Bilevel Optimization based on Iterative Approximation of Mappings

Ankur Sinha[1], Zhichao Lu[2], Kalyanmoy Deb[2] and Pekka Malo[3]

[1]Production and Quantitative Methods
Indian Institute of Management, Ahmedabad 380015 India
asinha@iima.ac.in

[2]Department of Electrical and Computer Engineering
Michigan State University, East Lansing, MI, USA
luzhicha@msu.edu, kdeb@egr.msu.edu

[3]Department of Information and Service Economy
Aalto University School of Business, PO Box 21220, 00076 Aalto, Finland
pekka.malo@aalto.fi

Abstract—**A large number of application problems involve two levels of optimization, where one optimization task is nested inside the other. These problems are known as bilevel optimization problems and have been studied by both classical optimization community and evolutionary optimization community. Most of the solution procedures proposed until now are either computationally very expensive or applicable to only small classes of bilevel optimization problems adhering to mathematically simplifying assumptions. In this paper, we propose an evolutionary optimization method that tries to reduce the computational expense by iteratively approximating two important mappings in bilevel optimization; namely, the lower level rational reaction mapping and the lower level optimal value function mapping. The algorithm has been tested on a large number of test problems and comparisons have been performed with other algorithms. The results show the performance gain to be quite significant. To the best knowledge of the authors, a combined theory-based and population-based solution procedure utilizing mappings has not been suggested yet for bilevel problems.**

Index Terms—**Bilevel optimization, Evolutionary Algorithms, Stackelberg Games, Mathematical Programming**

I. Introduction

Interest in bilevel optimization has been growing due to a number of new applications that are arising in different fields of science and engineering. Bilevel programming is quite common in the area of defense where these problems are studied as attacker-defender problems. The problem was introduced by Bracken and McGill [9] in the area of mathematical programming, where an inner optimization problem acts as a constraint to an outer optimization problem. One of the follow-up papers by Bracken and McGill [10] highlighted the applications of bilevel programming in defense. Since then a number of studies on homeland security [12], [52], [3] have been performed, where it is common to have bilevel, trilevel and even multilevel optimization models. In the area of operations research, bilevel optimization is gaining importance in the context of interdiction and protection of hub-and-spoke networks [26], as most of the critical infrastructures like transportation and communications are predominantly hub-and-spoke. In other game theoretic settings, bilevel optimization has been used in transportation [33], [15], [11], optimal tax policies [25], [42], [39], investigation of strategic behavior in deregulated markets [22], model production processes [35] and optimization of retail channel structures [55]. The applications extend to a variety of other domains, like, facility location [24], [47], [46], chemical engineering [45], [14], structural optimization [8], [13], and optimal control [34], [2] problems. While new applications that are inherently bilevel in nature are arising at a fast pace, the development of computationally efficient algorithms for such problems has not kept the pace with the applications.

A significant body of literature exists on bilevel optimization and its optimality conditions [30], [18], [17], [19], [54] in the classical optimization literature. However, on the algorithm front most attention has been given to only simple instances of bilevel optimization where the objective functions and constraints are linear [53], [7], quadratic [6], [20], [1] or convex [31]. This is not surprising given the fact that bilevel optimization is difficult to an extent that merely evaluating the bilevel optimality of a given solution is an NP-hard task [48]. Researchers have also attempted to solve these problems using computational techniques like evolutionary algorithms. Most of the bilevel algorithms relying on evolutionary framework have been nested in nature [32], [57], [28], [58]. One of the drawbacks of such an approach is that it might be able to solve small instances of bilevel problems, but as soon as the problem scales-up beyond a few variables,

the computational requirements increase tremendously. However, the evolutionary algorithms still have a niche in solving these problems as it maintains a population at each iteration of the algorithm. A population of points may allow modeling various mappings in bilevel optimization to reduce the computational expense [40]. Some studies in this direction are [41], [44], [36], [37]. We believe that exploiting some of the mathematical properties of bilevel problems through modeling of various mappings in bilevel is the way forward in solving such problems.

In this paper, we focus on two important mappings in bilevel optimization borrowed from the mathematical optimization literature. The first mapping is the lower level reaction set mapping (known as Ψ-mapping), which provides the lower level optimal solution(s) corresponding to any given upper level vector. Considering the upper level problem as the leader's problem and the lower level problem as the follower's problem, the reaction set mapping represents the rational decisions of the follower corresponding to any decision taken by the leader. The second mapping is the lower level value function mapping (known as φ-mapping) that provides the optimal objective function value to the follower's problem for any given leader's decision. While the first mapping can be a set-valued mapping, the second mapping is always single-valued. We work with meta-modeling techniques that try to approximate these two mappings and develop a computationally efficient evolutionary algorithm for solving bilevel problems. The algorithm has been tested on a number of test problems, and the computational gain when compared with other techniques is found to be significant. In this paper, we also extend an existing test-suite of bilevel test problems [38] with a couple of additional problems to better evaluate our proposed solution procedure.

The paper is organized as follows. To begin with, we provide a brief literature survey of bilevel optimization using evolutionary algorithms. This is followed by various formulations of the bilevel optimization problem and discussion of the two mappings that we approximate in this paper. Thereafter, we provide the bilevel evolutionary optimization algorithm which is an extension of the algorithm proposed in the previous studies [44], [36], [37]. Following this, we provide the empirical results on a number of test problems. A comparative study with other approaches is also included. Finally, we end the paper with the conclusions section.

II. A Survey on Evolutionary Bilevel Optimization

Most of the evolutionary algorithms for bilevel optimization are nested in nature, where one optimization algorithm is used within the other. The outer algorithm handles the upper level problem and the inner algorithm handles the lower level problem. Such a structure necessitates that the inner algorithm is called for every upper level point generated by the outer algorithm. Therefore, nested approaches can be quite computationally demanding, and can only be applied to small scale problems. One can find studies with evolutionary algorithm being used for the upper level problem and classical approach being used for the lower level problem. If the lower level problem is complex, researchers have used evolutionary algorithms at both levels. Below we provide a review of evolutionary bilevel optimization algorithms from the past.

Mathieu et al. [32] was one of the first to propose a bilevel algorithm using evolutionary algorithms. He used a genetic algorithm to handle the upper level problem and linear programming to solve the lower level problem for every upper level member generated using genetic operations. This study was followed by nesting the Frank-Wolfe algorithm (reduced gradient method) within a genetic algorithm in Yin [57]. Other authors utilized similar nested schemes in [29], [28], [58]. Studies involving evolutionary algorithms at both levels include [4], [5], where authors have used differential evolution at both levels in the first study, and differential evolution within ant colony optimization in the second study.

Replacing the lower level problem in bilevel optimization with its KKT conditions is a common approach for solving the problem both in classical as well as evolutionary computation literature. Some of the evolutionary studies that utilize this idea include [21], [50]. The approach has been popular and even recently researchers are relying on reducing the bilevel problem into single level problem using KKT and solving the reduced problem using evolutionary algorithm, for example, see [51], [23], [27], [49].

While KKT conditions can only be applied to problems where the lower level adheres to certain mathematically simplifying assumptions, the researchers are exploring techniques that can solve more general instances of bilevel optimization problems. Some of the approaches are based on meta-modeling the mappings within bilevel optimization, while others may be based on meta-modeling the entire bilevel problem itself. Studies in this direction include [44], [36], [37]. In this paper, we aim to develop an algorithm that tries to capture two important mappings in bilevel optimization; namely, the lower level reaction set mapping and the lower level value function mapping, in order to reduce the computational complexity of the problem.

III. Different Bilevel Formulations

We will start this section by providing a general formulation for bilevel optimization. This is followed by various proposals that researchers have made for reducing a bilevel problem into a single-level problem. The two levels in a bilevel problem are also known as

the leader's (upper) and follower's (lower) problems in the domain of game theory. In general, the variables, objectives and constraints are different for the two levels. The upper level variables are treated as parameters while optimizing the lower level problem. A general bilevel formulation has been provided below (for brevity, we ignore equality constraints):

Definition 1: For the upper-level objective function $F : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}$ and lower-level objective function $f : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}$, the bilevel optimization problem is given by

$$\begin{aligned} &\text{“min”}_{x_u \in X_U, x_l \in X_L} F(x_u, x_l) \\ &\text{subject to} \\ &\quad x_l \in \underset{x_l \in X_L}{\operatorname{argmin}} \{f(x_u, x_l) : g_j(x_u, x_l) \leq 0, j = 1, \ldots, J\} \\ &\quad G_k(x_u, x_l) \leq 0, k = 1, \ldots, K \end{aligned}$$

where $G_k : X_U \times X_L \rightarrow \mathbb{R}, k = 1, \ldots, K$ denotes the upper level constraints, and $g_j : X_U \times X_L \rightarrow \mathbb{R}$ denotes the lower level constraints.

The bilevel problem in Definition 1 is ill-defined if there exists more than one lower level optimal solutions for some upper level variables. In such a case, the decision of the lower level remains unclear. However, in case where the lower level optimal solution is single-valued, the lone optimal solution is the rational choice. Optimizing bilevel problems from either optimistic or pessimistic position are two common approaches that researchers have utilized to handle the ambiguity arising from multiple lower level optimal solutions. In an optimistic position, it is assumed that the lower level chooses that optimal solution which is favorable at the upper level. In a pessimistic position, the upper level optimizes its problem according to the worst case scenario. In other words, the lower level may choose a solution from the optimal set that is least favorable at the upper level. In this paper, we assume an optimistic position while solving bilevel optimization problems.

In case when certain mathematically simplifying assumptions like continuities and convexities are satisfied, often the lower level optimization task in Definition 1 is replaced with its KKT conditions. However, the reduced formulation is not simple to handle, as it induces non-convexities and discreteness into the problem through the complementary slackness conditions. We do not utilize any properties of the KKT based reduction in this paper, rather we focus on two different formulations in the development of the evolutionary algorithm in this paper.

A. Lower Level Reaction Set Mapping

The formulation provided in Definition 1 can also be stated as follows:

Definition 2: Let $\Psi : \mathbb{R}^n \rightrightarrows \mathbb{R}^m$ be the reaction set mapping,

$$\Psi(x_u) = \underset{x_l \in X_L}{\operatorname{argmin}} \{f(x_u, x_l) : g_j(x_u, x_l) \leq 0, j = 1, \ldots, J\},$$

which represents the constraint defined by the lower-level optimization problem, i.e. $\Psi(x_u) \subset X_L$ for every $x_u \in X_U$. Then the following gives an alternative formulation for the bilevel optimization problem:

$$\begin{aligned} &\min_{x_u \in X_U, x_l \in X_L} F(x_u, x_l) \\ &\text{subject to} \\ &\quad x_l \in \Psi(x_u) \\ &\quad G_k(x_u, x_l) \leq 0, k = 1, \ldots, K \end{aligned}$$

Using the above definition, a bilevel problem can be reduced to a single level constrained problem given that the Ψ-mapping can somehow be determined. Unfortunately this is rarely the case. Studies in the evolutionary computation literature that rely on iteratively approximation of this mapping to reduce the lower level optimization calls could be found in [44], [36], [37]. To illustrate the idea, let's consider the Figure 1. To acquire sufficient data for constructing the Ψ-mapping approximation, a few lower level problems need to be optimized completely for their corresponding upper level decision vectors in the beginning. For instance, the lower level decisions for the upper level decisions a, b, c, d, e and f are determined by optimizing the lower level problem, which are then used to locally approximate the Ψ-mapping. This has been shown in Figure 1. Even though the actual Ψ-mapping is still unknown, the local approximation can then be substituted to identify the lower level optimal decision for every new upper level member to avoid the lower level optimization task. This procedure of approximating the mapping and utilizing it to predict the lower level optimum needs to be repeated iteratively until convergence to the bilevel optimum. The idea works well when the Ψ-mapping is single valued. If the lower level has multiple optimal solutions for some upper level members as shown in Figure 2, then identifying as well as approximating the mapping is not a straightforward task.

B. Lower Level Optimal Value Function Mapping

Another formulation for the bilevel optimization problem in Definition 1 can be written using the optimal lower level value function: [56]:

Definition 3: Let $\varphi : X_U \rightarrow R$ be the lower level optimal value function mapping,

$$\varphi(x_u) = \min_{x_l \in X_L} \{f(x_u, x_l) : g_j(x_u, x_l) \leq 0, j = 1, \ldots, J\},$$

which represents the optimal function value at the lower level for any given upper level decision vector. Using this lower level optimal value function, the bilevel

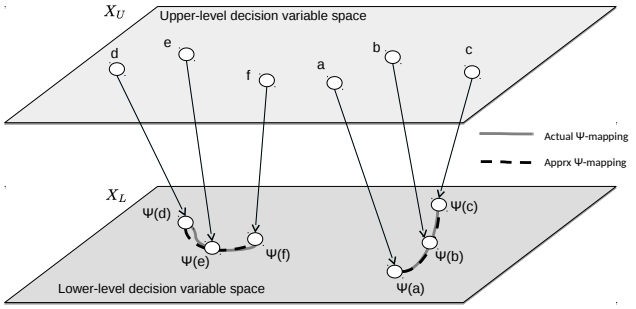

Fig. 1. Solving the lower level optimization problem completely for a, b, c, d, e and f provides the corresponding lower level optimal members $\Psi(a), \Psi(b), \Psi(c), \Psi(d), \Psi(e)$ and $\Psi(f)$, where Ψ-mapping is assumed to be single valued. Such a mapping can be approximated.

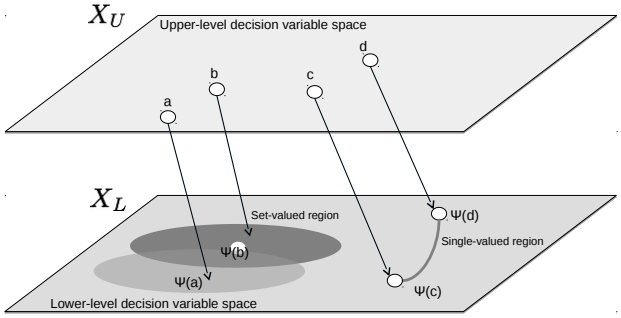

Fig. 2. A scenario where the the Ψ-mapping is set-valued in some regions and single-valued in other regions. If the Ψ-mapping is set-valued then identifying as well as approximating the mapping is not a straightforward task.

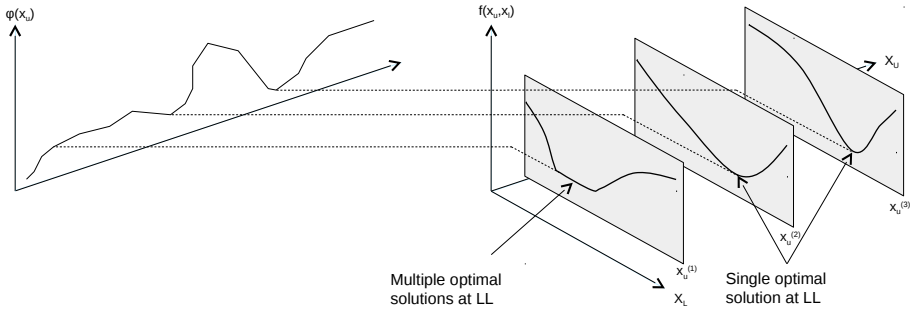

Fig. 3. An example showing a φ-mapping.

optimization problem can be expressed as:

$$\begin{aligned} \min_{x_u \in X_U, x_l \in X_L} \quad & F(x_u, x_l) \\ \text{subject to} \quad & \\ & f(x_u, x_l) \leq \varphi(x_u) \\ & g_j(x_u, x_l) \leq 0, j = 1, \ldots, J \\ & G_k(x_u, x_l) \leq 0, k = 1, \ldots, K. \end{aligned}$$

As in the case of Ψ-mapping, if the φ-mapping can be somehow determined, a bilevel problem can be reduced to a single level problem as described in Definition 3. Along the process of an algorithm, the φ-mapping can be approximated and used to solve the reduced single level problem formulation in an iterative manner. Such an evolutionary algorithm has been recently discussed in [41]. The approximation of the optimal value function (φ) mapping is, in general, less complicated than the reaction set (Ψ) mapping, in the sense that, the φ-mapping is always scalar-valued regardless of the lower level variable dimension and whether or not there exist multiple lower level optimal solutions. However, the φ-mapping based reduction is not necessarily always better than the Ψ-mapping based reduction. Definition 3 requires the problem to be solved with respect to upper as well as lower level variables, while in Definition 2 the lower level variables are readily available from the Ψ-mapping. The Ψ-mapping based reduction also contains fewer constraints. Therefore, clearly there is a trade-off.

IV. Evaluating the Performance of Ψ and φ Mappings on Test Problems

In this section, we implement the Ψ and φ mappings separately in two different nested algorithms to evaluate the advantages and disadvantages of using the two mappings as a local search. For evaluating the two mappings, we choose a set of simple test problems that are provided in Tables X and XI. Firstly, we create a nested algorithm that utilizes an evolutionary approach for solving the upper level problem and sequential quadratic programming (SQP) for solving the lower level problem. Most of the lower level problems in the considered test cases being convex, explains the choice for using sequential quadratic programming (SQP) at

the lower level. We enhance the nested approach by allowing it to approximate the Ψ and φ mappings and measure the performance gain provided by using each of the mappings separately. The implementation of the approaches has been outlined through the Figure 4. The flowchart without the overlapping box provides the steps involved in the nested approach. In case the idea involving Ψ and φ mappings has to be used then the local search (as mentioned in the overlapping box) is conducted every k generations of the nested algorithm after the update step. A detailed description of the nested algorithm has been provided below.

1) Create a random population of size N comprising of upper level variables
2) Solve the lower level optimization problem using SQP for each upper level variable.
3) Evaluate the fitness of each population member using upper level function and constraints (refer to Section VI-B)
4) Choose 2μ population members using tournament selection and apply genetic operators (refer to Section VI-C) to produce λ offspring.
5) Solve the lower level optimization problem using SQP for each offspring.
6) Evaluate the fitness of each offspring using using upper level function and constraints
7) Form a pool consisting of $r + \lambda$ members, where r members are chosen randomly from the population and λ members are the offspring. Use the best r members from this pool to replace the chosen r members from the population.
8) Perform a termination check (refer to Section VI-D) and proceed to Step 5 if termination check is false, otherwise stop.

The parameters used in the implementation of the above procedure are $N = 50, \mu = 2, \lambda = 3$ and $r = 2$. The lower level SQP terminates when the improvement in the lower level function value is less than 1e-6.

A. Approximating the Ψ-mapping

Let $\mathcal{H}$ be the hypothesis space. The hypothesis space consists of all functions that can be used to generate a mapping between the upper level decision vectors and optimal lower level decision vectors. Given a sample $\mathcal{S}$, consisting of upper level points and corresponding optimal lower level points, we would like to identify a model $\hat{\Psi} \in \mathcal{H}$ that minimizes the empirical error on the sample, i.e.

$$\hat{\psi} = \underset{h \in \mathcal{H}}{\operatorname{argmin}} \sum_{i \in \mathcal{I}} L(h(x_u^{(i)}), \bar{x}_l^{(i)}), \tag{1}$$

where $L : X_L \times X_L \rightarrow \mathbb{R}$ denotes the prediction error, $x_u^{(i)}$ is any given upper level vector and $\bar{x}_l^{(i)}$ is its corresponding optimal solution. The prediction error may be calculated as follows:

$$L(h(x_u^{(i)}), \bar{x}_l^{(i)}) = |\bar{x}_l^{(i)} - h(x_u^{(i)})|^2.$$

We have restricted the hypothesis space $\mathcal{H}$ to consist of second-order polynomials which reduces the error minimization problem to an ordinary quadratic regression problem. The sample $\mathcal{S}$ can be created from the population members or an archive. It should be noted that this can approximate only single-valued mapping and will fail if the mapping becomes set-valued.

B. Approximating the φ-mapping

Once again, let $\mathcal{H}$ be the hypothesis space of functions and $\mathcal{S}$ be a sample of upper and corresponding lower level points, our aim is to identify a model $\hat{\varphi} \in \mathcal{H}$ that minimizes the empirical error on the sample, i.e.

$$\hat{\varphi} = \underset{u \in \mathcal{H}}{\operatorname{argmin}} \sum_{i \in \mathcal{I}} L(u(x_u^{(i)}), \bar{f}^{(i)}), \tag{2}$$

where $L : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$ denotes the prediction error, $x_u^{(i)}$ is any given upper level vector and $\bar{f}^{(i)}$ is its corresponding optimal function value. The prediction error can once again be computed as follows:

$$L(u(x_u^{(i)}), f^{(i)}) = |\bar{f}^{(i)} - u(x_u^{(i)})|^2.$$

We have once again restricted the hypothesis space $\mathcal{H}$ to consist of second-order polynomials. Since the φ-mapping is always single valued, approximating it will not involve similar issues as for the Ψ-mapping.

V. Comparison Results for Ψ vs φ Approximations

For comparing Ψ-approximation approach against φ-approximation approach, we use a set of 8 test problems selected from the literature given in Tables X and XI. Table I compares the median function evaluations at both level for three algorithms Ψ-approximation, φ-approximation and nested algorithm. The results have been produced from 31 runs of the algorithm and further details about the runs can be found in Figures 5 and 6. Both Ψ-approximation and φ-approximation perform equally well and outperform the nested approach in this study. The differences in the performance of Ψ-approximation and φ-approximation can be attributed to differences in the quality of approximation produced during the intermediate steps of the algorithm. In Table II, we provide a comparison of the meta-modeling results with other evolutionary approaches [50], [51] to provide an idea about the extent of savings that can be produced using meta-modeling techniques. The advantage is quite clear as the savings are better by multiple order of magnitudes on the set of test problems considered in this study.

It should be noted that the Ψ approximation idea would fail if the Ψ-mapping in bilevel optimization is set valued. Next, we test this hypothesis, by modifying the 8 test problems such that each test problem necessarily has a set-valued Ψ-mapping. To achieve this, we add two additional lower level variables (y_p and y_q) in each test

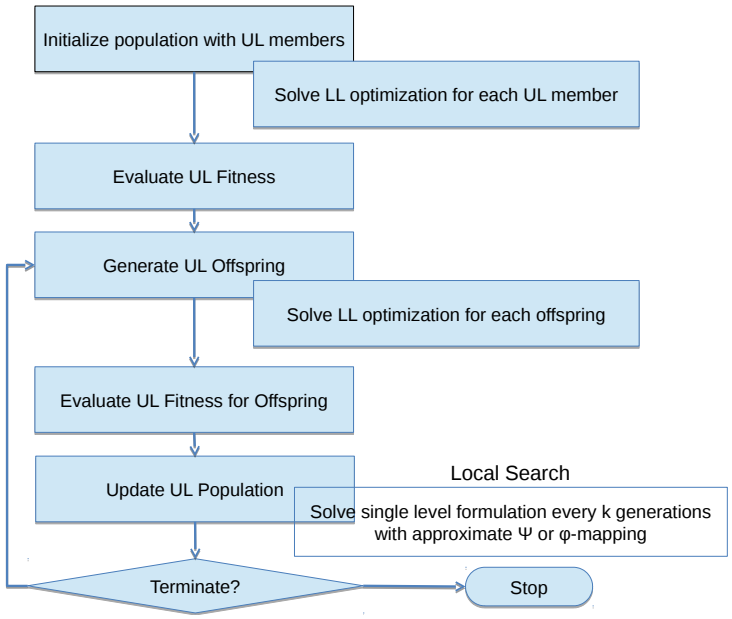

Fig. 4. Nested approach with evolutionary algorithm at upper level (UL) and SQP at lower level (LL). Local search based on Ψ or φ mapping may be performed to make the nested approach faster.

TABLE I
MEDIAN FUNCTION EVALUATIONS REQUIRED AT UPPER LEVEL (UL) AND THE LOWER LEVEL (LL) FROM 31 RUNS OF Ψ-APPROXIMATION ALGORITHM, φ-APPROXIMATION ALGORITHM AND NESTED ALGORITHM.

	UL Func. Evals.			LL Func. Evals.			Savings	
	φ Appx Med	Ψ Appx Med	Nested Med	φ Appx Med	Ψ Appx Med	Nested Med	$\frac{\text{Nested}-\varphi\text{ Appx}}{\text{Nested}}$	$\frac{\text{Nested}-\Psi\text{ Appx}}{\text{Nested}}$
TP1	137	138	-	1539	1948	-	Large	Large
TP2	158	187	444	1614	2820	5252	69%	47%
TP3	198	132	642	2710	1461	6530	59%	78%
TP4	309	419	1760	2976	6347	18073	83%	66%
TP5	165	244	633	2571	2763	6616	62%	59%
TP6	115	91	153	1512	1170	2169	30%	46%
TP7	169	130	196	2386	1426	2606	9%	44%
TP8	201	328	423	2443	4722	8233	69%	42%

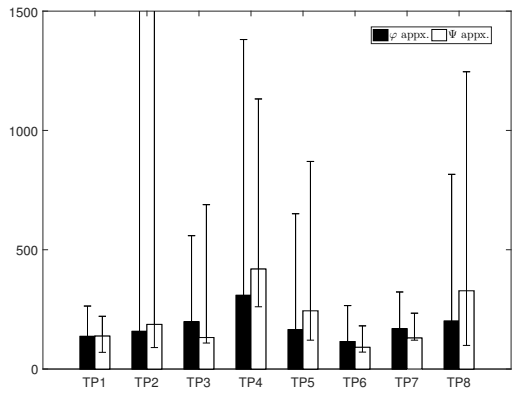

Fig. 5. Error plot from 31 runs for the upper level function evaluations on test problems 1 to 8.

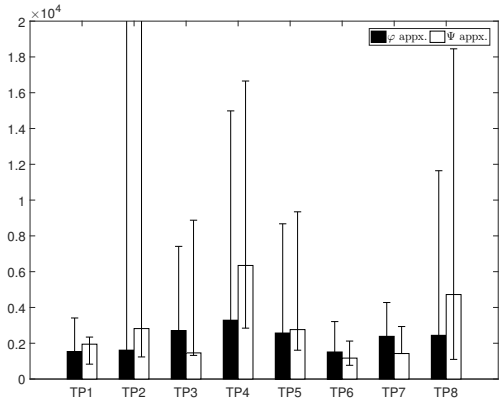

Fig. 6. Error plot from 31 runs for the lower level function evaluations on test problems 1 to 8.

TABLE II
MEAN OF TOTAL FUNCTION EVALUATIONS (UL EVALUATIONS +LL EVALUATIONS) REQUIRED BY DIFFERENT APPROACHES.

	Mean Func. Evals. (UL+LL)				
	φ Appx.	Ψ Appx.	Nested	WJL [50]	WLD [51]
TP1	1,611	2,421	34,462	85,499	86,067
TP2	1,923	3,262	6,235	256,227	171,346
TP3	2,624	1,482	8,125	92,526	95,851
TP4	3,612	6,721	19,948	291,817	211,937
TP5	2,812	3,388	7,398	77,302	69,471
TP6	1,578	1,034	1,534	163,701	65,942
TP7	2,110	1,456	2,286	1,074,742	944,105
TP8	2,734	4,434	5,325	213,522	182,121

TABLE III
STATISTICS FOR UPPER LEVEL FUNCTION EVALUATIONS FOR φ-APPROXIMATION ALGORITHM ON THE MODIFIED TEST PROBLEMS (M-TP). THE φ-APPROXIMATION ALGORITHM AND NESTED ALGORITHM FAIL ON ALL THE MODIFIED TEST PROBLEMS.

	φ Appx.			Ψ Appx.	Nested
	Min	Med	Max	Min/Med/Max	Min/Med/Max
m-TP1	138	192	344	-	-
m-TP2	124	236	-	-	-
m-TP3	140	242	699	-	-
m-TP4	185	545	2,582	-	-
m-TP5	172	242	977	-	-
m-TP6	159	181	559	-	-
m-TP7	119	227	501	-	-
m-TP8	158	462	2,119	-	-

problem. Both the upper and lower level functions are modified as shown below:

$$F^{new}(x_u, x_l) = F(x_u, x_l) + y_p^2 + y_q^2$$
$$f^{new}(x_u, x_l) = f(x_u, x_l) + (y_p - y_q)^2$$
$$y_p, y_q \in [-1, 1]$$

The modification makes the lower level problem have infinitely many optimal solutions (for all $y_p = y_q$) for any given upper level vector. Out of the many optimal solutions the upper level prefers the solution where $y_p = y_q = 0$. After this simple modification, we once again solve the test problems using φ-approximation and Ψ-approximation approaches. As shown in Tables III and IV, the φ-approximation algorithm still works but Ψ-approximation algorithm completely fails. Function evaluations for φ-approximation algorithm increases slightly than before because of additional variables in the problem.

TABLE IV
STATISTICS FOR LOWER LEVEL FUNCTION EVALUATIONS FROM 31 RUNS OF THE φ-APPROXIMATION ALGORITHM ON THE MODIFIED TEST PROBLEMS (M-TP). THE φ-APPROXIMATION ALGORITHM AND NESTED ALGORITHM FAIL ON ALL THE MODIFIED TEST PROBLEMS.

	φ Appx.			Ψ Appx.	Nested
	Min	Med	Max	Min/Med/Max	Min/Med/Max
m-TP1	1,988	2,477	8,334	-	-
m-TP2	2,394	4,420	-	-	-
m-TP3	1,404	3,321	12,353	-	-
m-TP4	1,911	5,632	25,356	-	-
m-TP5	3,129	4,166	15,345	-	-
m-TP6	2,498	3,464	9,325	-	-
m-TP7	1,476	5,635	12,256	-	-
m-TP8	2,721	6,324	28,993	-	-

Therefore, the φ-approximation idea clearly has an advantage over the Ψ-approximation idea. Moreover, φ-mapping is always a scalar valued mapping as compared to Ψ-mapping which is usually vector valued and can also be set-valued. However, there is a trade-off. The reduced single level problem formed using Ψ-mapping may usually be a little easier to handle as compared to the single level problem formed using φ-mapping. The reason being that in case of Ψ-mapping the lower level variables are readily available, and the reduced problem does not involve lower level constraints. For φ-mapping, the reduced problem has to be solved both with respect to upper and lower level variables, and the formulation involves both upper and lower level constraints. Given the pros and cons of using the two mappings, next, we would like to develop an evolutionary algorithm that is capable of utilizing the better of the two mappings while solving a bilevel optimization problem.

VI. BILEVEL EVOLUTIONARY ALGORITHM BASED ON Ψ AND φ-MAPPING APPROXIMATIONS

In this section, we provide the bilevel evolutionary algorithm that approximates the Ψ as well as the φ mapping during the intermediate steps of the algorithm. From the previous experiments and the properties of the two mappings we infer that there can be situations when the approximation of the Ψ-mapping may fail, while when Ψ-mapping can be approximated it offers the advantage of completely ignoring the lower level functions and constraints. Acknowledging this fact, we utilize both the approximations in our algorithm. The algorithm adaptively decides to use one of the mappings based on the quality of fit obtained when approximating the two mappings. Local quadratic approximations are created for the two mappings from a sample of points in the vicinity of the point around which we want to create an approximation. Introducing local approximation, is expected to improve the quality of approximations significantly. Given a sample dataset, the steps for creating an approximation are the same as discussed in Sections IV-A and IV-B. The algorithm also maintains an archive so as to maintain a large dataset for creating and validating the approximations. Deviating from the nested algorithm, we employ the approximated Ψ and φ mappings to avoid frequent lower level optimization calls. An earlier version of the algorithm [44], [36], [37] that relied on Ψ-mapping approximation alone was referred as Bilevel Evolutionary Algorithm based on Quadratic Approximations (BLEAQ). We keep the same terminology and refer to the newer

version of the algorithm as BLEAQ-II. The pseudocode for the algorithm has been provided in Table V.

A. Initialization

The initialization in the algorithm is done by creating random upper level members $x_u^{(1)}, \ldots, x_u^{(N)}$, and then solving the lower level optimization problem for each member to get optimal $x_l^{(1)}, \ldots, x_l^{(N)}$. There can be situations, where finding random feasible upper and lower level pair that satisfy both lower and upper level constraints in the problem can be difficult. In such situations, one can solve the following problem to create $(x_u^{(i)}, x_l^{(i)})$ pairs that satisfies all the constraints to begin with.

$$\begin{aligned} \min_{x_u \in X_U, x_l \in X_L} \quad & 0 \\ \text{subject to} \quad & \\ & G_k(x_u, x_l) \leq 0, k = 1, \ldots, K, \\ & g_j(x_u, x_l) \leq 0, j = 1, \ldots, J. \end{aligned}$$

The above problem can be solved using any standard procedure like a greedy GA or SQP with a random starting point to arrive at a feasible solution. As soon as a feasible member is found, the method stops. Solving the above method repeatedly with random population (in case of GA) or a random starting point (in case of SQP) can provide the starting population of upper level members $x_u^{(1)}, \ldots, x_u^{(N)}$ for the BLEAQ-II algorithm. For this given set of upper level members, we know that at least one feasible lower level member exists and we still need to solve the lower level problem to find the optimal lower level solutions $x_u^{(1)}, \ldots, x_u^{(N)}$. [1]

B. Constraint handling and Fitness Assignment

The proposed approach always assigns higher fitness to a feasible member over a non-feasible member. For two given members, $(x_u^{(i)}, x_l^{(i)})$ and $(x_u^{(j)}, x_l^{(j)})$, if both members are feasible with respect to constraints then it looks at the function value. If both members are infeasible, then it looks at the overall constraint violation. This fitness assignment scheme is similar to the one proposed in [16]. At the lower level the idea can be implemented directly using lower level constraints and lower level function value. At the upper level, for any given upper and lower level pair, we only consider upper level function and constraints, without considering if the corresponding lower level vector is optimal. The information about a lower level vector corresponding to an upper level vector being optimal is stored using tagging (0 or 1).

[1] In case of upper level constraints containing both upper and lower level variables, one can find it difficult to arrive at a $(x_u^{(i)}, x_l^{(i)})$ pair that is feasible with respect to all the constraints and the lower level vector is optimal for the given upper level vector. Many formulations of bilevel optimization, therefore, do not consider lower level variables in upper level constraints.

C. Genetic operations

Offspring are produced in the BLEAQ-II approach using standard crossover and mutation operators. Genetic operations at the upper level involve only upper level variables, and the operations at the lower level involve only lower level variables. We utilize parent centric crossover (PCX) and polynomial mutation for generating the offspring. The crossover operator used in the algorithm is similar to the parent-centric crossover (PCX) operator proposed in [43]. The operator uses three parents and produces offspring around the index parent as described below.

$$c = z^{(p)} + \omega_\xi d + \omega_\eta \frac{p^{(2)} - p^{(1)}}{2} \tag{3}$$

where,

- $z^{(p)}$ is the *index* parent (the best parent among three parents)
- $d = z^{(p)} - g$, where g is the mean of μ parents
- $p^{(1)}$ and $p^{(2)}$ are the other two parents
- $\omega_\xi = 0.1$ and $\omega_\eta = 0.1$ are the two parameters.

D. Termination Criteria

A variance based termination criterion has been used at both levels; some other termination criterion like termination based on no improvement may also be used. Variance based termination allows the algorithm to terminate automatically when the variance of the population becomes small. At the upper level, the variance of the population at any generation, T, is computed as follows:

$$\alpha_u^T = \frac{\sum_{i=1}^n \sigma^2(x_i)|_T}{\sum_{i=1}^n \sigma^2(x_i)|_0}, \tag{4}$$

When the value of α_u^T at any generation T becomes less than α_u^{stop} then the algorithm terminates. In the above equation, n is the number of upper level variables, $\sigma^2(x_i)|_T$ is the variance across dimension i at generation T and $\sigma^2(x_i)|_0$ is the variance across dimension i in the initial population. A similar termination scheme can be used when the evolutionary algorithm is executed at the lower level.

E. Lower Level Optimization

At the lower level, we utilize SQP if the problem is convex, otherwise we use the lower level evolutionary algorithm described in Table VI that uses similar genetic operations as used at the upper level.

F. Offspring Update

For an offspring $x^{(j)} = (x_u^{(j)}, x_l^{(j)})$, the lower level vector $x_l^{(j)}$ is updated either using Ψ-approximation or φ-approximation. An update using Ψ-approximation is straightforward. However, if an update has to be done using φ-approximation, it requires to solve the following auxiliary optimization problem. In the auxiliary problem x_u is fixed as $x_u^{(j)}$ and the problem is solved only with

TABLE V
STEP-BY-STEP PROCEDURE FOR BLEAQ-II

Step	Description
1	**Initialization**: Generate an initial upper level population $x_u^{(1)}, \ldots, x_u^{(N)}$ randomly or by a problem-specific method (see Section VI-A). **(a)** For each $x_u^{(j)}$, find a corresponding optimal lower level solution $x_l^{(j)} \in \Psi(x_u^{(j)})$ by solving the lower level problem. Set $\mathcal{P} = \{(x_u^{(j)}, x_l^{(j)}), j = 1, \ldots, N\}$ (see Section VI-E). **(b)** Tag all vectors $(x_u^{(j)}, x_l^{(j)}) \in \mathcal{P}$ for which a lower level optimization has been successfully performed as **1** and store them in the archive $\mathcal{A}$. **(c)** Assign fitness to all the members based on upper level function and constraints (refer to Section VI-B).
2	**Reproduction**: **(a) Parent selection**: Randomly choose 2μ members from the population $\mathcal{P}$, and perform a tournament selection based on the upper level fitness. This produces μ parents, denoted by $\mathcal{P}_{\text{par}}$. **(b) Offspring generation**: Create λ offsprings, denoted by $\mathcal{P}_{\text{off}}$, from the set of parents $\mathcal{P}_{\text{par}}$ using genetic operators (refer to Section VI-C).
3	**Offspring Update**: For each offspring $x^{(j)} = (x_u^{(j)}, x_l^{(j)}) \in \mathcal{P}_{\text{off}}$ produced in the previous step, update the lower level decision $x_l^{(j)}$ using the following strategy: **(a) Optimization**: If the number of Tag **1** members in $\mathcal{P}$ is less than half of the size of $\mathcal{P}$, perform lower level optimization to ensure that $x_l^{(j)} \in \Psi(x_u^{(j)})$ (as described in Step 1.(b)). If the lower level optimization is successful, tag the offsprings as **1** and add it to the archive $\mathcal{A}$. **(b) Approximations**: If the number of Tag **1** members in $\mathcal{P}$ is more than half of the size of $\mathcal{P}$, for each offspring $x^{(j)} = (x_u^{(j)}, x_l^{(j)}) \in \mathcal{P}_{\text{off}}$, use its neighboring members in the archive $\mathcal{A}$ to construct a local quadratic approximation for Ψ-mapping (q_Ψ) as well as φ-mapping (q_φ). Compare the mean squared error of the approximations (e_{mse}^{Ψ}, e_{mse}^{φ}). If $e_{mse}^{\Psi} \leq e_{mse}^{\varphi}$ then update the lower level decision associated with the upper level $x_u^{(j)}$ by setting $x_l^{(j)} = q_\Psi(x_u^{(j)})$; otherwise solve the auxiliary optimization problem in Section VI-F by fixing $x_u^{(j)}$ and varying $x_l^{(j)}$; the optimal $x_l^{(j)}$ is paired with $x_u^{(j)}$ to form the offspring.
4	**Improvements**: Identify the Tag **1** member in the current generation in $\mathcal{P}$ with the best fitness, denoted as $x_{best}^{(j)}$. Perform a local search in the vicinity of $x_{best}^{(j)}$ after every k generations and update $x_{best}^{(j)}$ if there is an improvement. **(a) Local search**: Construct local quadratic approximations of both Ψ-mapping and φ-mapping using members in the vicinity of $x_{best}^{(j)}$ in the archive $\mathcal{A}$ and record the mean squared error of the approximations (e_{mse}^{Ψ}, e_{mse}^{φ}). Apply local search in $x_{best}^{(j)}$ vicinity using one of the two single level reduction methods described in Sections III-A and III-B (refer to Section VI-G).
5	**Termination check:** Perform a termination check. If false, proceed to the next generation (Step 2).

TABLE VI
THE LOWER LEVEL EVOLUTIONARY ALGORITHM IS DESCRIBED BELOW THAT TAKES AN UPPER LEVEL MEMBER AS INPUT AND SOLVES THE CORRESPONDING LOWER LEVEL PROBLEM.

Step	Description
1	**Initialization**: Generate an initial lower level population $x_l^{(1)}, \ldots, x_l^{(N)}$ randomly and assign fitness using lower level objective and constraints.
2	**Genetic Operations**: Randomly choose 2μ members from the population, and perform a tournament selection leading to μ parents. Create λ offspring using the genetic operations described in Section VI-C. Assign fitness to each offspring.
3	**Update**: Choose r members randomly from the population and pool them with λ offspring. Sort the pool by fitness and replace the r members from the population by the best r members from the pool.
4	**Termination check:** Perform a termination check as described in Section VI-D. If false, proceed to the next generation (Step 2).

respect to x_l. The optimal x_l replaces the lower level vector $x_l^{(j)}$ of the offspring.

$$\begin{aligned} \min_{x_l \in X_L} \quad & \hat{F}(x_u, x_l) \\ \text{subject to} \\ & \hat{f}(x_u, x_l) \leq \hat{\varphi}(x_u) \\ & \hat{g}_j(x_u, x_l) \leq 0, j = 1, \ldots, J \\ & \hat{G}_k(x_u, x_l) \leq 0, k = 1, \ldots, K. \end{aligned}$$

In the above formulation we use hat for all the functions and constraints as we solve the auxiliary problem on approximated functions and constraints. We use linear approximations for all the constraints, while quadratic approximation is used for the other functions. The auxiliary problem may have to be solved frequently if the lower level problem contains multiple optimal solutions. Solving the auxiliary problem with approximated functions helps in saving actual function evaluations. Note that in the ideal case the auxiliary problem will lead to an optimistic lower level solution corresponding to the fixed $x_u^{(j)}$.

G. Local Search

The algorithm utilizes local search after every k generations of the algorithm. The local search is performed by meta-modeling the upper and lower level functions and constraints along with the Ψ and the φ-mappings in the vicinity of the best member in the population. Once the Ψ and the φ-mappings are available, the quality of the two mappings are assessed by the mean square error of the approximations (i.e. e_{mse}^{Ψ} and e_{mse}^{φ}). The better mapping and the corresponding single level reduction (described in Section III-A and III-B) with approximated functions is solved using SQP to arrive at $x_u^{(LS)}$. A lower level optimization corresponding to $x_u^{(LS)}$ is solved and if the member is found to be better than $x_{best}^{(j)}$ then $x_{best}^{(j)}$ is updated. In case the member is not better than the best member found so far, then the next local search is performed using the exact upper/lower level objective functions and constraints.

H. Parameters and Platform

The algorithm has been implemented in MATLAB. At the upper and lower level, the parameters used in the algorithm are:

1) $\mu = 3$
2) $\lambda = 2$
3) $r = 2$
4) Probability of crossover = 0.9
5) Probability of mutation = 0.1
6) N = 50 (Population size at upper level)
7) n = 50 (Population size at lower level)

VII. RESULTS

In this study, we consider three algorithms, the nested approach described in Figure 4, BLEAQ [44], [36], [37], and our proposed BLEAQ-II. To assess the performance of each algorithm, 31 runs have been performed for each test instance. During every simulation process, the algorithm is terminated if the function value accuracy reaches the objective function accuracy of 10^{-2} at both levels. For each run, the upper and lower level function evaluations required until termination is recorded separately.

A. Standard test problems

We first present the empirical results on 8 standard test problems selected from the literature (referred to as TP1-TP8). The description for these test problems has been provided in the Appendix A. Table VII contains the median upper level (UL) function evaluations, lower level (LL) function evaluations and BLEAQ-II's overall function evaluation savings as compared to other approaches from 31 runs of the algorithms. The overall function evaluations for any algorithm is simply the sum of upper and lower level function evaluations. For instance, for

TABLE VII
MEDIAN FUNCTION EVALUATIONS ON TP TEST SUITE

	UL Func. Evals.			LL Func. Evals.			BLEAQ-II Savings	
	BLEAQ-II Med	BLEAQ Med	Nested Med	BLEAQ-II Med	BLEAQ Med	Nested Med	$\frac{\text{BLEAQ}-\text{BLEAQ-II}}{\text{BLEAQ}}$	$\frac{\text{Nested}-\text{BLEAQ-II}}{\text{Nested}}$
TP1	136	155	-	242	867	-	63%	98%
TP2	255	185	436	440	971	5,686	40%	63%
TP3	158	155	633	224	894	6,867	64%	98%
TP4	198	357	1,755	788	1,772	19,764	54%	98%
TP5	272	243	576	967	1,108	6,558	8%	84%
TP6	161	155	144	323	687	1,984	43%	97%
TP7	112	255	193	287	987	2,870	68%	95%
TP8	241	189	403	467	913	7,996	36%	61%

the median run with TP1, BLEAQ-II requires 63% less overall function evaluations as compared to BLEAQ, and 98% less overall function evaluations as compared to the nested approach.

All these test problems are bilevel problems with small number of variables, and all the three algorithms were able to solve the 8 test instances successfully. A significant computational saving can be observed for both BLEAQ-II and BLEAQ, as compared to the nested approach as shown in the Savings column of Table VII. The performance gain going from BLEAQ to BLEAQ-II is quite significant for these simple test problems even though none of them lead to multiple lower level optimal solutions. Detailed comparison between BLEAQ and BLEAQ-II in terms of upper and lower level function evaluations is provided through Figures 7 and 8.

B. Scalable test problems

Next, we compare the results for the three algorithms on the scalable SMD test suite which contains 12 test problems in the original paper [38]. We extend this test suite in this paper to a set of 14 test problems by adding two additional scalable test problems. The description for the additional SMD test problems can be found in Appendix B. First we analyze the performance of the algorithms on a smaller version of the test problems which consists of 5 variables, and then we provide the comparison results on 10-variable instances of the SMD test problems. For the 5 variable version of the SMD test problems, we used the settings as $p = 1$, $q = 2$ and $r = 1$ for all SMD problems except SMD6 and SMD14. For the 5 variable version of SMD6 and SMD14, we used $p = 1$, $q = 0$, $r = 1$ and $s = 2$. For the 10 variable version of the SMD test problems, we used the settings as $p = 3$, $q = 3$ and $r = 2$ for all SMD problems except SMD6 and SMD14. For the 10 variable version of SMD6 and SMD14, we used $p = 3$, $q = 1$, $r = 2$ and $s = 2$.

Table VIII provides the median function evaluations and overall savings for the three algorithms on the set of 14 SMD problems. These test problems contain 2 variables at the upper level and 3 variables at the lower level and offer a variety of tunable complexities to the algorithms. For instances, the test set contains problems which are multimodal at the upper and the lower levels, contain multiple optimal solutions at the lower level, contain constraints at the upper and/or lower levels etc. It can be found that BLEAQ-II is able to solve the entire set of 14 SMD test problems, while BLEAQ fails on 2 test problems. The overall savings with BLEAQ-II is higher as compared to BLEAQ for all the test problems. The test problems that contain multiple lower level solutions include SMD6 and SMD14, for which BLEAQ is unable to handle the problem. Further details about the required overall function evaluations from 31 runs can be found in Figures 9.

Results for the high dimensional SMD test problems have been provided in Table IX. BLEAQ-II leads to much higher savings as compared to BLEAQ, and with higher dimensions BLEAQ is found to once again fail on SMD6 and also on SMD7 and SMD8. Both methods outperform the nested method on most of the test problems. We do not provide results for SMD9 to SMD14 as none of the algorithms were able to handle these problems. It is noteworthy that SMD9 to SMD14 offer difficulties like multi-modalities and highly constrained regions, which none of the algorithms were able to handle with the parameter setting used in this paper. Details for the 31 runs on each of these test problems can be found in Figure 10.

Through Figures 11 and 12, we provide the quality of prediction of the lower level optimal solution made by the Ψ-mapping and φ-mapping approach over the course of the algorithm. It is interesting to note that the quality of φ-approximation is better in the case of SMD1 test problem in Figure 11, therefore, the prediction decisions are mostly made using the φ-approximation approach. However, for SMD13 in Figure 12, which involves a difficult φ-mapping, the prediction decisions are made using the Ψ-approximation approach. Both these mappings are found to be improving with an increase in generations of the algorithm. The two figures show the adaptive nature of the BLEAQ-II algorithm in choosing the right

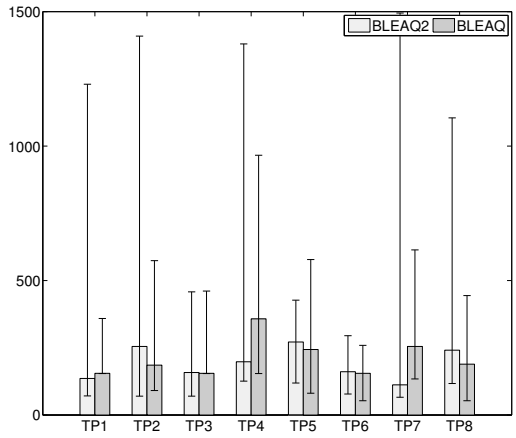

Fig. 7. Bar chart (31 runs/samples) for the upper level function evaluations required for TP 1 to 8.

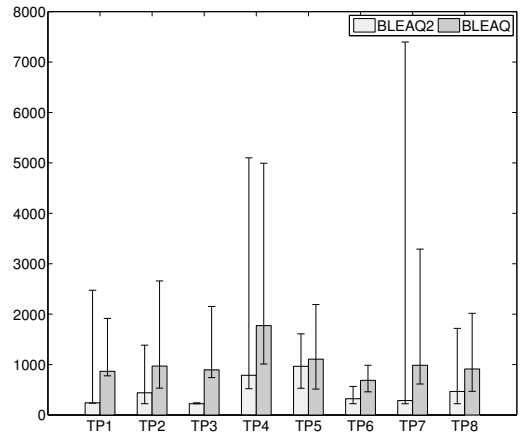

Fig. 8. Bar chart (31 runs/samples) for the lower level function evaluations required for TP 1 to 8.

TABLE VIII
MEDIAN FUNCTION EVALUATIONS ON LOW DIMENSION SMD TEST SUITE

	UL Func. Evals.			LL Func. Evals.			BLEAQ-II Savings	
	BLEAQ-II Med	BLEAQ Med	Nested Med	BLEAQ-II Med	BLEAQ Med	Nested Med	$\frac{\text{BLEAQ}-\text{BLEAQ-II}}{\text{BLEAQ}}$	$\frac{\text{Nested}-\text{BLEAQ-II}}{\text{Nested}}$
SMD1	123	98	164	8,462	13,425	104,575	37%	92%
SMD2	114	88	106	7,264	11,271	74,678	35%	90%
SMD3	264	91	136	12,452	15,197	101,044	17%	87%
SMD4	272	110	74	8,600	12,469	59,208	29%	85%
SMD5	126	80	93	14,490	19,081	73,500	24%	80%
SMD6	259	-	116	914	-	3,074	Large	63%
SMD7	180	98	67	8,242	12,580	56,056	34%	85%
SMD8	644	228	274	22,866	35,835	175,686	35%	87%
SMD9	201	125	127	10,964	16,672	101,382	34%	89%
SMD10	780	431	-	19,335	43,720	-	54%	Large
SMD11	1735	258	260	134,916	158,854	148,520	14%	8%
SMD12	203	557	-	25,388	135,737	-	81%	Large
SMD13	317	126	211	13,729	17,752	138,089	21%	90%
SMD14	1,014	-	168	12,364	-	91,197	Large	85%

approximation strategy based on the difficulties involved in a bilevel optimization problem.

VIII. CONCLUSIONS

In this paper, we have presented a computationally efficient evolutionary algorithm for solving bilevel optimization problems. The algorithm is based on iterative approximations of two important theoretically motivated mappings; namely, the lower level rational reaction mapping and the lower level optimal value function mapping. The paper discusses about the pros and cons of utilizing these mappings in an evolutionary bilevel optimization algorithm by embedding them in a nested approach. Thereafter, an algorithm is developed that adaptively decides to use one of the mappings during the execution based on the characteristics of the bilevel optimization problem being solved. The proposed algorithm has been tested on a wide variety of bilevel test problems and it has been able to perform significantly better than other approaches in terms of computational requirements.

REFERENCES

[1] F. Al-Khayyal, R. Horst, and P. Pardalos. Global optimization of concave functions subject to quadratic constraints: an application in nonlinear bilevel programming. *Annals of Operations Research*, 34:125–147, 1992.

[2] Sebastian Albrecht, K. Ramirez-Amaro, Federico Ruiz-Ugalde, David Weikersdorfer, M. Leibold, Michael Ulbrich, and Michael Beetz. Imitating human reaching motions using physically inspired optimization principles. In *Humanoid Robots (Humanoids), 2011 11th IEEE-RAS International Conference on*, pages 602–607. IEEE, 2011.

[3] B. An, F. Ordóñez, M. Tambe, E. Shieh, R. Yang, C. Baldwin, J. DiRenzo III, K. Moretti, B. Maule, and G. Meyer. A Deployed Quantal Response-Based Patrol Planning System for the U.S. Coast Guard. *Interfaces*, 43(5):400–420, 2013.

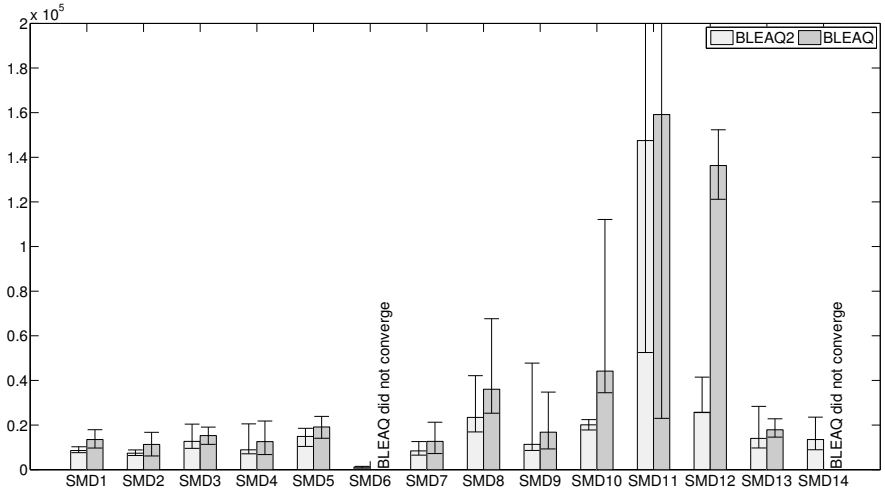

Fig. 9. Bar chart for overall function evaluations for SMD 1 - 14.

TABLE IX
MEDIAN FUNCTION EVALUATIONS ON HIGH DIMENSION SMD TEST SUITE

	UL Func. Evals.			LL Func. Evals.			BLEAQ-II Savings	
	BLEAQ-II Med	BLEAQ Med	Nested Med	BLEAQ-II Med	BLEAQ Med	Nested Med	$\frac{\text{BLEAQ}-\text{BLEAQ-II}}{\text{BLEAQ}}$	$\frac{\text{Nested}-\text{BLEAQ-II}}{\text{Nested}}$
SMD1	670	370	760	52,866	61,732	1,776,426	14%	97%
SMD2	510	363	652	44,219	57,074	1,478,530	22%	97%
SMD3	1369	630	820	68,395	90,390	1,255,015	23%	94%
SMD4	580	461	765	35,722	59,134	1,028,802	39%	96%
SMD5	534	464	645	65,873	92,716	1,841,569	29%	96%
SMD6	584	-	824	3,950	-	156,2003	Large	99%
SMD7	1,486	-	-	83,221	-	-	Large	Large
SMD8	6,551	-	-	231,040	-	-	Large	Large

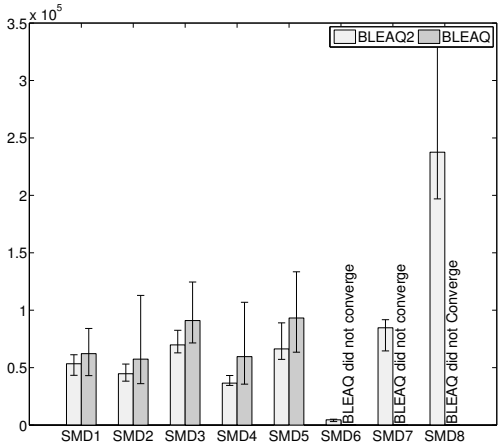

Fig. 10. Bar chart for overall function evaluations for 10-dimension SMD 1 - 8.

[4] J. Angelo, E. Krempser, and H. Barbosa. Differential evolution for bilevel programming. In *Proceedings of the 2013 Congress on Evolutionary Computation (CEC-2013)*. IEEE Press, 2013.

[5] Jaqueline S. Angelo and Helio J. C. Barbosa. A study on the use of heuristics to solve a bilevel programming problem. *International Transactions in Operational Research*, 2015.

[6] J. Bard and J. Moore. A branch and bound algorithm for the bilevel programming problem. *SIAM Journal on Scientific and Statistical Computing*, 11:281–292, 1990.

[7] O. Ben-Ayed. Bilevel linear programming. *Computers and Operations Research*, 20:485–501, 1993.

[8] M. P. Bendsoe. Optimization of structural topology, shape, and material. Technical report, 1995.

[9] J. Bracken and J. McGill. Mathematical programs with optimization problems in the constraints. *Operations Research*, 21:37–44, 1973.

[10] Jerome Bracken and James T. McGill. Defense applications of mathematical programs with optimization problems in the constraints. *Operations Research*, 22(5):1086–1096, 1974.

[11] Luce Brotcorne, Martine Labbe, Patrice Marcotte, and Gilles Savard. A bilevel model for toll optimization on a multicommodity transportation network. *Transportation Science*, 35(4):345–358, 2001.

[12] G. Brown, M. Carlyle, D. Diehl, J. Kline, and K. Wood. A Two-Sided Optimization for Theater Ballistic Missile Defense. *Operations Research*, 53(5):745–763, 2005.

[13] Snorre Christiansen, Michael Patriksson, and Laura Wynter. Stochastic bilevel programming in structural optimization. Technical report, Structural and Multidisciplinary Optimization, 1997.

[14] Peter A. Clark and Arthur W. Westerberg. Bilevel programming for steady-state chemical process designï£¡i. fundamentals and algorithms. *Computers & Chemical Engineering*, 14(1):87–97, 1990.

[15] Isabelle Constantin and Michael Florian. Optimizing frequencies in a transit network: a nonlinear bi-level programming approach. *International Transactions in Operational Research*, 2(2):149 – 164, 1995.

[16] K. Deb. An efficient constraint handling method for genetic algorithms. *Computer Methods in Applied Mechanics and Engineering*, 186(2–4):311–338, 2000.

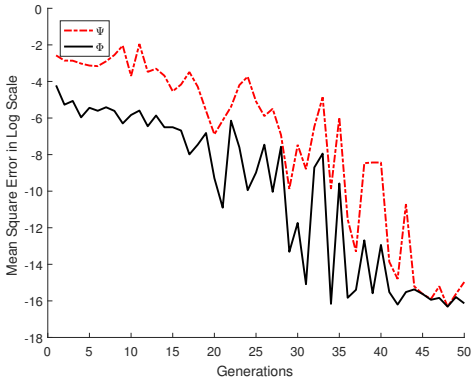

Fig. 11. Approximation error (in terms of Euclidean distance) of a predicted lower level optimal solution when using localized Ψ and φ-mapping during the intermediate generations of the BLEAQ-II algorithm on the 5-variable SMD1 test problem.

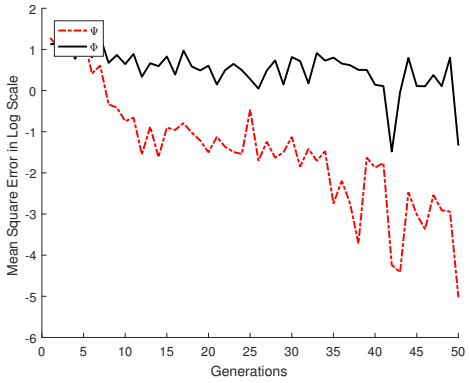

Fig. 12. Approximation error (in terms of Euclidean distance) of a predicted lower level optimal solution when using localized Ψ and φ-mapping during the intermediate generations of the BLEAQ-II algorithm on the 5-variable SMD13 test problem.

[17] S. Dempe, J. Dutta, and B. S. Mordukhovich. New necessary optimality conditions in optimistic bilevel programming. *Optimization*, 56(5-6):577–604, 2007.

[18] Stephan Dempe. *Foundations of Bilevel Programming*. Kluwer Academic Publishers, Secaucus, NJ, USA, 2002.

[19] Stephan Dempe, Boris S. Mordukhovich, and Alain Bertrand Zemkoho. Necessary optimality conditions in pessimistic bilevel programming. *Optimization*, 63(4):505–533, 2014.

[20] T. Edmunds and J. Bard. Algorithms for nonlinear bilevel mathematical programming. *IEEE Transactions on Systems, Man, and Cybernetics*, 21:83–89, 1991.

[21] S. Reza Hejazi, Azizollah Memariani, G. Jahanshahloo, and Mohammad Mehdi Sepehri. Linear bilevel programming solution by genetic algorithm. *Computers & Operations Research*, 29(13):1913–1925, 2002.

[22] X. Hu and D. Ralph. Using EPECs to Model Bilevel Games in Restructured Electricity Markets with Locational Prices. *Operations Research*, 55(5):809–827, 2007.

[23] Yan Jiang, Xuyong Li, Chongchao Huang, and Xianing Wu. Application of particle swarm optimization based on chks smoothing function for solving nonlinear bilevel programming problem. *Applied Mathematics and Computation*, 219(9):4332–4339, 2013.

[24] Qin Jin and Shi Feng. Bi-level simulated annealing algorithm for facility location. *Systems Engineering*, 2:007, 2007.

[25] M. Labbé, P. Marcotte, and G. Savard. A Bilevel Model of Taxation and Its Application to Optimal Highway Pricing. *Management Science*, 44(12):1608–1622, 1998.

[26] Ting L. Lei. Identifying critical facilities in hub-and-spoke networks: A hub interdiction median problem. *Geographical Analysis*, 45(2):105–122, 2013.

[27] Hecheng Li. A genetic algorithm using a finite search space for solving nonlinear/linear fractional bilevel programming problems. *Annals of Operations Research*, pages 1–16, 2015.

[28] Hecheng Li and Yuping Wang. A hybrid genetic algorithm for solving nonlinear bilevel programming problems based on the simplex method. *International Conference on Natural Computation*, 4:91–95, 2007.

[29] Xiangyong Li, Peng Tian, and Xiaoping Min. A hierarchical particle swarm optimization for solving bilevel programming problems. In Leszek Rutkowski, Ryszard Tadeusiewicz, Lotfi A. Zadeh, and Jacek M. Zurada, editors, *Artificial Intelligence and Soft Computing - ICAISC 2006*, volume 4029 of *Lecture Notes in Computer Science*, pages 1169–1178. Springer Berlin Heidelberg, 2006.

[30] Maria Beatrice Lignola and Jacqueline Morgan. Existence of solutions to bilevel variational problems in banach spaces. In *Equilibrium Problems: Nonsmooth Optimization and Variational Inequality Models*, pages 161–174. Springer, 2001.

[31] Guoshan Liu, Jiye Han, and Shouyang Wang. A trust region algorithm for bilevel programing problems. *Chinese science bulletin*, 43(10):820–824, 1998.

[32] R. Mathieu, L. Pittard, and G. Anandalingam. Genetic algorithm based approach to bi-level linear programming. *Operations Research*, 28(1):1–21, 1994.

[33] Athanasios Migdalas. Bilevel programming in traffic planning: Models, methods and challenge. *Journal of Global Optimization*, 7(4):381–405, 1995.

[34] Katja Mombaur, Anh Truong, and Jean-Paul Laumond. From human to humanoid locomotion1an inverse optimal control approach. *Autonomous robots*, 28(3):369–383, 2010.

[35] M.G. Nicholls. Aluminium Production Modeling - A Nonlinear Bilevel Programming Approach. *Operations Research*, 43(2):208–218, 1995.

[36] A. Sinha, P. Malo, and K. Deb. Efficient evolutionary algorithm for single-objective bilevel optimization. *arXiv preprint arXiv:1303.3901*, 2013.

[37] A. Sinha, P. Malo, and K. Deb. An improved bilevel evolutionary algorithm based on quadratic approximations. In *2014 IEEE Congress on Evolutionary Computation (CEC-2014)*, pages 1870–1877. IEEE Press, 2014.

[38] A. Sinha, P. Malo, and K. Deb. Test problem construction for single-objective bilevel optimization. *Evolutionary Computation Journal*, 22(3):439–477, 2014.

[39] A. Sinha, P. Malo, and K. Deb. Transportation policy formulation as a multi-objective bilevel optimization problem. In *2015 IEEE Congress on Evolutionary Computation (CEC-2015)*. IEEE Press, 2015.

[40] A. Sinha, P. Malo, and K. Deb. Evolutionary bilevel optimization: An introduction and recent advances. In S. Bechikh, R. Dutta, and A. Gupta, editors, *Recent Advances in Evolutionary Multi-objective Optimization*. Springer, 2016.

[41] A. Sinha, P. Malo, and K. Deb. Solving optimistic bilevel programs by iteratively approximating lower level optimal value function. In *2016 IEEE Congress on Evolutionary Computation (CEC-2016)*. IEEE Press, 2016.

[42] A. Sinha, P. Malo, A. Frantsev, and K. Deb. 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-2013)*. IEEE Press, 2013.

[43] A. Sinha, A. Srinivasan, and K. Deb. A population-based, parent centric procedure for constrained real-parameter optimization. In *2006 IEEE Congress on Evolutionary Computation (CEC-2006)*, pages 239–245. IEEE Press, 2006.

[44] Ankur Sinha, Pekka Malo, and Kalyanmoy Deb. Evolutionary algorithm for bilevel optimization using approximations of the lower level optimal solution mapping. *European Journal of Operational Research*, 257(2):395–411, 2017.

[45] W.R. Smith and R.W. Missen. *Chemical Reaction Equilibrium Analysis: Theory and Algorithms*. John Wiley & Sons, New York, 1982.

[46] Huijun Sun, Ziyou Gao, and Jianjun Wu. A bi-level programming model and solution algorithm for the location of logistics distribution centers. *Applied Mathematical Modelling*, 32(4):610 – 616, 2008.

[47] Takeshi Uno, Hideki Katagiri, and Kosuke Kato. An evolutionary multi-agent based search method for stackelberg solutions of bilevel facility location problems. *International Journal of Innovative Computing, Information and Control*, 4(5):1033–1042, 2008.

[48] L. N. Vicente, G. Savard, and J. J. Judice. Descent approaches for quadratic bilevel programming. *Journal of Optimization Theory and Applications*, 81(1):379–399, 1994.

[49] Zhongping Wan, Guangmin Wang, and Bin Sun. A hybrid intelligent algorithm by combining particle swarm optimization with chaos searching technique for solving nonlinear bilevel programming problems. *Swarm and Evolutionary Computation*, 8:26–32, 2013.

[50] Y. Wang, Y. C. Jiao, and H. Li. An evolutionary algorithm for solving nonlinear bilevel programming based on a new constraint-handling scheme. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 32(2):221–232, 2005.

[51] Yuping Wang, Hong Li, and Chuangyin Dang. A new evolutionary algorithm for a class of nonlinear bilevel programming problems and its global convergence. *INFORMS Journal on Computing*, 23(4):618–629, 2011.

[52] L. Wein. Homeland Security: From Mathematical Models to Policy Implementation: The 2008 Philip McCord Morse Lecture. *Operations Research*, 57(4):801–811, 2009.

[53] U. Wen and S. Hsu. Linear bi-level programming problems - a review. *Journal of the Operational Research Society*, 42:125–133, 1991.

[54] Wolfram Wiesemann, Angelos Tsoukalas, Polyxeni-Margarita Kleniati, and Berç Rustem. Pessimistic bilevel optimization. *SIAM Journal on Optimization*, 23(1):353–380, 2013.

[55] N. Williams, P.K. Kannan, and S. Azarm. Retail Channel Structure Impact on Strategic Engineering Product Design. *Management Science*, 57(5):897–914, 2011.

[56] Jane J Ye and Daoli Zhu. New necessary optimality conditions for bilevel programs by combining the mpec and value function approaches. *SIAM Journal on Optimization*, 20(4):1885–1905, 2010.

[57] Y. Yin. Genetic algorithm based approach for bilevel programming models. *Journal of Transportation Engineering*, 126(2):115–120, 2000.

[58] Xiaobo Zhu, Qian Yu, and Xianjia Wang. A hybrid differential evolution algorithm for solving nonlinear bilevel programming with linear constraints. In *Cognitive Informatics, 2006. ICCI 2006. 5th IEEE International Conference on*, volume 1, pages 126–131. IEEE, 2006.

APPENDIX A
STANDARD TEST PROBLEMS

In this section, we provide some of the standard bilevel test problems chosen from the literature. Most of these test problems are small with only small number of variables at both levels.

TABLE X
STANDARD TEST PROBLEMS TP1-TP5. (NOTE THAT $x = x_u$ AND $y = x_l$)

Problem	Formulation	Best Known Sol.
TP1		
$n = 2,\ m = 2$	$\underset{(x,y)}{\text{Minimize}}\ F(x,y) = (x_1-30)^2 + (x_2-20)^2 - 20y_1 + 20y_2,$ s.t. $y \in \underset{(y)}{\text{argmin}} \left\{ \begin{array}{l} f(x,y) = (x_1-y_1)^2 + (x_2-y_2)^2 \\ 0 \le y_i \le 10, \quad i = 1,2 \end{array} \right\},$ $x_1 + 2x_2 \ge 30, x_1 + x_2 \le 25, x_2 \le 15$	$F = 225.0$ $f = 100.0$
TP2		
$n = 2,\ m = 2$	$\underset{(x,y)}{\text{Minimize}}\ F(x,y) = 2x_1 + 2x_2 - 3y_1 - 3y_2 - 60,$ s.t. $y \in \underset{(y)}{\text{argmin}} \left\{ \begin{array}{l} f(x,y) = (y_1 - x_1 + 20)^2 + (y_2 - x_2 + 20)^2 \\ x_1 - 2y_1 \ge 10, x_2 - 2y_2 \ge 10 \\ -10 \ge y_i \ge 20, \quad i = 1,2 \end{array} \right\},$ $x_1 + x_2 + y_1 - 2y_2 \le 40,$ $0 \le x_i \le 50, \quad i = 1,2.$	$F = 0.0$ $f = 100.0$
TP3		
$n = 2,\ m = 2$	$\underset{(x,y)}{\text{Minimize}}\ F(x,y) = -(x_1)^2 - 3(x_2)^2 - 4y_1 + (y_2)^2,$ s.t. $y \in \underset{(y)}{\text{argmin}} \left\{ \begin{array}{l} f(x,y) = 2(x_1)^2 + (y_1)^2 - 5y_2 \\ (x_1)^2 - 2x_1 + (x_2)^2 - 2y_1 + y_2 \ge -3 \\ x_2 + 3y_1 - 4y_2 \ge 4 \\ 0 \le y_i, \quad i = 1,2 \end{array} \right\},$ $(x_1)^2 + 2x_2 \le 4,$ $0 \le x_i, \quad i = 1,2$	$F = -18.6787$ $f = -1.0156$
TP4		
$n = 2,\ m = 3$	$\underset{(x,y)}{\text{Minimize}}\ F(x,y) = -8x_1 - 4x_2 + 4y_1 - 40y_2 - 4y_3,$ s.t. $y \in \underset{(y)}{\text{argmin}} \left\{ \begin{array}{l} f(x,y) = x_1 + 2x_2 + y_1 + y_2 + 2y_3 \\ y_2 + y_3 - y_1 \le 1 \\ 2x_1 - y_1 + 2y_2 - 0.5y_3 \le 1 \\ 2x_2 + 2y_1 - y_2 - 0.5y_3 \le 1 \\ 0 \le y_i, \quad i = 1,2,3 \end{array} \right\},$ $0 \le x_i, \quad i = 1,2$	$F = -29.2$ $f = 3.2$
TP5		
$n = 2,\ m = 2$	$\underset{(x,y)}{\text{Minimize}}\ F(x,y) = rt(x)x - 3y_1 - 4y_2 + 0.5t(y)y,$ s.t. $y \in \underset{(y)}{\text{argmin}} \left\{ \begin{array}{l} f(x,y) = 0.5t(y)hy - t(b(x))y \\ -0.333y_1 + y_2 - 2 \le 0 \\ y_1 - 0.333y_2 - 2 \le 0 \\ 0 \le y_i, \quad i = 1,2 \end{array} \right\},$ where $h = \begin{pmatrix} 1 & 3 \\ 3 & 10 \end{pmatrix}, b(x) = \begin{pmatrix} -1 & 2 \\ 3 & -3 \end{pmatrix} x, r = 0.1$ $t(\cdot)$ denotes transpose of a vector	$F = -3.6$ $f = -2.0$

TABLE XI
STANDARD TEST PROBLEMS TP6-TP8. (NOTE THAT $x = x_u$ AND $y = x_l$)

Problem	Formulation	Best Known Sol.
TP6		
$n = 1$, $m = 2$	$\underset{(x,y)}{\text{Minimize}}\ F(x,y) = (x_1 - 1)^2 + 2y_1 - 2x_1$, s.t. $y \in \underset{(y)}{\text{argmin}} \left\{ \begin{array}{l} f(x,y) = (2y_1 - 4)^2 + \\ (2y_2 - 1)^2 + x_1 y_1 \\ 4x_1 + 5y_1 + 4y_2 \leq 12 \\ 4y_2 - 4x_1 - 5y_1 \leq -4 \\ 4x_1 - 4y_1 + 5y_2 \leq 4 \\ 4y_1 - 4x_1 + 5y_2 \leq 4 \\ 0 \leq y_i, \quad i = 1,2 \end{array} \right\}$, $0 \leq x_1$	$F = -1.2091$ $f = 7.6145$
TP7		
$n = 2$, $m = 2$	$\underset{(x,y)}{\text{Minimize}}\ F(x,y) = -\frac{(x_1+y_1)(x_2+y_2)}{1+x_1y_1+x_2y_2}$, s.t. $y \in \underset{(y)}{\text{argmin}} \left\{ \begin{array}{l} f(x,y) = \frac{(x_1+y_1)(x_2+y_2)}{1+x_1y_1+x_2y_2} \\ 0 \leq y_i \leq x_i, \quad i = 1,2 \end{array} \right\}$, $(x_1)^2 + (x_2)^2 \leq 100$ $x_1 - x_2 \leq 0$ $0 \leq x_i, \quad i = 1,2$	$F = -1.96$ $f = 1.96$
TP8		
$n = 2$, $m = 2$	$\underset{(x,y)}{\text{Minimize}}\ F(x,y) = \lvert 2x_1 + 2x_2 - 3y_1 - 3y_2 - 60 \rvert$, s.t. $y \in \underset{(y)}{\text{argmin}} \left\{ \begin{array}{l} f(x,y) = (y_1 - x_1 + 20)^2 + \\ (y_2 - x_2 + 20)^2 \\ 2y_1 - x_1 + 10 \leq 0 \\ 2y_2 - x_2 + 10 \leq 0 \\ -10 \leq y_i \leq 20, \quad i = 1,2 \end{array} \right\}$, $x_1 + x_2 + y_1 - 2y_2 \leq 40$ $0 \leq x_i \leq 50, \quad i = 1,2$	$F = 0.0$ $f = 100.0$

APPENDIX B
ADDITIONAL SMD TEST PROBLEMS

SMD test problems [38] are a set of 12 scalable test problems that offer a variety of controllable difficulties to an algorithm. We add two more test problems to the previous test-suite in this paper. Both these problems contain a difficult φ-mapping, among other difficulties. The upper and lower level functions follow the following structure to induce difficulties due to convergence, interaction, and function dependence between the two levels. The vectors x_u and x_l are further divided into two sub-vectors. The φ-mapping is defined by the function f_1.

$$\begin{aligned} &F(x_u, x_l) = F_1(x_{u1}) + F_2(x_{l1}) + F_3(x_{u2}, x_{l2}) \\ &f(x_u, x_l) = f_1(x_{u1}, x_{u2}) + f_2(x_{l1}) + f_3(x_{u2}, x_{l2}) \\ &\text{where} \\ &\qquad x_u = (x_{u1}, x_{u2}) \quad \text{and} \quad x_l = (x_{l1}, x_{l2}) \end{aligned} \tag{5}$$

TABLE XII
SMD TEST PROBLEMS. (NOTE THAT $(x_{u1}, x_{u2}) = (a, b)$ AND $(x_{l1}, x_{l2}) = (c, d)$)

Problem	Formulation	Solution
SMD13		
	$F_1 = (a_1 - 1)^2 + \sum_{i=1}^{p-1} \left((a_i - 1)^2 + (a_{i+1} - (a_i)^2)^2\right)$,	
	$F_2 = -\sum_{i=1}^{q} \sum_{j=1}^{i} (c_j)^2$,	
	$F_3 = \sum_{i=1}^{r} \sum_{j=1}^{i} (b_j)^2 - \sum_{i=1}^{r} (b_i - \log d_i)^2$,	
	$f_1 = \sum_{i=1}^{p} \left(\lvert a_i \rvert + 2 \lvert \sin(a_i) \rvert\right)$,	$a_i = 1 \; \forall \; i$,
	$f_2 = \sum_{i=1}^{q} \sum_{j=1}^{i} (c_j)^2$,	$b_i = 0 \; \forall \; i$,
	$f_3 = \sum_{i=1}^{r} (b_i - \log d_i)^2$,	$c_i = 0 \; \forall \; i$,
	$a_i \in [-5, 10], \quad \forall \quad i \in \{1, 2, \ldots, p\}$,	$d_i = 1 \; \forall \; i$.
	$b_i \in [-5, e], \quad \forall \quad i \in \{1, 2, \ldots, r\}$,	
	$c_i \in [-5, 10], \quad \forall \quad i \in \{1, 2, \ldots, q\}$,	
	$d_i \in (0, 10], \quad \forall \quad i \in \{1, 2, \ldots, r\}$.	
SMD14		
	$F_1 = (a_1 - 1)^2 + \sum_{i=1}^{p-1} \left((a_i - 1)^2 + (a_{i+1} - (a_i)^2)^2\right)$,	
	$F_2 = -\sum_{i=1}^{q} \lvert c_i \rvert^{i+1} + \sum_{i=q+1}^{q+s} (c_i)^2$,	
	$F_3 = \sum_{i=1}^{r} i (b_i)^2 - \sum_{i=1}^{r} \lvert d_i \rvert$,	
	$f_1 = \sum_{i=1}^{p} \lfloor a_i \rfloor$,	$a_i = 1 \; \forall \; i$,
	$f_2 = \sum_{i=1}^{q} \lvert c_i \rvert^{i+1} + \sum_{i=q+1, i=i+2}^{q+s-1} (c_{i+1} - c_i)^2$,	$b_i = 0 \; \forall \; i$,
	$f_3 = \sum_{i=1}^{r} \lvert (b_i)^2 - (d_i)^2 \rvert$,	$c_i = 0 \; \forall \; i$,
	$a_i \in [-5, 10], \quad \forall \quad i \in \{1, 2, \ldots, p\}$,	$d_i = 0 \; \forall \; i$.
	$b_i \in [-5, 10], \quad \forall \quad i \in \{1, 2, \ldots, r\}$,	
	$c_i \in [-5, 10], \quad \forall \quad i \in \{1, 2, \ldots, q + s\}$,	
	$d_i \in [-5, 10], \quad \forall \quad i \in \{1, 2, \ldots, r\}$.	