Approximate Bilevel Optimization with Population-based Evolutionary Algorithms

Kalyanmoy Deb, Ankur Sinha, Pekka Malo, and Zhichao Lu

COIN Report Number 2019011
Computational Optimization and Innovation (COIN) Laboratory
Michigan State University, East Lansing, USA
http://www.coin-lab.org

Abstract Population-based optimization algorithms, such as evolutionary algorithms, have enjoyed a lot of attention in the past three decades in solving challenging search and optimization problems. In this chapter, we discuss recent population-based evolutionary algorithms for solving different types of bilevel optimization problems, as they pose numerous challenges to an optimization algorithm. Evolutionary bilevel optimization (EBO) algorithms are gaining attention due to their flexibility, implicit parallelism, and ability to customize for specific problem solving tasks. Starting with surrogate-based single-objective bilevel optimization problems, we discuss how EBO methods are designed for solving multi-objective bilevel problems. They show promise for handling various practicalities associated with bilevel problem solving. The chapter concludes with results on an agro-economic bilevel problem. The chapter also presents a number of challenging single and multi-objective bilevel optimization test problems, which should encourage further development of more efficient bilevel optimization algorithms.

1 Introduction

Bilevel and multi-level optimization problems are omni-present in practice. This is because in many practical problems there is a hierarchy of two or more problems

Kalyanmoy Deb
Michigan State University, East Lansing, MI 48824, USA, e-mail: kdeb@egr.msu.edu

Ankur Sinha
Indian Institute of Management, Ahmedabad, India e-mail: asinha@iima.ac.in

Pekka Malo
Aalto University School of Business, Helsinki, Finland e-mail: pekka.malo@aalto.fi

Zhichao Lu
Michigan State University, East Lansing, MI 48824, USA, e-mail: luzhicha@msu.edu

involving different variables, objectives and constraints, arising mainly from the involvement of different hierarchical stakeholders to the overall problem. Consider an agro-economic problem for which clearly there are at least two sets of stakeholders: policy makers and farmers. Although the overall goal is to maximize food production, minimize cost of cultivation, minimize water usage, minimize environmental impact, maximum sustainable use of the land and others, clearly, the overall solution to the problem involves an optimal design on the following variables which must be settled by using an optimization algorithm: crops to be grown, amount of irrigation water and fertilizers to be used, selling price of crops, fertilizer taxes to be imposed, and others. Constraints associated with the problem are restricted run-off of harmful chemicals, availability of limited budget for agriculture, restricted use of land and other resources, and others. One can attempt to formulate the overall problem as a multi-objective optimization problem involving all the above-mentioned variables, objectives, and constraints agreeable to both stake-holders in a single level. Such a solution procedure has at least two difficulties. First, the number of variables, objectives, and constraints of the resulting problem often becomes large, thereby making the solution of the problem difficult to achieve to any acceptable accuracy. Second, such a single-level process is certainly not followed in practice, simply from an easier managerial point of view. In reality, policy-makers come up with fertilizer taxes and agro-economic regulations so that harmful environmental affects are minimal, crop production is sustainable for generations to come, and revenue generation is adequate for meeting the operational costs. On the other hand, once a set of tax and regulation policies are announced, farmers consider them to decide on the crops to be grown, amount and type of fertilizers to be used, and irrigated water to be utilized to obtain maximum production with minimum cultivation cost. Clearly, policy-makers are at the upper level in this overall agro-economic problem solving task and farmers are at the lower level. However, upper level problem solvers must consider how a solution (tax and regulation policies) must be utilized optimally by the lower level problem solvers (farmers, in this case). Also, it is obvious that the optimal strategy of the lower level problem solvers directly depends on the tax and regulation policies declared by the upper level problem solvers. The two levels of problems are intricately linked in such bilevel problems, but, interestingly, the problem is not symmetric between upper and lower levels; instead the upper level's objectives and constraints control the final optimal solution more than that of the lower level problem.

When both upper and lower level involve a single objective each, the resulting bilevel problem is called a single-objective bilevel problem. The optimal solution in this case is usually a single solution describing both upper and lower level optimal variable values. In many practical problems, each level independently or both levels may involve more than one conflicting objectives. Such problems are termed here as multi-objective bilevel problems. These problems usually have more than one bilevel solution. For implementation purposes, a single bilevel solution must be chosen using the preference information of both upper and lower level problem solvers. Without any coordinated effort by upper level decision makers with lower level decision-

makers, a clear choice of the overall bilevel solution is uncertain, which provides another dimension of difficulty to solve multi-objective bilevel problems.

It is clear from the above discussion that bilevel problems are reality in practice, however their practice is uncommon, simply due to the fact that the nested nature of two optimization problems makes the solution procedure computationally expensive. The solution of such problems calls for flexible yet efficient optimization methods, which can handle different practicalities and are capable of finding approximate optimal solutions in a quick computational manner. Recently, evolutionary optimization methods have been shown to be applicable to such problems because of ease of customization that helps in finding near-optimal solution, mainly due to their population approach, use of a direct optimization approach instead of using any gradient, presence of an implicit parallelism mechanism constituting a parallel search of multiple regions simultaneously, and their ability to find multiple optimal solutions in a single application.

In this chapter, we first provide a brief description of metaheuristics in Section 2 followed by a step-by-step procedure of an evolutionary optimization algorithm as a representative of various approaches belonging to this class of algorithms. Thereafter, in Section 3, we discuss the bilevel formulation and difficulties involved in solving a bilevel optimization problem. Then, we provide the past studies on population-based methods for solving bilevel problems in Section 4. Thereafter, we discuss in detail some of the recent efforts in evolutionary bilevel optimization in Section 5. In Section 6, we present two surrogate-assisted single-objective evolutionary bilevel optimization (EBO) algorithms – BLEAQ and BLEAQ2. Simulation results of these two algorithms are presented next on a number of challenging standard and scalable test problems provided in the Appendix. Bilevel evolutionary multi-objective optimization (BL-EMO) algorithms are described next in Section 7. The ideas of optimistic and pessimistic bilevel Pareto-optimal solution sets are differentiated and an overview of research developments on multi-objective bilevel optimization is provided. Thereafter, in Section 8, a multi-objective agro-economic bilevel problem is described and results using the proposed multi-objective EBO algorithm are presented. Finally, conclusions of this extensive chapter is drawn in Section 9.

2 Metaheuristics Algorithms for Optimization

Heuristics refers to relevant guiding principles, or effective rules, or partial problem information related to the problem class being addressed. When higher level heuristics are utilized in the process of creating new solutions in a search or optimization methodology for solving a larger class of problems, the resulting methodology is called a *metaheuristic*. The use of partial problem information may also be used in metaheuristics to speed up the search process in arriving at a near-optimal solution, but an exact convergence to the optimal solution is usually not guaranteed. For many practical purposes, metaheuristics based optimization algorithms are more

pragmatic approaches [14]. These methods are in rise due to the well-known *no-free-lunch* (NFL) theorem.

Theorem 1 *No single optimization algorithm is most computationally effective for solving all problems.*

Although somewhat intuitive, a proof of a more specific and formal version of the above theorem was provided in [66]. A corollary to the NFL theorem is that for solving a *specific* problem class, there exists a *customized* algorithm which would perform the best; however the same algorithm may not work so well on another problem class.

It is then important to ask a very important question: 'How does one develop a customized optimization algorithm for a specific problem class?'. The search and optimization literature does not provide a ready-made answer to the above question for every possible problem class, but the literature is full of different application problems and the respective developed customized algorithms for solving the problem class. Most of these algorithms use relevant heuristics derived from the description of the problem class.

We argue here that in order to utilize problem heuristics in an optimization algorithm, the basic structure of the algorithm must allow heuristics to be integrated easily. For example, the well-known steepest-descent optimization method cannot be customized much with available heuristics, as the main search must always take place along the negative of the gradient of the objective functions to allow an improvement or non-deterioration in the objective value from one iteration to the next. In this section, we discuss EAs that are population-based methods belonging to the class of metaheuristics. There exist other population-based metaheuristic methods which have also been used for solving bilevel problems, which we do not elaborate here, but provide a brief description below:

- **Differential Evolution (DE) [59]:** DE is a steady-state optimization procedure which uses three population members to create a "mutated" point. It is then compared with the best population member and a recombination of variable exchanges is made to create the final offspring point. An optional selection between offspring and the best population member is used. DE cannot force its population members to stay within specified variable bounds and special operators are needed to handle them as well as other constraints. DE is often considered as a special version of an evolutionary algorithm, described later.
- **Particle Swarm Optimization (PSO) [28, 32]:** PSO is a generational optimization procedure which creates one new offspring point for each parent population member by making a vector operation with parent's previous point, its best point since the start of a run and the best-ever population point. Like DE, PSO cannot also force its population members to stay within specified variable bounds and special operators are needed to handle them and other constraints.
- **Other Metaheuristics [5, 22]:** There exists more than 100 other metaheuristics-based approaches, which use different natural and physical phenomenon. Variable bounds and constraints are often handled using penalty functions or by using special operators.

These algorithms, along with evolutionary optimization algorithms described below, allow flexibility for customizing the solution procedure by enabling any heuristics or rules to be embedded in their operators.

Evolutionary algorithms (EAs) are mostly synonymous to metaheuristics based optimization methods. EAs work with multiple points (called a *population*) at each iteration (called a *generation*). Here are the usual steps of a generational EA:

1: An initial population P_0 of size N (population size) is created, usually at random within the supplied variable bounds. Every population member is evaluated (objective functions and constraints) and a combined *fitness* or a selection function is evaluated for each population member. Set the generation counter $t = 0$.
2: A termination condition is checked. If not satisfied, continue with Step 3, else report the best point and stop.
3: Select better population members of P_t by comparing them using the fitness function and store them in mating pool M_t.
4: Take pairs of points from M_t at a time and recombine them using a *recombination* operator to create one or more new points.
5: Newly created points are then locally perturbed by using a *mutation* operator. The mutated point is then stored in an offspring population Q_t. Steps 4 and 5 are continued until Q_t grows to a size of N.
6: Two populations P_t and Q_t are combined and N best members are saved in the new parent population P_{t+1}. The generation counter is incremented by one and the algorithm moves to Step 2.

The recombination operator is unique in EAs and is responsible for recombining two different population members to create new solutions. The selection and recombination operators applied on a population constitutes an implicitly parallel search, providing their power. In the above generational EA, $NT_{\max}$ is the total number of evaluations, where $T_{\max}$ is the number of generations needed to terminate the algorithm. Besides the above generational EA, steady-state EAs exist, in which the offspring population Q_t consists of a single new mutated point. In the steady-state EA, the number of generations $T_{\max}$ needed to terminate would be more than that needed for a generational EA, but the overall number of solution evaluations needed for the steady-state EA may be less, depending on the problem being solved.

Each of the steps in the above algorithm description – initialization (Step 1), termination condition (Step 2), selection (Step 3), recombination (Step 4), mutation (Step 5) and survival (Step 6) – can be changed or embedded with problem information to make the overall algorithm customized for a problem class.

3 Bilevel Formulation and Challenges

Bilevel optimization problems have two optimization problems staged in a hierarchical manner [54]. The outer problem is often referred to as the upper level problem or the leader's problem, and the inner problem is often referred to as the lower level

problem or the follower's problem. Objective and constraint functions of both levels can be functions of all variables of the problem. However, a part of the variable vector, called the upper level variable vector, remains fixed for the lower level optimization problem. For a given upper level variable vector, an accompanying lower level variable set which is optimal to the lower level optimization problem becomes a candidate feasible solution for the upper level optimization problem, subject to satisfaction of other upper level variable bounds and constraint functions. Thus, in such nested problems, the lower level variable vector depends on the upper level variable vector, thereby causing a strong variable linkage between the two variable vectors. Moreover, the upper level problem is usually sensitive to the quality of lower level optimal solution, which makes solving the lower level optimization problem to a high level of accuracy important.

The main challenge in solving bilevel problems is the computational effort needed in solving nested optimization problems, in which for every upper level variable vector, the lower level optimization problem must be solved to a reasonable accuracy. One silver lining is that depending on the complexities involved in two levels, two different optimization algorithms can be utilized, one for upper level and one for lower level, respectively, instead of using the same optimization method for both levels. However, such a nested approach may also be computationally expensive for large scale problems. All these difficulties provide challenges to optimization algorithms while solving a bilevel optimization problem with a reasonable computational complexity.

There can also be situations where the lower level optimization in a bilevel problem has multiple optimal solutions for a given upper level vector. Therefore, it becomes necessary to define, which solution among the multiple optimal solutions at the lower level should be considered. In such cases, one assumes either of the two positions: the *optimistic* position or the *pessimistic* position. In the optimistic position, the follower is assumed to favorable to the leader and chooses the solution that is best for the leader from the set of multiple lower level optimal solutions. In the pessimistic position, it is assumed that the follower may not be favorable to the leader (in fact, the follower is antagonistic to leader's objectives) and may choose the solution that is worst for the leader from the set of multiple lower level optimal solutions. Intermediate positions are also possible and are more pragmatic, which can be defined with the help of selection functions. Most of the literature on bilevel optimization usually focuses on solving optimistic bilevel problems. A general formulation for the bilevel optimization problem is provided below.

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

$$\min_{\mathbf{x}_u, \mathbf{x}_l} \quad F(\mathbf{x}_u, \mathbf{x}_l), \tag{1}$$

$$\text{subject to} \quad \mathbf{x}_l \in \operatorname*{argmin}_{\mathbf{x}_l} \{ f(\mathbf{x}_u, \mathbf{x}_l) : g_j(\mathbf{x}_u, \mathbf{x}_l) \leq 0, \quad j = 1, \ldots, J \}, \tag{2}$$

$$G_k(\mathbf{x}_u, \mathbf{x}_l) \leq 0, \quad k = 1, \ldots, K, \tag{3}$$

where $G_k : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}$, $k = 1, \ldots, K$ denotes the upper level constraints, and $g_j : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}$, $j = 1, \ldots, J$ represents the lower level constraints, respectively. Variables $\mathbf{x}_u$ and $\mathbf{x}_l$ are n and m dimensional vectors, respectively. It is important to specify the position one is taking while solving the above formulation.

4 Non-Surrogate-based EBO Approaches

A few early EBO algorithms were primarily either nested approaches or used the KKT conditions of the lower level optimization problem to reduce the bilevel problem to a single level and then a standard algorithm was applied. In this section, we provide a review of such approaches. It is important to note that the implicit parallel search of EAs described before still plays a role in constituting an efficient search within both the upper and lower level optimization tasks.

4.1 Nested Methods

Nested EAs are a popular approach to handle bilevel problems, where lower level optimization problem is solved corresponding to each and every upper level member [56, 41, 36]. Though effective, nested strategies are computationally very expensive and not viable for large scale bilevel problems. Nested methods in the area of EAs have been used in primarily two ways. The first approach has been to use an EA at the upper level and a classical algorithm at the lower level, while the second approach has been to utilize EAs at both levels. Of course, the choice between two approaches is determined by the complexity of the lower level optimization problem.

One of the first EAs for solving bilevel optimization problems was proposed in the early 1990s. Mathieu et al. [35] used a nested approach with genetic algorithm at the upper level, and linear programming at the lower level. Another nested approach was proposed in [69], where the upper level was an EA and the lower level was solved using Frank-Wolfe algorithm (reduced gradient method) for every upper level member. The authors demonstrated that the idea can be effectively utilized to solve non-convex bilevel optimization problems. Nested PSO was used in [32] to solve bilevel optimization problems. The effectiveness of the technique was shown on a number of standard test problems with small number of variables, but the computational expense of the nested procedure was not reported. A hybrid approach was proposed in [30], where simplex-based crossover strategy was used at the upper level, and the lower level was solved using one of the classical approaches. The authors report the generations and population sizes required by the algorithm that can be used to compute the upper level function evaluations, but they do not explicitly report the total number of lower level function evaluations, which presumably is high.

DE based approaches have also been used, for instance, in [72], authors used DE at the upper level and relied on the interior point algorithm at the lower level;

similarly, in [2] authors have used DE at both levels. Authors have also combined two different specialized EAs to handle the two levels, for example, in [3] authors use an ant colony optimization to handle the upper level and DE to handle the lower level in a transportation routing problem. Another nested approach utilizing ant colony algorithm for solving a bilevel model for production-distribution planning is [9]. Scatter search algorithms have also been employed for solving production-distribution planning problems, for instance [10].

Through a number of approaches involving EAs at one or both levels, the authors have demonstrated the ability of their methods in solving problems that might otherwise be difficult to handle using classical bilevel approaches. However, as already stated, most of these approaches are practically non-scalable. With increasing number of upper level variables, the number of lower level optimization tasks required to be solved increases exponentially. Moreover, if the lower level optimization problem itself is difficult to solve, numerous instances of such a problem cannot be solved, as required by these methods.

4.2 Single-level Reduction Using Lower Level KKT Conditions

Similar to the studies in the area of classical optimization, many studies in the area of evolutionary computation have also used the KKT conditions of the lower level to reduce the bilevel problem into a single-level problem. Most often, such an approach is able to solve problems that adhere to certain regularity conditions at the lower level because of the requirement of the KKT conditions. However, as the reduced single-level problem is solved with an EA, usually the upper level objective function and constraints can be more general and not adhering to such regularities. For instance, one of the earliest papers using such an approach is by Hejazi et al. [24], who reduced the linear bilevel problem to single-level and then used a genetic algorithm, where chromosomes emulate the vertex points, to solve the problem. Another study [8] also proposed a single level reduction for linear bilevel problems. Wang et al. [63] reduced the bilevel problem into a single-level optimization problem using KKT conditions, and then utilized a constraint handling scheme to successfully solve a number of standard test problems. Their algorithm was able to handle non-differentiability at the upper level objective function, but not elsewhere. Later on, Wang et al. [64] introduced an improved algorithm that was able to handle non-convex lower level problem and performed better than the previous approach [63]. However, the number of function evaluations in both approaches remained quite high (requiring function evaluations to the tune of 100,000 for 2 to 5 variable bilevel problems). In [62], the authors used a simplex-based genetic algorithm to solve linear-quadratic bilevel problems after reducing it to a single level task. Later, Jiang et al. [27] reduced the bilevel optimization problem into a non-linear optimization problem with complementarity constraints, which is sequentially smoothed and solved with a PSO algorithm. Along similar lines of using lower level optimality conditions, Li [29] solved a fractional bilevel optimization problem by utilizing optimality results

of the linear fractional lower level problem. In [60], the authors embed the chaos search technique in PSO to solve single-level reduced problem. The search region represented by the KKT conditions can be highly constrained that poses challenges for any optimization algorithm. To address this concern, in a recent study [58], the authors have used approximate KKT-conditions for the lower level problem. One of the theoretical concerns of using the KKT conditions to replace lower level problem directly is that the associated constraint qualification conditions must also be satisfied for every lower level solution.

5 Surrogate-based EBO Approaches

The earlier approaches suffered with some drawbacks like high computational requirements, or reliance on the KKT conditions of the lower level problem. To overcome this, recent research has focused on surrogate-based methods for bilevel problems. Researchers have used surrogates in different ways for solving bilevel problems that we discuss in this section.

Surrogate-based solution methods are commonly used for optimization problems [61], where actual function evaluations are expensive. A meta-model or surrogate model is an approximation of the actual model that is relatively quicker to evaluate. Based on a small sample from the actual model, a surrogate model can be trained and used subsequently for optimization. Given that, for complex problems, it is hard to approximate the entire model with a small set of sample points, researchers often resort to iterative meta modeling techniques, where the actual model is approximated locally during iterations.

Bilevel optimization problems contain an inherent complexity that leads to a requirement of large number of evaluations to solve the problem. Metamodeling of the lower level optimization problem, when used with population-based algorithms, offers a viable means to handle bilevel optimization problems. With good lower level solutions being supplied at the upper level, EA's implicit parallel search power constructs new and good upper level solutions, making the overall search efficient. In this subsection, we discuss three ways in which metamodeling can be applied to bilevel optimization. There are two important mappings in bilevel optimization, referred to as the rational reaction set and lower level optimal value function. We refer the readers to Figure 1, which provides an understanding of these two mappings graphically for a hypothetical bilevel problem.

5.1 Reaction Set Mapping

One of the approaches to solve bilevel optimization problems using EAs would be through iterative approximation of the reaction set mapping Ψ. The bilevel formulation in terms of the Ψ-mapping can be written as below.

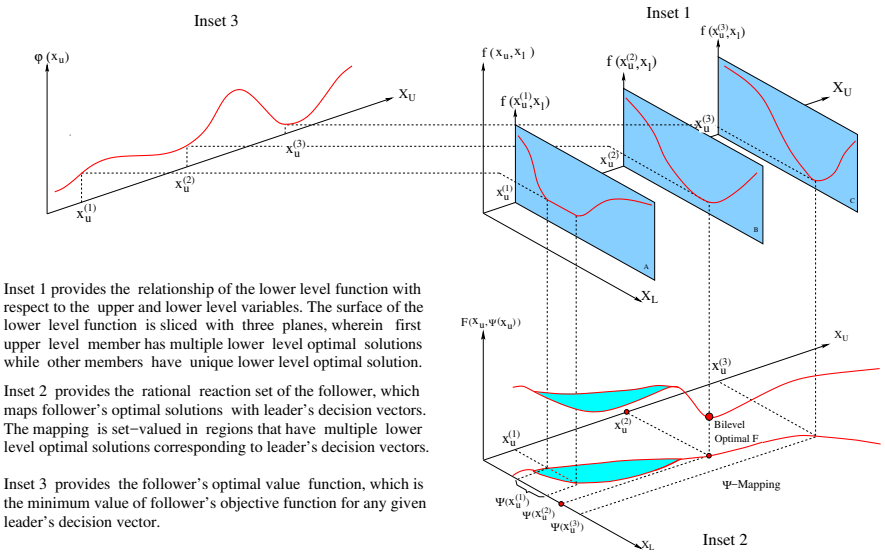

Fig. 1 Graphical representation of rational reaction set (Ψ) and lower level optimal value function (φ).

$$\min_{\mathbf{x}_u, \mathbf{x}_l} \quad F(\mathbf{x}_u, \mathbf{x}_l), \tag{4}$$

$$\text{subject to} \quad \mathbf{x}_l \in \Psi(\mathbf{x}_u), \tag{5}$$

$$G_k(\mathbf{x}_u, \mathbf{x}_l) \leq 0, \quad k = 1, \ldots, K. \tag{6}$$

If the Ψ-mapping in a bilevel optimization problem is known, it effectively reduces the problem to single level optimization. However, this mapping is seldom available; therefore, the approach could be to solve the lower level problem for a few upper level members and then utilize the lower level optimal solutions and corresponding upper level members to generate an approximate mapping $\hat{\Psi}$. It is noteworthy that approximating a set-valued Ψ-mapping offers its own challenges and is not a straightforward task. Assuming that an approximate mapping, $\hat{\Psi}$, can be generated, the following single level optimization problem can be solved for a few generations of the algorithm before deciding to further refine the reaction set.

$$\begin{aligned} \min_{\mathbf{x}_u, \mathbf{x}_l} \quad & F(\mathbf{x}_u, \mathbf{x}_l), \\ \text{subject to} \quad & \mathbf{x}_l \in \hat{\Psi}(\mathbf{x}_u), \\ & G_k(\mathbf{x}_u, \mathbf{x}_l) \leq 0, \quad k = 1, \ldots, K. \end{aligned}$$

EAs that rely on this idea to solve bilevel optimization problems are [51, 48, 47, 4]. In some of these studies, authors have used quadratic approximation to approximate the local reaction set. This helps in saving lower level optimization calls when the approximation for the local reaction set is good. In case the approximations generated

by the algorithm are not acceptable, the method defaults to a nested approach. It is noteworthy that a bilevel algorithm that uses a surrogate model for reaction set mapping may need not be limited to quadratic models but other models can also be used.

5.2 Optimal Lower Level Value Function

Another way to use metamodeling is through the approximation of the optimal value function φ. If the φ-mapping is known, the bilevel problem can once again be reduced to single level optimization problem as follows [68],

$$\begin{aligned} \min_{\mathbf{x}_u, \mathbf{x}_l} \quad & F(\mathbf{x}_u, \mathbf{x}_l), \\ \text{subject to} \quad & f(\mathbf{x}_u, \mathbf{x}_l) \leq \varphi(\mathbf{x}_u), \\ & g_j(\mathbf{x}_u, \mathbf{x}_l) \leq 0, \quad j = 1, \ldots, J, \\ & G_k(\mathbf{x}_u, \mathbf{x}_l) \leq 0, \quad k = 1, \ldots, K. \end{aligned}$$

However, since the value function is seldom known, one can attempt to approximate this function using metamodeling techniques. The optimal value function is a single-valued mapping; therefore, approximating this function avoids the complexities associated with set-valued mapping. As described previously, an approximate mapping $\hat{\varphi}$, can be generated with the population members of an EA and the following modification in the first constraint: $f(\mathbf{x}_u, \mathbf{x}_l) \leq \hat{\varphi}(\mathbf{x}_u)$. Evolutionary optimization approaches that rely on this idea can be found in [43, 46, 52].

5.3 Bypassing Lower Level Problem

Another way to use a meta-model in bilevel optimization would be to completely by-pass the lower level problem, as follows:

$$\begin{aligned} \min_{\mathbf{x}_u} \quad & \hat{F}(\mathbf{x}_u), \\ \text{subject to} \quad & \hat{G}_k(\mathbf{x}_u) \leq 0, \quad k = 1, \ldots, K. \end{aligned}$$

Given that the optimal $\mathbf{x}_l$ are essentially a function of $\mathbf{x}_u$, it is possible to construct a single level approximation of the bilevel problem by ignoring $\mathbf{x}_l$ completely and writing the objective function and constraints for the resulting single level problem as a function of only $\mathbf{x}_u$. However, the landscape for such a single level problem can be highly non-convex, disconnected, and non-differentiable. Advanced metamodeling techniques might be required to use this approach, which may be beneficial for certain classes of bilevel problems. A training set for the metamodel can be constructed by solving a few lower level problems for different $\mathbf{x}_u$. Both upper level objective F and

constraint set (G_k) can then be meta-modeled using $\mathbf{x}_u$ alone. Given the complex structure of such a single-level problem, it might be sensible to create such an approximation locally.

5.4 Limitations and Assumptions of Surrogate-based Approaches

The idea of using function approximations within EAs makes them considerably more powerful than simple random search. However, these algorithms are still constrained by certain background assumptions, and are not applicable to arbitrary problems. In order for the approximations to work, we generally need to impose some constraining assumptions on the bilevel problem to even ensure the existence of the underlying mappings that the algorithms are trying to approximate numerically.

For example, when using function approximations for the reaction set mapping in single objective problems, we need to assume that the lower level problem is convex with respect to the lower level variables. If the lower level problem is not convex, the function could be composed by global as well as local optimal solutions, and if we consider only global optimal solutions in the lower level problem, the respective function could become discontinuous. This challenge can be avoided by imposing the convexity assumption. Furthermore, we generally need to require that the lower level objective function as well as the constraints are twice continuously differentiable. For more insights, we refer to [51], where the assumptions are discussed in greater detail in case of both polyhedral and nonlinear constraints. When these assumptions are not met, there are also no guarantees that the approximation based algorithms can be expected to work any better than a nested search.

In the next section, we describe recent surrogate-based EBO algorithms, but would like to highlight that they are not exempt from the above convexity and differentiability assumptions.

6 Single-objective Surrogate-based EBO Algorithms

In this section, we discuss the working of two surrogate-based EBO algorithms – BLEAQ and BLEAQ2 – that rely on the approximation of the Ψ- and φ-mappings. In a single-objective bilevel problems, both upper and lower level problems have a single objective function: $F(\mathbf{x}_u, \mathbf{x}_l)$ and $f(\mathbf{x}_u, \mathbf{x}_l)$. There can be multiple constraints at each level. We describe these algorithms in the next two subsections.

6.1 Ψ-Approach and BLEAQ Algorithm

In the Ψ-approach, the optimal value of each lower level variable is approximated using a surrogate model of the upper level variable set. Thus, a total of $m = |\mathbf{x}_l|$ number of metamodels must be constructed from a few exact lower level optimization results.

After the lower level problem is optimized for a few upper level variable sets, the optimal lower level variable can be modeled using a quadratic function of upper level variables: $x^*_{l,i} = q_i(\mathbf{x}_u)$. In fact, the algorithm creates various local quadratic models, but for simplicity we skip the details here. Thereafter, upper level objective and constraint functions can be expressed in terms of the upper level variables only in an implicit manner, as follows:

$$\begin{aligned} \min_{\mathbf{x}_u} \quad & F\left(\mathbf{x}_u, \mathbf{q}(\mathbf{x}_u)\right), \\ \text{subject to} \quad & G_k\left(\mathbf{x}_u, \mathbf{q}(\mathbf{x}_u)\right) \leq 0, \quad k = 1, 2, \ldots, K. \end{aligned}$$

Note that $\mathbf{q}(\mathbf{x}_u)$ represents the quadratic approximation for the lower level vector. The dataset for creating the surrogate $\mathbf{q}(\mathbf{x}_u)$, is constructed by identifying the optimal lower level vectors corresponding to various upper level vectors by solving the following problem:

$$\mathbf{x}^*_l = \text{argmin}\left\{ f(\mathbf{x}_u, \mathbf{x}_l) | g_j(\mathbf{x}_u, \mathbf{x}_l) \leq 0, \quad j = 1, 2, \ldots, J \right\}.$$

The validity of the quadratic approximation is checked after a few iterations. If the existing quadratic model does not provide an accurate approximation, new points are introduced to form a new quadratic model. We call this method BLEAQ [51, 48]. Note that this method will suffer whenever the lower level has multiple optimal solutions at the lower level, and also it requires multiple models $q_i(\mathbf{x}_u)$, $\forall\, i \in \{1, \ldots, m\}$, to be created.

6.2 φ-Approach and BLEAQ2 Algorithm

Next, we discuss the φ-approach that overcomes the various drawbacks of the Ψ-approach. In the φ-approach, an approximation of the optimal lower level objective function value $\varphi(\mathbf{x}_u)$ as a function of $\mathbf{x}_u$ is constructed that we refer to as $\hat{\varphi}(\mathbf{x}_u)$. Thereafter, the following single-level problem is solved:

$$\begin{aligned} \text{Minimize } & F\left(\mathbf{x}_u, \mathbf{x}_l\right), \\ \text{subject to } & f(\mathbf{x}_u, \mathbf{x}_l) \leq \hat{\varphi}(\mathbf{x}_u), \\ & G_k\left(\mathbf{x}_u, \mathbf{x}_l\right) \leq 0, \quad k = 1, 2, \ldots, K. \end{aligned} \tag{7}$$

The above formulation allows a single surrogate model to be constructed and the solution of the above problem provides optimal $\mathbf{x}_u$ and $\mathbf{x}_l$ vectors.

In the modified version of BLEAQ, that we called BLEAQ2 [46], both Ψ and φ-approaches are implemented with a check. The model that has higher level of accuracy is used to choose the lower level solution for any given upper level vector. Both Ψ and φ models are approximated locally and updated iteratively.

6.3 Experiments and Results

Next, we assess the performance of three algorithms, the nested approach, BLEAQ, and BLEAQ2 on a set of bilevel test problems. We perform 31 runs for each test instance. For each run, the upper and lower level function evaluations required until termination are recorded separately. Information about the various parameters and their settings can be found in [46, 51].

6.3.1 Results on Non-scalable Test Problems

We first present the empirical results on eight non-scalable test problems selected from the literature (referred to as TP1-TP8). The description for these test problems is provided in the Appendix. Table 1 contains the median upper level (UL) function evaluations, lower level (LL) function evaluations and BLEAQ2'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 the median run with TP1, BLEAQ2 requires 63% less overall function evaluations as compared to BLEAQ, and 98% less overall function evaluations as compared to the nested approach.

Table 1 Median function evaluations on non-scalable test problems TP1 to TP8. While computing savings, we compare the total function evaluations (sum of upper and lower level function evaluations) of one algorithm against the other. Savings for BLEAQ2 when compared against an algorithm A is given as (A – BLEAQ2) /A, where the name of the algorithm denotes the total function evaluations required by the algorithm.

	UL Func. Evals.			LL Func. Evals.			BLEAQ2 Savings	
	BLEAQ2 Med	BLEAQ Med	Nested Med	BLEAQ2 Med	BLEAQ Med	Nested Med	BLEAQ2 vs BLEAQ	BLEAQ2 vs Nested
TP1	136	155	-	242	867	-	63%	Large%
TP2	255	185	436	440	971	5,686	40%	89%
TP3	158	155	633	224	894	6,867	64%	95%
TP4	198	357	1,755	788	1,772	19,764	54%	95%
TP5	272	243	576	967	1,108	6,558	8%	83%
TP6	161	155	144	323	687	1,984	43%	77%
TP7	112	255	193	287	987	2,870	68%	87%
TP8	241	189	403	467	913	7,996	36%	92%

All these test problems are bilevel problems with small number of variables, and all the three algorithms were able to solve the eight test instances successfully. A significant computational saving can be observed for both BLEAQ2 and BLEAQ, as compared to the nested approach, as shown in the 'Savings' column of Table 1. The performance gain going from BLEAQ to BLEAQ2 is quite significant for these test problems even though none of them leads to multiple lower level optimal solutions. Detailed comparison between BLEAQ and BLEAQ2 in terms of upper and lower level function evaluations is provided through Figures 2 and 3, respectively. It can

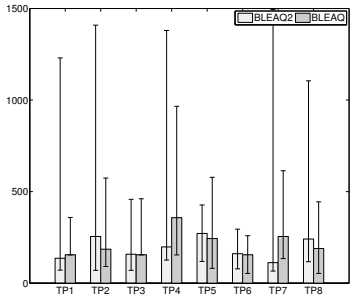

Fig. 2 Variation of upper level function evaluations required by BLEAQ and BLEAQ2 algorithms in 31 runs applied to TP1 to TP8.

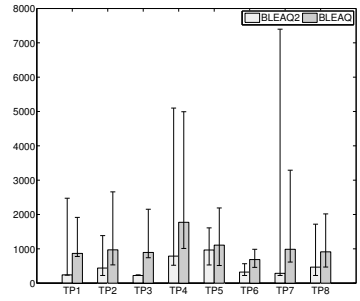

Fig. 3 Variation of lower level function evaluations required by BLEAQ and BLEAQ2 algorithms in 31 runs applied to TP1 to TP8.

be observed that BLEAQ2 requires comparatively much less number of lower level function evaluations than BLEAQ algorithm, while there is no conclusive argument can be made for the number of upper level function evaluations. However, since the lower level problem is solved more often, as shown Table 1, BLEAQ2 requires significantly less number of overall function evaluations to all eight problems.

6.3.2 Results on Scalable Test Problems

Next, we compare the performance of the three algorithms on the scalable SMD test suite (presented in the Appendix) which contains 12 test problems [49]. The test suite was later extend to 14 test problems by adding two additional scalable test problems. First we analyze the performance of the algorithms on five variables, and then we provide the comparison results on 10-variable instances of the SMD test problems. For the five-variable version of the SMD test problems, we use $p = 1$, $q = 2$ and $r = 1$ for all SMD problems except SMD6 and SMD14. For the five-variable version of SMD6 and SMD14, we use $p = 1$, $q = 0$, $r = 1$ and $s = 2$. For the 10-variable version of the SMD test problems, we use $p = 3$, $q = 3$ and $r = 2$ for all SMD problems except SMD6 and SMD14. For the 10-variable version of SMD6 and SMD14, we use $p = 3$, $q = 1$, $r = 2$ and $s = 2$. In their five-variable versions,

there are two variables at the upper level and three variables at the lower level. They also offer a variety of tunable complexities to any algorithm. For instance, the test set contains problems which are multi-modal 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.

Table 2 provides the median function evaluations and overall savings for the three algorithms on all 14 five-variable SMD problems. The table indicates that BLEAQ2

Table 2 Median function evaluations on five-variable SMD test problems.

	UL Func. Evals.			LL Func. Evals.			BLEAQ2 Savings	
	BLEAQ2 Med	BLEAQ Med	Nested Med	BLEAQ2 Med	BLEAQ Med	Nested Med	BLEAQ2 vs BLEAQ	BLEAQ2 vs 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%

is able to solve the entire set of 14 SMD test problems, while BLEAQ fails on two test problems. The overall savings with BLEAQ2 is larger as compared to BLEAQ for all problems. Test problems SMD6 and SMD14 which contain multiple lower level solutions, BLEAQ is unable to handle them. Further details about the required overall function evaluations from 31 runs are provided in Figure 4.

Results for the 10-variable SMD test problems are presented in Table 3. BLEAQ2 leads to much higher savings as compared to BLEAQ. BLEAQ is found to fail again 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 are able to handle these problems. It is noteworthy that SMD9 to SMD14 offer difficulties with multi-modalities and having highly constrained search space, which none of the algorithms are able to handle with the parameter setting used here. Details for the 31 runs on each of these test problems are presented in Figure 5.

The advantage of BLEAQ2 algorithm comes from the use of both Ψ and φ-mapping based surrogate approaches. We pick two SMD problems – SMD1 and SMD13 – to show that one of the two surrogate approaches perform better de-

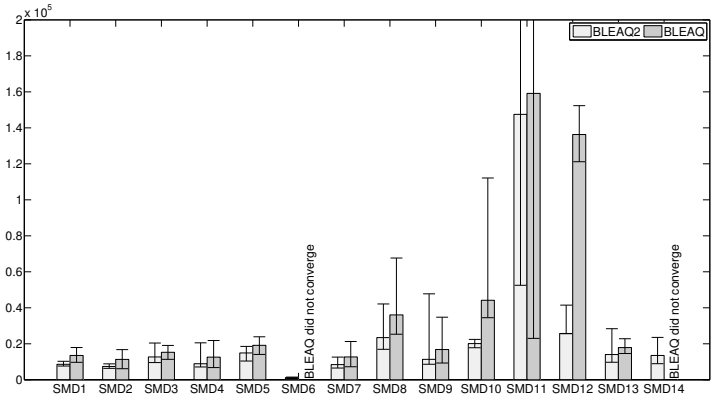

Fig. 4 Overall function evaluations needed by BLEAQ and BLEAQ2 for solving five-dimensional SMD1 to SMD14 problems.

Table 3 Median function evaluations on 10-variable SMD test problems.

	UL Func. Evals.			LL Func. Evals.			BLEAQ2 Savings	
	BLEAQ2 Med	BLEAQ Med	Nested Med	BLEAQ2 Med	BLEAQ Med	Nested Med	BLEAQ2 vs BLEAQ	BLEAQ2 vs 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

pending on their suitability on the function landscape. Figure 6 shows that in SMD1 problem, φ-approximation performs better and Figure 7 shows that Ψ-approximation is better on SMD13. In these figures, the variation of the Euclidean distance of lower level solution from exact optimal solution with generations is shown. While both approaches reduce the distance in a noisy manner, BLEAQ2 does it better by using the best of both approaches than BLEAQ which uses only the Ψ-approximation. The two figures show the adaptive nature of the BLEAQ2 algorithm in choosing the right approximation strategy based on the difficulties involved in a bilevel optimization problem.

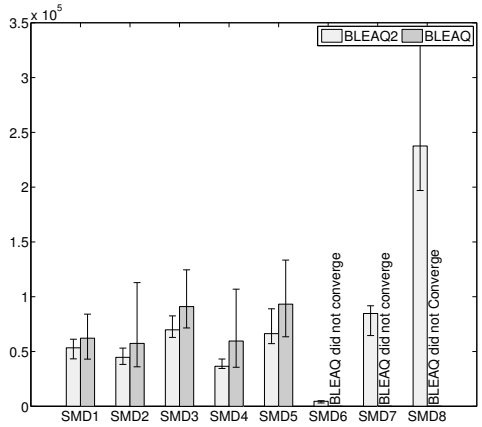

Fig. 5 Overall function evaluations needed by BLEAQ and BLEAQ2 for solving 10-dimensional SMD1 to SMD14 problems.

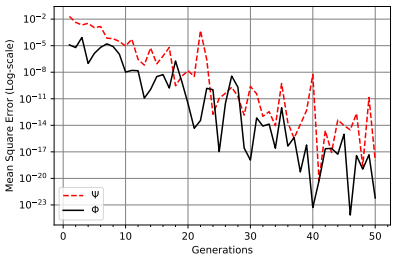

Fig. 6 Approximation error (in terms of Euclidean distance) of a predicted lower level optimal solution when using localized Ψ and φ-mapping during the BLEAQ2 algorithm on the five-variable SMD1 test problem.

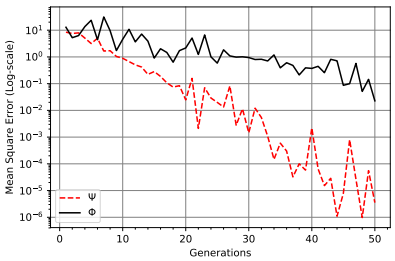

Fig. 7 Approximation error (in terms of Euclidean distance) of a predicted lower level optimal solution when using localized Ψ and φ-mapping during the BLEAQ2 algorithm on the five-variable SMD13 test problem.

6.4 Other Single-objective EBO Studies

Most often, uncertainties arise from unavoidable variations in implementing an optimized solution. Thus, the issue of uncertainty handling is of great practical significance. Material properties, measurement errors, manufacturing tolerances, interdependence of parameters, environmental conditions, etc. are all sources of uncertainties, which, if not considered during the optimization process, may lead to an optimistic solution without any practical relevance. A recent work [34] introduces the concept of robustness and reliability in bilevel optimization problems arising from uncertainties in both lower and upper level decision variables and parameters. The effect of uncertainties on the final robust/reliable bilevel solution was clearly demonstrated in the study using simple, easy-to-understand test problems, followed by a couple of application problems. The topic of uncertainty handling in bilevel

problems is highly practical and timely with the overall growth in research in bilevel methods and in uncertainty handling methods.

Bilevel methods may also be used for achieving an adaptive parameter tuning for optimization algorithms, for instance [57]. The upper level problem considers the algorithmic parameters as variables and the lower level problem uses the actual problem variables. The lower level problem is solved using the algorithmic parameters described by the upper level multiple times and the resulting performance of the algorithm is then used as an objective of the upper level problem. The whole process is able to find optimized parameter values along with the solution of the a number of single-objective test problems.

In certain scenarios, bilevel optimization solution procedures can also be applied to solve single level optimization problems in a more efficient way. For example, the optimal solution of a single-level primal problem can be obtained by solving a dual problem constructed from the Lagranrian function of dual variables. The dual problem formulation is a bilevel (min-max) problem with respect to two sets of variables: upper level involves Lagrangian multipliers or dual variables and lower level involves problem variables or primal variables. A study used evolutionary optimization method to solve the (bilevel) dual problem using a co-evolutionary approach [13]. For zero duality gap problems, the proposed bilevel approach not only finds the optimal solution to the problem, but also produces the Lagrangian multipliers corresponding to the constraints.

7 Multi-objective Bilevel Optimization

Quite often, a decision maker in a practical optimization problem is interested in optimizing multiple conflicting objectives simultaneously. This leads us to the growing literature on multi-objective optimization problem solving [12, 11]. Multiple objectives can also be realized in the context of bilevel problems, where either a leader, or follower, or both might be facing multiple objectives in their own levels [70, 1, 31, 25]. This gives rise to multi-objective bilevel optimization problems that is defined below.

Definition 2 For the upper level objective function $F : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^p$ and lower level objective function $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^q$, the multi-objective bilevel problem is given by

$$\begin{aligned} \min_{\mathbf{x}_u, \mathbf{x}_l} \quad & F(\mathbf{x}_u, \mathbf{x}_l) = (F_1(\mathbf{x}_u, \mathbf{x}_l), \ldots, F_p(\mathbf{x}_u, \mathbf{x}_l)), \\ \text{subject to} \quad & \mathbf{x}_l \in \operatorname*{argmin}_{\mathbf{x}_l} \{ f(\mathbf{x}_u, \mathbf{x}_l) = (f_1(\mathbf{x}_u, \mathbf{x}_l), \ldots, f_q(\mathbf{x}_u, \mathbf{x}_l)) : \\ & \qquad g_j(\mathbf{x}_u, \mathbf{x}_l) \leq 0, \quad j = 1, \ldots, J \}, \\ & G_k(\mathbf{x}_u, \mathbf{x}_l) \leq 0, \quad k = 1, \ldots, K, \end{aligned}$$

where $G_k : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$, $k = 1, \ldots, K$ denotes the upper level constraints, and $g_j : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ represents the lower level constraints, respectively.

Bilevel problems with multiple objectives at lower level are expected to be considerably more complicated than the single-objective case. Surrogate based methods can once again be used, but even verification of the conditions, for instance, when the use of approximations for the reaction set mapping are applicable, is a challenge and requires quite many tools from variational analysis literature. Some of these conditions are discussed in [53]. As remarked in [55], bilevel optimal solution may not exist for all problems. Therefore, additional regularity and compactness conditions are needed to ensure existence of a solution. This is currently an active area of research. Results have been presented by [21], who established necessary optimality conditions for optimistic multi-objective bilevel problems with the help of Hiriart-Urruty scalarization function.

7.1 Optimistic Versus Pessimistic Solutions in Multi-objective Bilevel Optimization

The optimistic or pessimistic position becomes more prominent in multi-objective bilevel optimization. In the presence of multiple objectives at the lower level, the set-valued mapping $\Psi(\mathbf{x}_u)$ normally represents a set of Pareto-optimal solutions corresponding to any given $\mathbf{x}_u$, which we refer as follower's Pareto-optimal frontier. A solution to the overall problem (with optimistic or pessimistic position) is expected to produce a trade-off frontier for the leader that we refer as the leader's Pareto-optimal frontier. In these problems, the lower level problem produces its own Pareto-optimal set and hence the upper level optimal set depends on which solutions from the lower level would be chosen by the lower level decision-makers. The optimistic and pessimistic fronts at the upper level mark the best and worst possible scenarios at the upper level, given that the lower level solutions are always Pareto-optimal.

Though optimistic position have commonly been studied in classical [20] and evolutionary [18] literature in the context of multi-objective bilevel optimization, it is far from realism to expect that the follower will cooperate (knowingly or unknowingly) to an extent that she chooses any point from her Pareto-optimal frontier that is most suitable for the leader. This relies on the assumption that the follower is indifferent to the entire set of optimal solutions, and therefore decides to cooperate. The situation was entirely different in the single-objective case, where, in case of multiple optimal solutions, all the solutions offered an equal value to the follower. However, this can not be assumed in the multi-objective case. Solution to the optimistic formulation in multi-objective bilevel optimization leads to the best possible Pareto-optimal frontier that can be achieved by the leader. Similarly, solution to the pessimistic formulation leads to the worst possible Pareto-optimal frontier at the upper level.

If the value function or the choice function of the follower is known to the leader, it provides an information as to what kind of trade-off is preferred by the follower. A knowledge of such a function effectively, casually speaking, reduces the

lower level optimization problem into a single-objective optimization task, where the value function may be directly optimized. The leader's Pareto-optimal frontier for such intermediate positions lies between the optimistic and the pessimistic frontiers. Figure 8 shows the optimistic and pessimistic frontiers for a hypothetical multi-objective bilevel problem with two objectives at upper and lower levels. Follower's frontier corresponding to $\mathbf{x}_u^{(1)}$, $\mathbf{x}_u^{(2)}$ and $\mathbf{x}_u^{(3)}$, and her decisions A_l, B_l and C_l are shown in the insets. The corresponding representations of the follower's frontier and decisions (A_u, B_u and C_u) in the leader's space are also shown.

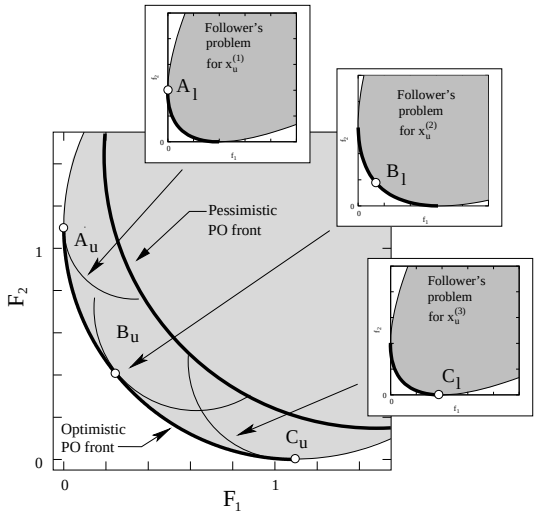

Fig. 8 Leader's Pareto-optimal (PO) frontiers for optimistic and pessimistic positions. Few follower's Pareto-optimal (PO) frontiers are shown (in insets) along with their representations in the leader's objective space. Taken from [54].

7.2 Bilevel Evolutionary Multi-objective Optimization Algorithms

There exists a significant amount of work on single objective bilevel optimization; however, little has been done on bilevel multi-objective optimization primarily because of the computational and decision making complexities that these problems offer. For results on optimality conditions in multi-objective bilevel optimization, the readers may refer to [21, 67]. On the methodology side, Eichfelder [19, 20] solved simple multi-objective bilevel problems using a classical approach. The lower level problems in these studies have been solved using a numerical optimization technique, and the upper level problem is handled using an adaptive exhaustive search method. This makes the solution procedure computationally demanding and non-scalable to large-scale problems. In another study, Shi and Xia [40] used ϵ-constraint method at both levels of multi-objective bilevel problem to convert the problem into an ϵ-constraint bilevel problem. The ϵ-parameter is elicited from the decision maker, and

the problem is solved by replacing the lower level constrained optimization problem with its KKT conditions.

One of the first studies, utilizing an evolutionary approach for multi-objective bilevel optimization was by Yin [69]. The study involved multiple objectives at the upper lever, and a single objective at the lower level. The study suggested a nested genetic algorithm, and applied it on a transportation planning and management problem. Multi-objective linear bilevel programming algorithms were suggested elsewhere [7]. Halter and Mostaghim [23] used a PSO based nested strategy to solve a multi-component chemical system. The lower level problem in their application was linear for which they used a specialized linear multi-objective PSO approach. A hybrid bilevel evolutionary multi-objective optimization algorithm coupled with local search was proposed in [18] (For earlier versions, refer [15, 44, 17, 16]). In the paper, the authors handled non-linear as well as discrete bilevel problems with relatively larger number of variables. The study also provided a suite of test problems for bilevel multi-objective optimization.

There has been some work done on decision making aspects at upper and lower levels. For example, in [42] an optimistic version of multi-objective bilevel optimization, involving interaction with the upper level decision maker, has been solved. The approach leads to the most preferred point at the upper level instead of the entire Pareto-frontier. Since multi-objective bilevel optimization is computationally expensive, such an approach was justified as it led to enormous savings in computational expense. Studies that have considered decision making at the lower level include [50, 55]. In [50], the authors have replaced the lower level with a value function that effectively reduces the lower level problem to single-objective optimization task. In [55], the follower's value function is known with uncertainty, and the authors propose a strategy to handle such problems. Other work related to bilevel multi-objective optimization can be found in [37, 38, 33, 39, 71].

7.3 BL-EMO for Decision-making Uncertainty

In most of the practical applications, a departure from the assumption of an indifferent lower level decision maker is necessary [55]. Instead of giving all decision-making power to the leader, the follower is likely to act according to her own interests and choose the most preferred lower level solution herself. As a result, lower level decision making has a substantial impact on the formulation of multi-objective bilevel optimization problems. First, the lower level problem is not a simple constraint that depends only on lower level objectives. Rather, it is more like a selection function that maps a given upper level decision to a corresponding Pareto-optimal lower level solution that it is most preferred by the follower. Second, while solving the bilevel problem, the upper level decision maker now needs to model the follower's behavior by anticipating her preferences towards different objectives. The following formulation of the problem is adapted from [55].

Definition 3 Let $\xi \in \Xi$ denote a vector of parameters describing the follower's preferences. If the upper level decision maker has complete knowledge of the follower's preferences, the follower's actions can then be modeled via selection mapping

$$\sigma : \mathbf{x}_u \times \Xi \to \mathbf{x}_l, \quad \sigma(\mathbf{x}_u, \xi) \in \Psi(\mathbf{x}_u), \tag{8}$$

where Ψ is the set-valued mapping defined earlier. The resulting bilevel problem can be rewritten as follows:

$$\begin{aligned} \min_{\mathbf{x}_u} \quad & F(\mathbf{x}_u, \mathbf{x}_l) = (F_1(\mathbf{x}_u, \mathbf{x}_l), \ldots, F_p(\mathbf{x}_u, \mathbf{x}_l)) \\ \text{subject to} \quad & \mathbf{x}_l = \sigma(\mathbf{x}_u, \xi) \in \Psi(\mathbf{x}_u) \\ & G_k(\mathbf{x}_u, \mathbf{x}_l) \leq 0, \quad k = 1, \ldots, K \end{aligned}$$

To model the follower's behavior, one approach is to consider the classical value function framework [55]. We can assume that the follower's preferences are characterized by a function $V : \mathbb{R}^q \times \Xi \to \mathbb{R}$ that is parameterized by the preference vector ξ. This allows us to write σ as a selection mapping for a value function optimization problem with $\mathbf{x}_u$ and ξ as parameters:

$$\sigma(\mathbf{x}_u, \xi) \in \operatorname*{argmin}_{\mathbf{x}_l} \{V(f(\mathbf{x}_u, \mathbf{x}_l), \xi) : \ g_j(\mathbf{x}_u, \mathbf{x}_l) \leq 0, \quad j = 1, \ldots, J\}. \tag{9}$$

When the solution is unique, the above inclusion can be treated as an equality that allows considering σ as a solution mapping for the problem. For most purposes, it is sufficient to assume that V is a linear form where ξ acts as a stochastic weight vector for the different lower level objectives:

$$V(f(\mathbf{x}_u, \mathbf{x}_l), \xi) = \sum_{i=1}^{q} f_i(\mathbf{x}_u, \mathbf{x}_l)\xi_i. \tag{10}$$

The use of linear value functions to approximate preferences is generally found to be quite effective and works also in situations where the number of objectives is large.

Usually, we have to assume that the follower's preferences are uncertain, i.e. $\xi \sim \mathcal{D}_\xi$, the value function parameterized by ξ is itself a random mapping. To address such problems, the leader can consider the following two-step approach [55]: (i) First, the leader can use her expectation of follower's preferences to obtain information about the location of the Pareto optimal front by solving the bilevel problem with fixed parameters and value function V. (ii) In the second step, the leader can examine the extent of uncertainty by estimating a confidence region around the Pareto optimal frontier corresponding to the expected value function (POF-EVF). Based on the joint evaluation of the expected solutions and the uncertainty observed at different parts of the frontier, the leader can make a better trade-off between her objectives while being aware of the probability of realizing a desired solution. Given the computational complexity of bilevel problems, carrying out these steps requires careful design. One implementation of such algorithm is discussed in detail in [55].

7.3.1 An Example

Consider an example which has two objectives at both the levels and the problem is scalable in terms of lower level variables; see Table 4. Choosing $m = 14$ will result in a 15 variable bilevel problem with 1 upper level variable and 14 lower level variables. Assume the follower's preferences to follow a linear value function with a bi-variate normal distribution for the weights.

Table 4 Two-objective bilevel example problem.

Example	Level	Formulation
Variables	Upper level	x_u
	Lower level	$\mathbf{x}_l = (x_{l,1}, \ldots, x_{l,m})$
Objectives	Upper level	$F_1(x_u, \mathbf{x}_l) = (x_{l,1} - 1)^2 + \sum_{i=2}^{m} (x_{l,i})^2 + (x_u)^2$
		$F_2(x_u, \mathbf{x}_l) = (x_{l,1} - 1)^2 + \sum_{i=2}^{m} (x_{l,i})^2 + (x_u - 1)^2$
	Lower level	$f_1(x_u, \mathbf{x}_l) = (x_{l,1})^2 + \sum_{i=2}^{m} (x_{l,i})^2$
		$f_2(x_u, \mathbf{x}_l) = \lvert x_u \rvert (x_{l,1} - x_u)^2 + \sum_{i=2}^{m} (x_{l,i})^2$
Constraints	Upper level	$-1 \leq (x_u, x_{l,1}, \ldots, x_{l,K}) \leq 2$
Preference uncertainty	Lower level	$V(f_1, f_2) = \xi_1 f_1 + \xi_2 f_2$,
		$\xi \sim N_2(\mu_\xi, \Sigma_\xi), \quad \mu_\xi = (1, 2), \quad \Sigma_\xi = \begin{bmatrix} 0.01 & 0 \\ 0 & 0.01 \end{bmatrix}$

For any x_u, the Pareto-optimal solutions of the lower level optimization problem are given by

$$\Psi(x_u) = \{\mathbf{x}_l \in \mathbb{R}^m \mid x_{l,1} \in [0, x_u],\ x_{l,i} = 0 \text{ for } i = 2, \ldots, m\}.$$

The best possible frontier at the upper level may be obtained when the lower level decision maker does not have any real decision-making power; see Figure 9.

Now let us consider the example with a lower level decision-maker, whose preferences are assumed to follow $V(f_1, f_2) = \xi_1 f_1 + \xi_2 f_2$. The upper level front corresponding to the expected value function is obtained by solving identifying the Pareto optimal front corresponding to the expected value function (POF-EVF). The outcome is shown in Figure 10, where the follower's influence on the bilevel solution is shown as shift of the expected frontier away from the leader's optimal frontier. The extent of decision uncertainty is described using the bold lines around the POF-EVF front. Each line corresponds to the leader's confidence region $C_\alpha(x_u)$ with $\alpha = 0.01$ at different x_u. When examining the confidence regions at different parts of the frontier, substantial variations can be observed.

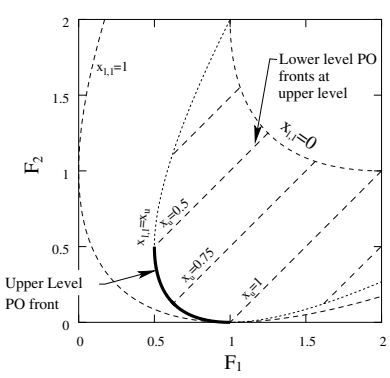

Fig. 9 Upper level Pareto-optimal front (without lower level decision making) and few representative lower level Pareto-optimal fronts in upper level objective space.

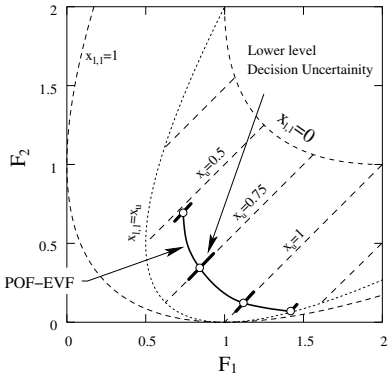

Fig. 10 Expected Pareto-optimal front at upper level and lower level decision uncertainty.

8 Application Studies of EBO

We consider an agri-environmental policy targeting problem for the Raccoon River Watershed, which covers roughly 9,400 km^2 in West-Central Iowa. Agriculture accounts for the majority of land use in the study area, with 75.3% of land in crop land, 16.3% in grassland, 4.4% in forests and just 4% in urban use [26]. The watershed also serves as the main source of drinking water for more than 500,000 people in Central Iowa. However, due to its high concentration of nitrate pollution from intensive fertilizer and livestock manure application, nitrate concentrations routinely exceed Federal limits, with citations dating back to the late 1980s.

Given the above issues, one of the objectives the policy maker faces is to reduce the extent of pollution caused by agricultural activities by controlling the amount of fertilizer used [65]. However, at the same time the policy maker does not intend to hamper agricultural activities to an extent of causing significant economic distress for the farmers.

Consider a producer (follower) $i \in \{1, \ldots, I\}$ trying to maximize her profits from agricultural production through N inputs $\mathbf{x}^i = \{x_1^i, \ldots, x_N^i\}$ and M outputs $\mathbf{y}^i = \{y_1^i, \ldots, y_M^i\}$. Out of the N inputs, x_N^i denotes the nitrogen fertilizer input for each farm. The policy maker must choose the optimal spatial allocation of taxes, $\boldsymbol{\tau} = \{\tau^1, \ldots, \tau^I\}$ for each farm corresponding to the nitrogen fertilizer usage $\mathbf{x}_N = \{x_N^1, \ldots, x_N^I\}$ so as to control the use of fertilizers. Tax vector $\boldsymbol{\tau} = \{\tau^1, \ldots, \tau^I\}$ denotes the tax policy for I producers that is expressed as a multiplier on the total cost of fertilizers. Note that the taxes can be applied as a constant for the entire basin, or they can be spatially targeted at semi-aggregated levels or individually at the farm-level. For generality, in our model we have assumed a different tax policy for each producer. The objectives of the upper level are to to jointly maximize environmental

benefits, $B(\mathbf{x}_N)$ – which consists of the total reduction of non-point source emissions of nitrogen runoff from agricultural land – while also maximizing the total basin's profit $\Pi(\boldsymbol{\tau}, \mathbf{x}_N)$. The optimization problem that the policy maker needs to solve in order to identify a Pareto set of efficient policies is given as follows:

$$\begin{aligned} \max_{\boldsymbol{\tau}, \mathbf{X}} \quad & F(\boldsymbol{\tau}, \mathbf{x}) = \big(\Pi(\boldsymbol{\tau}, \mathbf{x}_N),\ B(\mathbf{x}_N)\big) \\ \text{s.t.} \quad & x_N^i \in \operatorname*{argmax}_{\mathbf{x}^i} \{\pi^i(\tau^i, x_N^i) : (\tau^i,\ \mathbf{x}^i) \in \Omega^i\} \\ & x_n^i \geq 0,\ \forall\ i \in \{1, \ldots, I\},\ n \in \{1, \ldots, N\}, \\ & y_m^i \geq 0,\ \forall\ i \in \{1, \ldots, I\},\ m \in \{1, \ldots, M\}, \\ & \tau^i \geq 1,\ \forall\ i \in \{1, \ldots, I\}, \end{aligned} \tag{11}$$

The fertilizer tax, $\boldsymbol{\tau}$, serves as a multiplier on the total cost of fertilizer, so $\tau^i \geq 1$. The environmental benefit is the negative of pollution and therefore can be written as the negative of the total pollution caused by the producers, i.e. $B(\mathbf{x}_N) = -\sum_{i=1}^{I} p(x_N^i)$. Similarly the total basin profit can be written as the sum total of individual producer's profit, i.e. $\Pi(\boldsymbol{\tau}, \mathbf{x}_N) = \sum_{i=1}^{I} \pi(\tau^i, x_N^i)$. The lower level optimization problem for each agricultural producer can be written as:

$$\begin{aligned} \max_{x_N^i} \quad & \pi^i(\tau^i, x_N^i) = p^i y^i - \sum_{n=1}^{N-1} w_n x_n^i - \tau^i w_N x_N^i, \\ \text{s.t.} \quad & y^i \leq P^k(x^i), \end{aligned} \tag{12}$$

where w and p are the costs and prices of the fertilizer inputs x and crop yields y, respectively. $P^i(x^i)$ denotes the production frontier for producer i. Heterogeneity across producers, due primarily to differences in soil type, may prevent the use of a common production function that would simplify the solution of (11). Likewise, the environmental benefits of reduced fertilizer use vary across producers, due to location and hydrologic processes within the basin. This also makes the solution of (11) more complex.

The simulation results [6] considering all 1,175 farms in the Raccoon River Watershed are shown in Figure 11. The Pareto-optimal frontiers trading off the environmental footprints and economic benefits are compared among the three algorithms.

9 Conclusions

Bilevel optimization problems are omnipresent in practice. However, these problems are often posed as single-level optimization problems due to the existence of many single-level optimization algorithms. While this book presents various theoretical

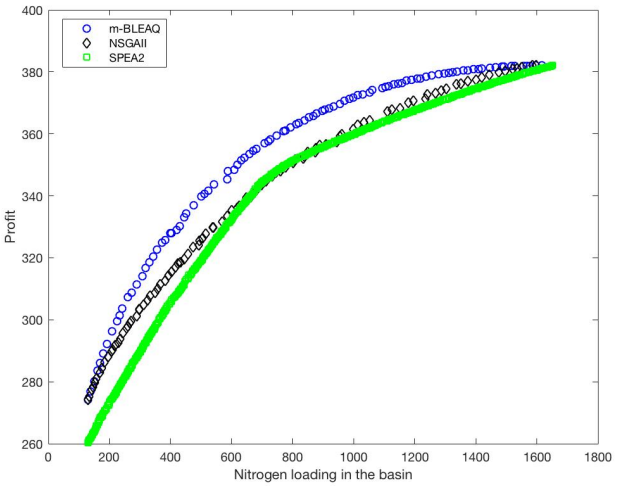

Fig. 11 The obtained Pareto-optimal frontiers from various methods. The lower level reaction set mapping is analytically defined in this problem. We directly supply the lower level optimal solutions in case of single-level optimization algorithm like NSGA-II and SPEA2 and compare the performance. Taken from [6].

and practical aspects of bilevel optimization, this chapter presents a few viable EAs for solving bilevel problems.

Bilevel problems involve two optimization problems which are nested in nature. Hence, they are computationally expensive to solve to optimality. In this chapter, we have discussed surrogate modeling based approaches for approximating the lower level optimum by a surrogate to speed up the overall computations. Moreover, we have presented multi-objective bilevel algorithms, lending to a number of research and application opportunities. We have also provided a set of systematic, scalable, challenging, single-objective and multi-objective unconstrained and constrained bilevel problems for the bilevel optimization community to develop more efficient algorithms.

Research on bilevel optimization by the evolutionary computation researchers has received a lukewarm interest so far, but hopefully this chapter has provided an overview of some of the existing efforts in the area of evolutionary computation toward solving bilevel optimization problems.

Acknowledgements Some parts of this chapter are adapted from authors' published papers on the topic. Further details can be obtained from the original studies, referenced at the end of this chapter.

Appendix: Bilevel Test Problems

Non-scalable Single-Objective Test Problems from Literature

In this section, we provide some of the standard bilevel test problems used in the bilevel studies. Most of these test problems involve only a few fixed number of variables at both upper and lower levels. The test problems (TPs) involve a single objective function at each level. Formulation of the problems are provided in Table 5.

Table 5: Standard test problems TP1-TP8. Note that $\mathbf{x}_u = \mathbf{x}$ and $\mathbf{x}_l = \mathbf{y}$.

Problem	Formulation	Best Known Sol.
TP1		
$n = 2$, $m = 2$	$\underset{(\mathbf{x},\mathbf{y})}{\text{Minimize}}\ F(\mathbf{x},\mathbf{y}) = (x_1-30)^2 + (x_2-20)^2 - 20y_1 + 20y_2,$ s.t. $\mathbf{y} \in \underset{(\mathbf{y})}{\text{argmin}}\ \{f(\mathbf{x},\mathbf{y}) = (x_1-y_1)^2 + (x_2-y_2)^2 :$ $0 \le y_i \le 10, \quad i = 1,2\},$ $x_1 + 2x_2 \ge 30,\ x_1 + x_2 \le 25,\ x_2 \le 15.$	$F = 225.0$ $f = 100.0$
TP2		
$n = 2$, $m = 2$	$\underset{(\mathbf{x},\mathbf{y})}{\text{Minimize}}\ F(\mathbf{x},\mathbf{y}) = 2x_1 + 2x_2 - 3y_1 - 3y_2 - 60,$ s.t. $\mathbf{y} \in \underset{(\mathbf{y})}{\text{argmin}}\ \{f(\mathbf{x},\mathbf{y}) = (y_1-x_1+20)^2 + (y_2-x_2+20)^2 :$ $x_1 - 2y_1 \ge 10,\ x_2 - 2y_2 \ge 10,$ $-10 \ge y_i \ge 20, \quad i = 1,2\},$ $x_1 + x_2 + y_1 - 2y_2 \le 40,$ $0 \le x_i \le 50, \quad i = 1,2.$	$F = 0.0$ $f = 100.0$
TP3		
$n = 2$, $m = 2$	$\underset{(\mathbf{x},\mathbf{y})}{\text{Minimize}}\ F(\mathbf{x},\mathbf{y}) = -(x_1)^2 - 3(x_2)^2 - 4y_1 + (y_2)^2,$ s.t. $\mathbf{y} \in \underset{(\mathbf{y})}{\text{argmin}}\ \{f(\mathbf{x},\mathbf{y}) = 2(x_1)^2 + (y_1)^2 - 5y_2 :$ $(x_1)^2 - 2x_1 + (x_2)^2 - 2y_1 + y_2 \ge -3,$ $x_2 + 3y_1 - 4y_2 \ge 4,$ $0 \le y_i, \quad i = 1,2\},$ $(x_1)^2 + 2x_2 \le 4,$ $0 \le x_i, \quad i = 1,2.$	$F = -18.6787$ $f = -1.0156$

Continued on next page

Problem	Formulation	Best Known Sol.
TP4		
	$\underset{(\mathbf{x},\mathbf{y})}{\text{Minimize}}\ F(\mathbf{x},\mathbf{y}) = -8x_1 - 4x_2 + 4y_1 - 40y_2 - 4y_3,$	
	s.t.	
	$\mathbf{y} \in \underset{(\mathbf{y})}{\text{argmin}}\ \{f(\mathbf{x},\mathbf{y}) = x_1 + 2x_2 + y_1 + y_2 + 2y_3 :$	
$n = 2,$	$y_2 + y_3 - y_1 \leq 1,$	
$m = 3$	$2x_1 - y_1 + 2y_2 - 0.5y_3 \leq 1,$	
	$2x_2 + 2y_1 - y_2 - 0.5y_3 \leq 1,$	
	$0 \leq y_i, \quad i = 1, 2, 3\},$	$F = -29.2$
	$0 \leq x_i, \quad i = 1, 2.$	$f = 3.2$
TP5		
	$\underset{(\mathbf{x},\mathbf{y})}{\text{Minimize}}\ F(\mathbf{x},\mathbf{y}) = rt(x)x - 3y_1 - 4y_2 + 0.5t(y)y,$	
	s.t.	
	$\mathbf{y} \in \underset{(\mathbf{y})}{\text{argmin}}\ \{f(\mathbf{x},\mathbf{y}) = 0.5t(y)hy - t(b(x))y :$	
	$-0.333y_1 + y_2 - 2 \leq 0,$	
$n = 2,$	$y_1 - 0.333y_2 - 2 \leq 0,$	
$m = 2$	$0 \leq y_i, \quad i = 1, 2\},$	
	where	
	$h = \begin{pmatrix} 1 & 3 \\ 3 & 10 \end{pmatrix}, b(x) = \begin{pmatrix} -1 & 2 \\ 3 & -3 \end{pmatrix} x, r = 0.1,$	$F = -3.6$
	$t(\cdot)$ denotes transpose of a vector.	$f = -2.0$
TP6		
	$\underset{(\mathbf{x},\mathbf{y})}{\text{Minimize}}\ F(\mathbf{x},\mathbf{y}) = (x_1 - 1)^2 + 2y_1 - 2x_1,$	
	s.t.	
	$\mathbf{y} \in \underset{(\mathbf{y})}{\text{argmin}}\ \{f(\mathbf{x},\mathbf{y}) = (2y_1 - 4)^2 + (2y_2 - 1)^2 + x_1y_1 :$	
	$4x_1 + 5y_1 + 4y_2 \leq 12,$	
$n = 1,$	$4y_2 - 4x_1 - 5y_1 \leq -4,$	
$m = 2$	$4x_1 - 4y_1 + 5y_2 \leq 4,$	
	$4y_1 - 4x_1 + 5y_2 \leq 4,$	
	$0 \leq y_i, \quad i = 1, 2\},$	$F = -1.2091$
	$0 \leq x_1.$	$f = 7.6145$
TP7		
	$\underset{(\mathbf{x},\mathbf{y})}{\text{Minimize}}\ F(\mathbf{x},\mathbf{y}) = -\frac{(x_1+y_1)(x_2+y_2)}{1+x_1y_1+x_2y_2},$	
	s.t.	
	$\mathbf{y} \in \underset{(\mathbf{y})}{\text{argmin}}\ \{f(\mathbf{x},\mathbf{y}) = \frac{(x_1+y_1)(x_2+y_2)}{1+x_1y_1+x_2y_2} :$	
$n = 2,$	$0 \leq y_i \leq x_i, \quad i = 1, 2\},$	
$m = 2$	$(x_1)^2 + (x_2)^2 \leq 100,$	
	$x_1 - x_2 \leq 0,$	$F = -1.96$
	$0 \leq x_i, \quad i = 1, 2.$	$f = 1.96$

Continued on next page

Problem	Formulation	Best Known Sol.
TP8		
	$\underset{(\mathbf{x},\mathbf{y})}{\text{Minimize}}\ F(\mathbf{x},\mathbf{y}) = \lvert 2x_1 + 2x_2 - 3y_1 - 3y_2 - 60\rvert,$	
	s.t.	
	$\mathbf{y} \in \underset{(\mathbf{y})}{\text{argmin}}\ \{f(\mathbf{x},\mathbf{y}) = (y_1 - x_1 + 20)^2 + (y_2 - x_2 + 20)^2 :$	
$n = 2$,	$2y_1 - x_1 + 10 \le 0,$	
$m = 2$	$2y_2 - x_2 + 10 \le 0,$	
	$-10 \le y_i \le 20, \quad i = 1, 2\},$	
	$x_1 + x_2 + y_1 - 2y_2 \le 40,$	$F = 0.0$
	$0 \le x_i \le 50, \quad i = 1, 2.$	$f = 100.0$

Scalable Single-Objective Bilevel Test Problems

Sinha-Malo-Deb (SMD) test problems [49] are a set of 14 scalable single-objective bilevel test problems that offer a variety of controllable difficulties to an algorithm. The SMD test problem suite was originally proposed with eight unconstrained and four constrained problems [49], it was later extended with two additional unconstrained test problems (SMD13 and SMD14) in [45]. 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 $\mathbf{x}_u = \mathbf{x}$ and $\mathbf{x}_l = \mathbf{y}$ are further divided into two sub-vectors. The φ-mapping is defined by the function f_1. Formulation of SMD test problems are provided in Table 6.

$$\begin{aligned} F(\mathbf{x},\mathbf{y}) &= F_1(\mathbf{x}^1) + F_2(\mathbf{y}^1) + F_3(\mathbf{x}^2,\mathbf{y}^2), \\ f(\mathbf{x},\mathbf{y}) &= f_1(\mathbf{x}^1,\mathbf{x}^2) + f_2(\mathbf{y}^1) + f_3(\mathbf{x}^2,\mathbf{y}^2), \end{aligned} \tag{13}$$

where, $\mathbf{x} = (\mathbf{x}^1, \mathbf{x}^2)$ and $\mathbf{y} = (\mathbf{y}^1, \mathbf{y}^2)$.

Table 6: SMD Test Problems 1-14. Note that $(\mathbf{x}^1, \mathbf{x}^2) = (\mathbf{a}, \mathbf{b})$ and $(\mathbf{y}^1, \mathbf{y}^2) = (\mathbf{c}, \mathbf{d})$.

Formulation	Variable Bounds
SMD1:	
$F_1 = \sum_{i=1}^{p} (a_i)^2,$	
$F_2 = \sum_{i=1}^{q} (c_i)^2,$	$a_i \in [-5, 10], \ \forall\, i \in \{1, 2, \ldots, p\},$
$F_3 = \sum_{i=1}^{r} (b_i)^2 + \sum_{i=1}^{r} (b_i - \tan d_i)^2,$	$b_i \in [-5, 10], \ \forall\, i \in \{1, 2, \ldots, r\},$
$f_1 = \sum_{i=1}^{p} (a_i)^2,$	$c_i \in [-5, 10], \ \forall\, i \in \{1, 2, \ldots, q\},$
$f_2 = \sum_{i=1}^{q} (c_i)^2,$	$d_i \in (\frac{-\pi}{2}, \frac{\pi}{2}), \ \forall\, i \in \{1, 2, \ldots, r\}.$
$f_3 = \sum_{i=1}^{r} (b_i - \tan d_i)^2.$	

Continued on next page

Formulation	Variable Bounds
SMD2:	
$F_1 = \sum_{i=1}^{p}(a_i)^2$,	
$F_2 = -\sum_{i=1}^{q}(c_i)^2$,	$a_i \in [-5, 10],\ \forall\, i \in \{1, 2, \ldots, p\}$,
$F_3 = \sum_{i=1}^{r}(b_i)^2 - \sum_{i=1}^{r}(b_i - \log d_i)^2$,	$b_i \in [-5, 1],\ \forall\, i \in \{1, 2, \ldots, r\}$,
$f_1 = \sum_{i=1}^{p}(a_i)^2$,	$c_i \in [-5, 10],\ \forall\, i \in \{1, 2, \ldots, q\}$,
$f_2 = \sum_{i=1}^{q}(c_i)^2$,	$d_i \in (0, e],\ \forall\, i \in \{1, 2, \ldots, r\}$.
$f_3 = \sum_{i=1}^{r}(b_i - \log d_i)^2$.	
SMD3:	
$F_1 = \sum_{i=1}^{p}(a_i)^2$,	
$F_2 = \sum_{i=1}^{q}(c_i)^2$,	$a_i \in [-5, 10],\ \forall\, i \in \{1, 2, \ldots, p\}$,
$F_3 = \sum_{i=1}^{r}(b_i)^2 + \sum_{i=1}^{r}((b_i)^2 - \tan d_i)^2$,	$b_i \in [-5, 10],\ \forall\, i \in \{1, 2, \ldots, r\}$,
$f_1 = \sum_{i=1}^{p}(a_i)^2$,	$c_i \in [-5, 10],\ \forall\, i \in \{1, 2, \ldots, q\}$,
$f_2 = q + \sum_{i=1}^{q}((c_i)^2 - \cos 2\pi c_i)$,	$d_i \in (\frac{-\pi}{2}, \frac{\pi}{2}),\ \forall\, i \in \{1, 2, \ldots, r\}$.
$f_3 = \sum_{i=1}^{r}((b_i)^2 - \tan d_i)^2$.	
SMD4:	
$F_1 = \sum_{i=1}^{p}(a_i)^2$,	
$F_2 = -\sum_{i=1}^{q}(c_i)^2$,	$a_i \in [-5, 10],\ \forall\, i \in \{1, 2, \ldots, p\}$,
$F_3 = \sum_{i=1}^{r}(b_i)^2 - \sum_{i=1}^{r}(\lvert b_i\rvert - \log(1 + d_i))^2$,	$b_i \in [-1, 1],\ \forall\, i \in \{1, 2, \ldots, r\}$,
$f_1 = \sum_{i=1}^{p}(a_i)^2$,	$c_i \in [-5, 10],\ \forall\, i \in \{1, 2, \ldots, q\}$,
$f_2 = q + \sum_{i=1}^{q}((c_i)^2 - \cos 2\pi c_i)$,	$d_i \in [0, e],\ \forall\, i \in \{1, 2, \ldots, r\}$.
$f_3 = \sum_{i=1}^{r}(\lvert b_i\rvert - \log(1 + d_i))^2$.	
SMD5:	
$F_1 = \sum_{i=1}^{p}(a_i)^2$,	
$F_2 = -\sum_{i=1}^{q}((c_{i+1} - c_i^2) + (c_i - 1)^2)$,	$a_i \in [-5, 10],\ \forall\, i \in \{1, 2, \ldots, p\}$,
$F_3 = \sum_{i=1}^{r} b_i^2 - \sum_{i=1}^{r}(\lvert b_i\rvert - d_i^2)^2$,	$b_i \in [-5, 10],\ \forall\, i \in \{1, 2, \ldots, r\}$,
$f_1 = \sum_{i=1}^{p}(a_i)^2$,	$c_i \in [-5, 10],\ \forall\, i \in \{1, 2, \ldots, q\}$,
$f_2 = \sum_{i=1}^{q}((c_{i+1} - c_i^2) + (c_i - 1)^2)$,	$d_i \in [-5, 10],\ \forall\, i \in \{1, 2, \ldots, r\}$.
$f_3 = \sum_{i=1}^{r}(\lvert b_i\rvert - d_i^2)^2$.	
SMD6:	
$F_1 = \sum_{i=1}^{p}(a_i)^2$,	
$F_2 = -\sum_{i=1}^{q} c_i^2 + \sum_{i=q+1}^{q+s} c_i^2$,	$a_i \in [-5, 10],\ \forall\, i \in \{1, 2, \ldots, p\}$,
$F_3 = \sum_{i=1}^{r} b_i^2 - \sum_{i=1}^{r}(b_i - d_i)^2$,	$b_i \in [-5, 10],\ \forall\, i \in \{1, 2, \ldots, r\}$,
$f_1 = \sum_{i=1}^{p}(a_i)^2$,	$c_i \in [-5, 10],\ \forall\, i \in \{1, 2, \ldots, q + s\}$,
$f_2 = \sum_{i=1}^{q} c_i^2 + \sum_{i=q+1, i=i+2}^{q+s-1}(c_{i+1} - c_i)^2$,	$d_i \in [-5, 10],\ \forall\, i \in \{1, 2, \ldots, r\}$.
$f_3 = \sum_{i=1}^{r}(b_i - d_i)^2$.	
SMD7:	
$F_1 = 1 + \frac{1}{400}\sum_{i=1}^{p}(a_i)^2 - \prod_{i=1}^{p}(\cos \frac{a_i}{\sqrt{i}})$,	
$F_2 = -\sum_{i=1}^{q} c_i^2$,	$a_i \in [-5, 10],\ \forall\, i \in \{1, 2, \ldots, p\}$,
$F_3 = \sum_{i=1}^{r} b_i^2 - \sum_{i=1}^{r}(b_i - \log d_i)^2$,	$b_i \in [-5, 10],\ \forall\, i \in \{1, 2, \ldots, r\}$,
$f_1 = \sum_{i=1}^{p} a_i^3$,	$c_i \in [-5, 10],\ \forall\, i \in \{1, 2, \ldots, q\}$,
$f_2 = \sum_{i=1}^{q} c_i^2$,	$d_i \in (0, e],\ \forall\, i \in \{1, 2, \ldots, r\}$.
$f_3 = \sum_{i=1}^{r}(b_i - \log d_i)^2$.	

Continued on next page

Formulation	Variable Bounds
SMD8:	
$F_1 = 20 + e - 20\exp\Big(-0.2\sqrt{\frac{1}{p}\sum_{i=1}^{p}(a_i)^2}\Big)$	
$\quad - \exp\Big(\frac{1}{p}\sum_{i=1}^{p}\cos 2\pi a_i\Big)$,	$a_i \in [-5, 10],\ \forall\, i \in \{1, 2, \ldots, p\}$,
$F_2 = -\sum_{i=1}^{q}((c_{i+1} - c_i^2) + (c_i - 1)^2)$,	$b_i \in [-5, 10],\ \forall\, i \in \{1, 2, \ldots, r\}$,
$F_3 = \sum_{i=1}^{r} b_i^2 - \sum_{i=1}^{r}(b_i - d_i^3)^2$,	$c_i \in [-5, 10],\ \forall\, i \in \{1, 2, \ldots, q\}$,
$f_1 = \sum_{i=1}^{p} \lvert a_i \rvert$,	$d_i \in [-5, 10],\ \forall\, i \in \{1, 2, \ldots, r\}$.
$f_2 = \sum_{i=1}^{q}((c_{i+1} - c_i^2) + (c_i - 1)^2)$,	
$f_3 = \sum_{i=1}^{r}(b_i - d_i^3)^2$.	
SMD9:	
$F_1 = \sum_{i=1}^{p}(a_i)^2$	
$F_2 = -\sum_{i=1}^{q}(c_i)^2$,	
$F_3 = \sum_{i=1}^{r} b_i^2 - \sum_{i=1}^{r}(b_i - \log(1 + d_i))^2$,	$a_i \in [-5, 10],\ \forall\, i \in \{1, 2, \ldots, p\}$,
$f_1 = \sum_{i=1}^{p} a_i^2$,	$b_i \in [-5, 1],\ \forall\, i \in \{1, 2, \ldots, r\}$,
$f_2 = \sum_{i=1}^{q} c_i^2$,	$c_i \in [-5, 10],\ \forall\, i \in \{1, 2, \ldots, q\}$,
$f_3 = \sum_{i=1}^{r}(b_i - \log(1 + d_i))^2$.	$d_i \in (-1, -1 + e],\ \forall\, i \in \{1, 2, \ldots, r\}$.
$G_1 : \sum_{i=1}^{p} a_i^2 + \sum_{i=1}^{r} b_i^2 - \Big\lfloor \sum_{i=1}^{p} a_i^2 + \sum_{i=1}^{r} b_i^2 + 0.5 \Big\rfloor \geq 0$	
$g_1 : \sum_{i=1}^{p} c_i^2 + \sum_{i=1}^{r} d_i^2 - \Big\lfloor \sum_{i=1}^{p} c_i^2 + \sum_{i=1}^{r} d_i^2 + 0.5 \Big\rfloor \geq 0$	
SMD10:	
$F_1 = \sum_{i=1}^{p}(a_i - 2)^2$	
$F_2 = -\sum_{i=1}^{q} c_i^2$,	
$F_3 = \sum_{i=1}^{r}(b_i^2 - 2)^2 - \sum_{i=1}^{r}(b_i - \tan d_i)^2$,	$a_i \in [-5, 10],\ \forall\, i \in \{1, 2, \ldots, p\}$,
$f_1 = \sum_{i=1}^{p} a_i^2$,	$b_i \in [-5, 10],\ \forall\, i \in \{1, 2, \ldots, r\}$,
$f_2 = \sum_{i=1}^{q}(c_i - 2)^2$,	$c_i \in [-5, 10],\ \forall\, i \in \{1, 2, \ldots, q\}$,
$f_3 = \sum_{i=1}^{r}(b_i - \tan d_i)^2$.	$d_i \in (\frac{-\pi}{2}, \frac{\pi}{2}),\ \forall\, i \in \{1, 2, \ldots, r\}$.
$G_j : a_j - \sum_{i=1, i\neq j}^{p} a_i^3 - \sum_{i=1}^{r} b_i^3 \geq 0, \forall j \in \{1, 2, \ldots, p\}$	
$G_{p+j} : b_j - \sum_{i=1, i\neq j}^{p} b_i^3 - \sum_{i=1}^{r} b_i^3 \geq 0, \forall j \in \{1, 2, \ldots, r\}$	
$g_j : c_j - \sum_{i=1, i\neq j}^{q} c_i^3 \geq 0, \forall j \in \{1, 2, \ldots, q\}$	
SMD11:	
$F_1 = \sum_{i=1}^{p} a_i^2$	
$F_2 = -\sum_{i=1}^{q} c_i^2$,	
$F_3 = \sum_{i=1}^{r} b_i^2 - \sum_{i=1}^{r}(b_i - \log d_i)^2$,	$a_i \in [-5, 10],\ \forall\, i \in \{1, 2, \ldots, p\}$,
$f_1 = \sum_{i=1}^{p} a_i^2$,	$b_i \in [-1, 1],\ \forall\, i \in \{1, 2, \ldots, r\}$,
$f_2 = \sum_{i=1}^{q} c_i^2$,	$c_i \in [-5, 10],\ \forall\, i \in \{1, 2, \ldots, q\}$,
$f_3 = \sum_{i=1}^{r}(b_i - \log d_i)^2$.	$d_i \in (\frac{1}{e}, e),\ \forall\, i \in \{1, 2, \ldots, r\}$.
$G_j : b_j \geq \frac{1}{\sqrt{r}} + \log d_j, \forall j \in \{1, 2, \ldots, r\}$	
$g_j : \sum_{i=1}^{r}(b_i - \log d_i) \geq 1$	

Continued on next page

Formulation	Variable Bounds
SMD12:	
$F_1 = \sum_{i=1}^{p}(a_i - 2)^2$	
$F_2 = \sum_{i=1}^{q} c_i^2$,	
$F_3 = \sum_{i=1}^{r}(b_i^2 - 2)^2 + \sum_{i=1}^{r} \tan\lvert d_i\rvert - \sum_{i=1}^{r}(b_i - \tan d_i)^2$,	
$f_1 = \sum_{i=1}^{p} a_i^2$,	$a_i \in [-5, 10],\ \forall\, i \in \{1, 2, \ldots, p\}$,
$f_2 = \sum_{i=1}^{q}(c_i - 2)^2$,	$b_i \in [-14.1, 14.1],\ \forall\, i \in \{1, 2, \ldots, r\}$,
$f_3 = \sum_{i=1}^{r}(b_i - \tan d_i)^2$.	$c_i \in [-5, 10],\ \forall\, i \in \{1, 2, \ldots, q\}$,
$G_1 : b_i - \tan d_i \geq 0, \forall i \in \{1, 2, \ldots, r\}$	$d_i \in (-1.5, 1.5),\ \forall\, i \in \{1, 2, \ldots, r\}$.
$G_2 : a_i - \sum_{i=1, i\neq j}^{p} a_i^3 - \sum_{i=1}^{r} b_i^3 \geq 0, \forall j \in \{1, 2, \ldots, p\}$	
$G_3 : b_i - \sum_{i=1, i\neq j}^{r} b_i^3 - \sum_{i=1}^{p} a_i^3 \geq 0, \forall j \in \{1, 2, \ldots, r\}$	
$g_1 : \sum_{i=1}^{r}(b_i - \tan d_i)^2 \geq 1$	
$g_2 : c_j - \sum_{i=1, i\neq j}^{p} c_i^3, \forall j \in \{1, 2, \ldots, q\}$	
SMD13:	
$F_1 = \sum_{i=1}^{p-1}(a_i - 1)^2 + (a_{i+1} - (a_i)^2)^2$,	
$F_2 = -\sum_{i=1}^{q}\sum_{j=1}^{i}(c_j)^2$,	$a_i \in [-5, 10],\ \forall\, i \in \{1, 2, \ldots, p\}$,
$F_3 = \sum_{i=1}^{r}\sum_{j=1}^{i}(b_j)^2 - \sum_{i=1}^{r}(b_i - \log d_i)^2$,	$b_i \in [-5, e],\ \forall\, i \in \{1, 2, \ldots, r\}$,
$f_1 = \sum_{i=1}^{p}(\lvert a_i\rvert + 2\lvert\sin(a_i)\rvert)$,	$c_i \in [-5, 10],\ \forall\, i \in \{1, 2, \ldots, q\}$,
$f_2 = \sum_{i=1}^{q}\sum_{j=1}^{i}(c_j)^2$,	$d_i \in (0, 10],\ \forall\, i \in \{1, 2, \ldots, r\}$.
$f_3 = \sum_{i=1}^{r}(b_i - \log d_i)^2$.	
SMD14:	
$F_1 = \sum_{i=1}^{p-1}(a_i - 1)^2 + (a_{i+1} - (a_i)^2)^2$,	
$F_2 = -\sum_{i=1}^{q}\lvert c_i\rvert^{i+1} + \sum_{i=q+1}^{q+s}(c_i)^2$,	$a_i \in [-5, 10],\ \forall\, i \in \{1, 2, \ldots, p\}$,
$F_3 = \sum_{i=1}^{r} i(b_i)^2 - \sum_{i=1}^{r}\lvert d_i\rvert$,	$b_i \in [-5, 10],\ \forall\, i \in \{1, 2, \ldots, r\}$,
$f_1 = \sum_{i=1}^{p}\lfloor a_i\rfloor$,	$c_i \in [-5, 10],\ \forall\, i \in \{1, 2, \ldots, q+s\}$,
$f_2 = \sum_{i=1}^{q}\lvert c_i\rvert^{i+1} + \sum_{i=q+1, i=i+2}^{q+s-1}(c_{i+1} - c_i)^2$,	$d_i \in [-5, 10],\ \forall\, i \in \{1, 2, \ldots, r\}$.
$f_3 = \sum_{i=1}^{r}\lvert(b_i)^2 - (d_i)^2\rvert$.	

Bi-objective Bilevel Test Problems

The Deb-Sinha (DS) test suite contains five test problems with two objectives at each level. All problems are scalable with respect to variable dimensions at both levels. Note that $\mathbf{x}_u = \mathbf{x}$ and $\mathbf{x}_l = \mathbf{y}$. The location and shape of respective upper and lower level Pareto-optimal fronts can be found at the original paper [15].

DS 1:

Minimize:

$$\begin{cases} F_1(\mathbf{x}, \mathbf{y}) = 1 + r - \cos(\alpha\pi x_1)) + \sum_{j=2}^{K}(x_j - \frac{j-1}{2})^2 + \tau\sum_{i=2}^{K}(y_i - x_i)^2 - r\ \cos\left(\gamma\frac{pi}{2}\frac{y_1}{x_1}\right) \\ F_2(\mathbf{x}, \mathbf{y}) = 1 + r - \sin(\alpha\pi x_1)) + \sum_{j=2}^{K}(x_j - \frac{j-1}{2})^2 + \tau\sum_{i=2}^{K}(y_i - x_i)^2 - r\ \sin\left(\gamma\frac{pi}{2}\frac{y_1}{x_1}\right) \end{cases}$$

subject to:

$$\mathbf{y} \in \underset{\mathbf{y}}{\operatorname{argmin}} \begin{cases} f_1(\mathbf{x},\mathbf{y}) = & y_1^2 + \sum_{i=2}^{K}(y_i - x_i)^2 + \sum_{i=2}^{K} 10(1 - \cos(\frac{\pi}{K}(y_i - x_i))) \\ f_2(\mathbf{x},\mathbf{y}) = & \sum_{i=1}^{K}(y_i - x_i)^2 + \sum_{i=2}^{K} 10|\sin(\frac{\pi}{K}(y_i - x_i))| \end{cases}$$

$$y_i \in [-K, K],\ i = 1, \ldots, K,\ x_1 \in [1, 4],\ x_j \in [-K, K],\ j = 2, \ldots, K.$$

Recommended parameter setting for this problem, $K = 10$ (overall 20 variables), $r = 0.1$, $\alpha = 1$, $\gamma = 1$, $\tau = 1$. This problem results in a convex upper level Pareto-optimal front in which one specific solution from each lower level Pareto-optimal front gets associated with each upper level Pareto-optimal solution.

DS 2:

Minimize:

$$\begin{cases} F_1(\mathbf{x},\mathbf{y}) = v_1(x_1) + \sum_{j=2}^{K}\left[x_j^2 + 10(1 - \cos(\frac{\pi}{K}x_j))\right] + \tau\sum_{i=2}^{K}(y_i - x_i)^2 - r\ \cos\left(\gamma\frac{\pi}{2}\frac{y_1}{x_1}\right) \\ F_2(\mathbf{x},\mathbf{y}) = v_2(x_1) + \sum_{j=2}^{K}\left[x_j^2 + 10(1 - \cos(\frac{\pi}{K}x_j))\right] + \tau\sum_{i=2}^{K}(y_i - x_i)^2 - r\ \sin\left(\gamma\frac{\pi}{2}\frac{y_1}{x_1}\right) \\ v_1(x_1) = \begin{cases} \cos(0.2\pi)x_1 + \sin(0.2\pi)\sqrt{|0.02\sin(5\pi x_1)|}, & \text{for } 0 \le x_1 \le 1; \\ x_1 - (1 - \cos(0.2\pi)), & \text{for } x_1 > 1. \end{cases} \\ v_2(x_1) = \begin{cases} -\sin(0.2\pi)x_1 + \cos(0.2\pi)\sqrt{|0.02\sin(5\pi x_1)|}, & \text{for } 0 \le x_1 \le 1; \\ 0.1(x_1 - 1) - \sin(0.2\pi), & \text{for } x_1 > 1. \end{cases} \end{cases}$$

subject to:

$$\mathbf{y} \in \underset{\mathbf{y}}{\operatorname{argmin}} \begin{cases} f_1(\mathbf{x},\mathbf{y}) = & y_1^2 + \sum_{i=2}^{K}(y_i - x_i)^2 \\ f_2(\mathbf{x},\mathbf{y}) = & \sum_{i=1}^{K} i(y_i - x_i)^2 \end{cases}$$

$$y_i \in [-K, K],\ i = 1, \ldots, K,\ x_1 \in [0.001, K],\ x_j \in [-K, K],\ j = 2, \ldots, K.$$

Recommended parameter setting for this problem, $K = 10$ (overall 20 variables), $r = 0.25$. Due to the use of periodic terms in v_1 and v_2 functions, the upper level Pareto front corresponds to only six discrete values of $y_1 = [0.001, 0.2, 0.4, 0.6, 0.8, 1]$. Setting $\tau = -1$ will introduces a conflict between upper and lower level problems. For this problem, a number of contiguous lower level Pareto-optimal solutions are Pareto-optimal at the upper level for each upper level Pareto-optimal variable vector.

DS 3:

Minimize:

$$\begin{cases} F_1(\mathbf{x},\mathbf{y}) = x_1 + \sum_{j=3}^{K}(x_j - j/2)^2 + \tau\sum_{i=3}^{K}(y_i - x_i)^2 - R(x_1)\cos\left(4\tan^{-1}(\frac{x_2 - y_2}{x_1 - y_1})\right), \\ F_2(\mathbf{x},\mathbf{y}) = x_2 + \sum_{j=3}^{K}(x_j - j/2)^2 + \tau\sum_{i=3}^{K}(y_i - x_i)^2 - R(x_1)\sin\left(4\tan^{-1}(\frac{x_2 - y_2}{x_1 - y_1})\right), \end{cases}$$

subject to :

$$\mathbf{y} \in \underset{\mathbf{y}}{\operatorname{argmin}} \begin{cases} f_1(\mathbf{x},\mathbf{y}) = y_1 + \sum_{i=3}^{K}(y_i - x_i)^2, \\ f_2(\mathbf{x},\mathbf{y}) = y_2 + \sum_{i=3}^{K}(y_i - x_i)^2, \\ \text{subject to :} \quad g_1(y) = (y_1 - x_1)^2 + (y_2 - x_2)^2 \leq r^2, \end{cases}$$

$$G(\mathbf{x}) = x_2 - (1 - x_1^2) \geq 0,$$

$$y_i \in [-K, K],\ i = 1, \ldots, K,\ x_j \in [0, K],\ j = 1, \ldots, K,\ x_1 \text{ is a multiple of } 0.1.$$

In this test problem, the variable x_1 is considered to be discrete, thereby causing only a few x_1 values to represent the upper level pareto front. Recommended parameter setting for this problem: $R(x_1) = 0.1 + 0.15|\sin(2\pi(x_1 0.1)|$ and use $r = 0.2$, $\tau = 1$, and $K = 10$. Like in DS2, in this problem, parts of lower level Pareto-optimal front become upper level Pareto-optimal.

DS 4:

Minimize:

$$\begin{cases} F_1(\mathbf{x},\mathbf{y}) = (1 - y_1)(1 + \sum_{j=2}^{K} y_j^2)x_1, \\ F_2(\mathbf{x},\mathbf{y}) = y_1(1 + \sum_{j=2}^{K} y_j^2)x_1, \end{cases}$$

subject to :

$$\mathbf{y} \in \underset{\mathbf{y}}{\operatorname{argmin}} \begin{cases} f_1(\mathbf{x},\mathbf{y}) = (1 - y_1)(1 + \sum_{j=K+1}^{K+L} y_j^2)x_1, \\ f_2(\mathbf{x},\mathbf{y}) = y_1(1 + \sum_{j=K+1}^{K+L} y_j^2)x_1, \end{cases}$$

$$G(\mathbf{x}) = (1 - y_1)x_1 + \tfrac{1}{2}x_1 y_1 - 1 \geq 0,$$

$$1 \leq x_1 \leq 2, \quad -1 \leq y_1 \leq 1, \quad -(K + L) \leq y_i \leq (K + L),\ i = 2, \ldots, (K + L).$$

For this problem, there are a total of $K + L + 1$ variables. The original study recommended $K = 5$ and $L = 4$. This problem has a linear upper level Pareto-optimal front in which a single lower level solution from a linear Pareto-optimal front gets associated with the respective upper level variable vector.

DS 5:

This problem exactly the same as DS4, except

$$G(\mathbf{x}) = (1 - y_1)x_1 + \frac{1}{2}x_1 y_1 - 2 + \frac{1}{5}\left[5(1 - y_1)x_1 + 0.2\right] \geq 0.$$

This makes a number of lower level Pareto-optimal solutions to be Pareto-optimal at the upper level for each upper level variable vector.

References

1. M. J. Alves and J. P. Costa. An algorithm based on particle swarm optimization for multiobjective bilevel linear problems. *Appl. Math. Comput.*, 247(C):547–561, November 2014.
2. 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.
3. J. S. Angelo and H. J. C. Barbosa. A study on the use of heuristics to solve a bilevel programming problem. *International Transactions in Operational Research*, 22(5):861–882, 2015.
4. J. S. Angelo, E. Krempser, and H. J. C. Barbosa. Differential evolution assisted by a surrogate model for bilevel programming problems. In *Evolutionary Computation (CEC), 2014 IEEE Congress on*, pages 1784–1791. IEEE, 2014.
5. S. Bandaru and K. Deb. Metaheuristic techniques (chapter 11). In R. N. Sengupta, A. Gupta, and J. Dutta, editors, *Decision Sciences: Theory and Practice*. Boca Raton: CRC Press, 2016.
6. B. Barnhart, Z. Lu, M. Bostian, A. Sinha, K. Deb, L. Kurkalova, M. Jha, and G. Whittaker. Handling practicalities in agricultural policy optimization for water quality improvements. In *GECCO '17: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1065–1072, New York, NY, USA, 2017. ACM.
7. H. I. Calvete and C. Galé. On linear bilevel problems with multiple objectives at the lower level. *Omega*, 39(1):33–40, 2011.
8. H. I. Calvete, C. Gale, and P. M. Mateo. A new approach for solving linear bilevel problems using genetic algorithms. *European Journal of Operational Research*, 188(1):14–28, 2008.
9. H. I. Calvete, C. Galé, and M. Oliveros. Bilevel model for production–distribution planning solved by using ant colony optimization. *Computers & Operations Research*, 38(1):320–327, 2011.
10. J.-F. Camacho-Vallejo, R. Muñoz-Sánchez, and J. L. González-Velarde. A heuristic algorithm for a supply chain's production-distribution planning. *Computers & Operations Research*, 61:110–121, 2015.
11. C. A. C. Coello, D. A. VanVeldhuizen, and G. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer, Boston, MA, 2002.
12. K. Deb. *Multi-objective optimization using evolutionary algorithms*. Wiley, Chichester, UK, 2001.
13. K. Deb, S. Gupta, J. Dutta, and B. Ranjan. Solving dual problems using a coevolutionary optimization algorithm. *Journal of Global Optimization*, 57:891–933, 2013.
14. K. Deb and C. Myburgh. A population-based fast algorithm for a billion-dimensional resource allocation problem with integer variables. *European Journal of Operational Research*, 261(2):460–474, 2017.
15. K. Deb and A. Sinha. Constructing test problems for bilevel evolutionary multi-objective optimization. In *2009 IEEE Congress on Evolutionary Computation (CEC-2009)*, pages 1153–1160. IEEE Press, 2009.
16. K. Deb and A. Sinha. An evolutionary approach for bilevel multi-objective problems. In *Cutting-Edge Research Topics on Multiple Criteria Decision Making, Communications in Computer and Information Science*, volume 35, pages 17–24. Berlin, Germany: Springer, 2009.
17. K. Deb and A. Sinha. Solving bilevel multi-objective optimization problems using evolutionary algorithms. In *Evolutionary Multi-Criterion Optimization (EMO-2009)*, pages 110–124. Berlin, Germany: Springer-Verlag, 2009.
18. K. Deb and A. Sinha. An efficient and accurate solution methodology for bilevel multi-objective programming problems using a hybrid evolutionary-local-search algorithm. *Evolutionary Computation Journal*, 18(3):403–449, 2010.
19. G. Eichfelder. Solving nonlinear multiobjective bilevel optimization problems with coupled upper level constraints. Technical Report Preprint No. 320, Preprint-Series of the Institute of Applied Mathematics, Univ. Erlangen-Nornberg, Germany, 2007.
20. G. Eichfelder. Multiobjective bilevel optimization. *Mathematical Programming*, 123(2):419–449, June 2010.
21. N. Gadhi and S. Dempe. Necessary optimality conditions and a new approach to multiobjective bilevel optimization problems. *Journal of Optimization Theory and Applications*, 155(1):100–114, 2012.
22. F. Glover and G. A. Kochenberger. *Handbook of Metaheuristics*. Springer, 2003.

23. W. Halter and S. Mostaghim. Bilevel optimization of multi-component chemical systems using particle swarm optimization. In *Proceedings of World Congress on Computational Intelligence (WCCI-2006)*, pages 1240–1247, 2006.
24. S. R. Hejazi, A. Memariani, G. Jahanshahloo, and M. M. Sepehri. Linear bilevel programming solution by genetic algorithm. *Computers & Operations Research*, 29(13):1913–1925, 2002.
25. Md M. Islam, H. K. Singh, and T. Ray. A nested differential evolution based algorithm for solving multi-objective bilevel optimization problems. In *Proceedings of the Second Australasian Conference on Artificial Life and Computational Intelligence - Volume 9592*, pages 101–112, Berlin, Heidelberg, 2016. Springer-Verlag.
26. M. K. Jha, P. W. Gassman, and J. G. Arnold. Water quality modeling for the raccoon river watershed using swat. *Transactions of the ASAE*, 50(2):479–493, 2007.
27. Y. Jiang, X. Li, C. Huang, and X. 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.
28. J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of IEEE International Conference on Neural Networks*, pages 1942–1948. IEEE Press, 1995.
29. H. 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.
30. H. Li and Y. Wand. A hybrid genetic algorithm for solving nonlinear bilevel programming problems based on the simplex method. *International Conference on Natural Computation*, 4:91–95, 2007.
31. H. Li, Q. Zhang, Q. Chen, L. Zhang, and Y.-C. Jiao. Multiobjective differential evolution algorithm based on decomposition for a type of multiobjective bilevel programming problems. *Knowledge-Based Systems*, 107:271–288, 2016.
32. X. Li, P. Tian, and X. Min. A hierarchical particle swarm optimization for solving bilevel programming problems. In *Proceedings of Artificial Intelligence and Soft Computing (ICAISC 2006)*, pages 1169–1178, 2006. Also LNAI 4029.
33. M. Linnala, E. Madetoja, H. Ruotsalainen, and J. Hämäläinen. Bi-level optimization for a dynamic multiobjective problem. *Engineering Optimization*, 44(2):195–207, 2012.
34. Z. Lu, K. Deb, and A. Sinha. Uncertainty handling in bilevel optimization for robust and reliable solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 26(Suppl. 2):1–24, 2018.
35. R. Mathieu, L. Pittard, and G. Anandalingam. Genetic algorithm based approach to bi-level linear programming. *Operations Research*, 28(1):1–21, 1994.
36. J.-A. MejÃŋa-de Dios and E. Mezura-Montes. A physics-inspired algorithm for bilevel optimization. In *2018 IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC)*, pages 1–6, 2018.
37. C. O. Pieume, L. P. Fotso, and P. Siarry. Solving bilevel programming problems with multicriteria optimization techniques. *OPSEARCH*, 46(2):169–183, 2009.
38. S. Pramanik and P. P. Dey. Bi-level multi-objective programming problem with fuzzy parameters. *International Journal of Computer Applications*, 30(10):13–20, September 2011. Published by Foundation of Computer Science, New York, USA.
39. S. Ruuska and K. Miettinen. Constructing evolutionary algorithms for bilevel multiobjective optimization. In *Evolutionary Computation (CEC), 2012 IEEE Congress on*, pages 1–7, june 2012.
40. X. Shi. and H. S. Xia. Model and interactive algorithm of bi-level multi-objective decision-making with multiple interconnected decision makers. *Journal of Multi-Criteria Decision Analysis*, 10(1):27–34, 2001.
41. H. K. Singh, M. M. Islam, T. Ray, and M. Ryan. Nested evolutionary algorithms for computationally expensive bilevel optimization problems: Variants and their systematic analysis. *Swarm and Evolutionary Computation*, 48:329–344, 2019.
42. A. Sinha. Bilevel multi-objective optimization problem solving using progressively interactive evolutionary algorithm. In *Proceedings of the Sixth International Conference on Evolutionary Multi-Criterion Optimization (EMO-2011)*, pages 269–284. Berlin, Germany: Springer-Verlag, 2011.

43. A. Sinha, S. Bedi, and K. Deb. Bilevel optimization based on kriging approximations of lower level optimal value function. In *2018 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8. IEEE, 2018.
44. A. Sinha and K. Deb. Towards understanding evolutionary bilevel multi-objective optimization algorithm. In *IFAC Workshop on Control Applications of Optimization (IFAC-2009)*, volume 7. Elsevier, 2009.
45. A. Sinha, Z. Lu, K. Deb, and P. Malo. Bilevel optimization based on iterative approximation of multiple mappings. *arXiv preprint arXiv:1702.03394*, 2017.
46. A. Sinha, Z. Lu, K. Deb, and P. Malo. Bilevel optimization based on iterative approximation of multiple mappings. *Journal of Heuristics*, 2019 (Forthcoming).
47. A. Sinha, P. Malo, and K. Deb. Efficient evolutionary algorithm for single-objective bilevel optimization. *arXiv preprint arXiv:1303.3901*, 2013.
48. 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.
49. A. Sinha, P. Malo, and K. Deb. Test problem construction for single-objective bilevel optimization. *Evolutionary Computation Journal*, 22(3):439–477, 2014.
50. A. Sinha, P. Malo, and K. Deb. Towards understanding bilevel multi-objective optimization with deterministic lower level decisions. In *Proceedings of the Eighth International Conference on Evolutionary Multi-Criterion Optimization (EMO-2015)*. Berlin, Germany: Springer-Verlag, 2015.
51. A. Sinha, P. Malo, and K. Deb. Evolutionary algorithm for bilevel optimization using approximations of the lower level optimal solution mapping. *European Journal of Operational Research*, 2016.
52. 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.
53. A. Sinha, P. Malo, and K. Deb. Approximated set-valued mapping approach for handling multiobjective bilevel problems. *Computers and Operations Research*, 77:194–209, 2017.
54. A. Sinha, P. Malo, and K. Deb. A review on bilevel optimization: From classical to evolutionary approaches and applications. *IEEE Transactions on Evolutionary Computation*, 22(2):276–295, 2018.
55. A. Sinha, P. Malo, K. Deb, P. Korhonen, and J. Wallenius. Solving bilevel multi-criterion optimization problems with lower level decision uncertainty. *IEEE Transactions on Evolutionary Computation*, 20(2):199–217, 2016.
56. A. Sinha, P. Malo, A. Frantsev, and K. Deb. Finding optimal strategies in a multi-period multi-leader-follower stackelberg game using an evolutionary algorithm. *Computers & Operations Research*, 41:374–385, 2014.
57. A. Sinha, P. Malo, P. Xu, and K. Deb. A bilevel optimization approach to automated parameter tuning. In *Proceedings of the 16th Annual Genetic and Evolutionary Computation Conference (GECCO 2014)*. New York: ACM Press, 2014.
58. A. Sinha, T. Soun, and K. Deb. Using Karush-Kuhn-Tucker proximity measure for solving bilevel optimization problems. *Swarm and Evolutionary Computation*, 44:496–510, 2019.
59. R. Storn and K. Price. Differential evolution – A fast and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11:341–359, 1997.
60. Z. Wan, G. Wang, and B. 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.
61. G. Wang and S. Shan. Review of metamodeling techniques in support of engineering design optimization. *Journal of Mechanical Design*, 129(4):370–380, 2007.
62. G. Wang, Z. Wan, X. Wang, and Y. Lu. Genetic algorithm based on simplex method for solving linear-quadratic bilevel programming problem. *Computers & Mathematics with Applications*, 56(10):2550–2555, 2008.

63. 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.
64. Y. Wang, H. Li, and C. 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.
65. G. Whittaker, R. Färe, S. Grosskopf, B. Barnhart, M. Bostian, G. Mueller-Warrant, and S. Griffith. Spatial targeting of agri-environmental policy using bilevel evolutionary optimization. *Omega*, 66(A):15–27, 2017.
66. D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997.
67. J. J. Ye. Necessary optimality conditions for multiobjective bilevel programs. *Mathematics of Operations Research*, 36(1):165–184, 2011.
68. J. J. Ye and D. 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.
69. Y. Yin. Genetic algorithm based approach for bilevel programming models. *Journal of Transportation Engineering*, 126(2):115–120, 2000.
70. T. Zhang, T. Hu, X. Guo, Z. Chen, and Y. Cheng. Solving high dimensional bilevel multiobjective programming problem using a hybrid particle swarm optimization algorithm with crossover operator. *Knowledge-Based Systems*, 53:13–19, 2013.
71. T. Zhang, T. Hu, Y. Zheng, and X. Guo. An improved particle swarm optimization for solving bilevel multiobjective programming problem. *Journal of Applied Mathematics*, 2012.
72. X. Zhu, Q. Yu, and X. 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.