of the genetic and evolutionary computation conference (GECCO-2007), UCL London, pp 1187–1194
- Feng L, Ong YS, Lim M, Tsang I (2014) Memetic search with inter-domain learning: a realization between CVRP and CARP. *IEEE Trans Evolut Comput* 19(5):644–658
- Gupta A, Kelly PA, Bickerton S, Walbran WA (2012) Simulating the effect of temperature elevation on clamping force requirements during rigid-tool liquid composite moulding processes. *Compos Part A: Appl Sci Manuf* 43(12):2221–2229
- Gupta A, Kelly PA, Ehrgott M, Bickerton S (2013) A surrogate model based evolutionary game-theoretic approach for optimizing non-isothermal compression RTM processes. *Compos Sci Technol* 84:92–100
- Gupta A, Kelly P, Ehrgott M, Bickerton S (2013) Applying bi-level multi-objective evolutionary algorithms for optimizing composites manufacturing processes. In: Evolutionary multi-criterion optimization. Springer, Berlin, Heidelberg, pp 615–627
- Gupta A, Ong YS, Feng L (2015) Multifactorial evolution: towards evolutionary multitasking. *IEEE Trans Evol Comput*. doi:10.1109/TEVC.2015.2458037 (accepted)
- Handoko SD, Gupta A, Kim HC, Chuin LH, Soon OY, Siew TP (2015) Solving multi-vehicle profitable tour problem via knowledge adoption in evolutionary bi-level programming. In: 2015 IEEE congress on evolutionary computation (CEC). IEEE, pp 2713–2720
- Karwowski J, Mańdziuk J (2015) A new approach to security games. In: International conference on artificial intelligence and soft computing. Lecture notes in artificial intelligence, vol 9120. Springer International Publishing, pp 402–411
- Krawiec K, Wieloch B (2010) Automatic generation and exploitation of related problems in genetic programming. In: 2010 IEEE congress on evolutionary computation (CEC). IEEE, pp 1–8
- Ong YS, Lim MH, Chen X (2010) Research frontier-memetic computation-past, present and future. *IEEE Comput Intell Mag* 5(2):24

24. Pan SJ, Yang Q (2010) A survey on transfer learning. *Knowl Data Eng IEEE Trans* 22(10):1345–1359
25. Rice J, Cloninger CR, Reich T (1978) Multifactorial inheritance with cultural transmission and assortative mating. I. Description and basic properties of the unitary models. *Am J Hum Genet* 30(6):618
26. Sinha A, Malo P, Deb K (2012) Unconstrained scalable test problems for single-objective bilevel optimization. In: 2012 IEEE congress on evolutionary computation (CEC). IEEE, pp 1–8
27. Sinha A, Malo P, Deb K (2013) Efficient evolutionary algorithm for single-objective bilevel optimization. [arXiv:1303.3901](https://arxiv.org/abs/1303.3901) (arXiv preprint). Accessed 11 Jan 2016
28. Sinha A, Malo P, Deb K (2014) An improved bilevel evolutionary algorithm based on quadratic approximations. In: 2014 IEEE congress on evolutionary computation (CEC). IEEE, pp 1870–1877
29. Sinha A, Malo P, Deb K (2014) Test problem construction for single-objective bilevel optimization. *Evolut Comput* 22(3):439–477
30. Sinha A, Malo P, Frantsev A, Deb K (2013). Multi-objective Stackelberg game between a regulating authority and a mining company: a case study in environmental economics. In: 2013 IEEE congress on evolutionary computation (CEC). IEEE, pp 478–485
31. Sinha A, Malo P, Frantsev A, Deb K (2014) Finding optimal strategies in a multi-period multi-leader-follower Stackelberg game using an evolutionary algorithm. *Comput Oper Res* 41:374–385
32. Verleye B, Walbran WA, Bickerton S, Kelly PA (2011) Simulation and experimental validation of force controlled compression resin transfer molding. *J Compos Mater* 45(7):815–829
33. Walbran WA (2011) Experimental validation of local and global force simulations for rigid tool liquid composite moulding processes. Doctoral dissertation, ResearchSpace, Auckland
34. Wang JY, Ehrgott M, Dirks KN, Gupta A (2014) A bilevel multi-objective road pricing model for economic, environmental and health sustainability. *Transport Res Proc* 3:393–402
35. Wen UP, Hsu ST (1991) Linear bi-level programming problems—a review. *J Oper Res Soc* 42(2):125–133