Efficient Evolutionary Algorithm for Single-Objective Bilevel Optimization

Ankur Sinha, Pekka Malo
Dept. of Information and Service Economy
Aalto University School of Business, Finland
Firstname.Lastname@aalto.fi

Kalyanmoy Deb, IEEE Fellow
Dept. of Mechanical Engineering
Indian Institute of Technology Kanpur, India
deb@iitk.ac.in

KanGAL Report Number 2013003

Abstract
Bilevel optimization problems are a class of challenging optimization problems, which contain two levels of optimization tasks. In these problems, the optimal solutions to the lower level problem become possible feasible candidates to the upper level problem. Such a requirement makes the optimization problem difficult to solve, and has kept the researchers busy towards devising methodologies, which can efficiently handle the problem. Despite the efforts, there hardly exists any effective methodology, which is capable of handling a complex bilevel problem. In this paper, we introduce bilevel evolutionary algorithm based on quadratic approximations (BLEAQ) of optimal lower level variables with respect to the upper level variables. The approach is capable of handling bilevel problems with different kinds of complexities in relatively smaller number of function evaluations. Ideas from classical optimization have been hybridized with evolutionary methods to generate an efficient optimization algorithm for generic bilevel problems. The efficacy of the algorithm has been shown on two sets of test problems. The first set is a recently proposed SMD test set, which contains problems with controllable complexities, and the second set contains standard test problems collected from the literature. The proposed method has been evaluated against two benchmarks, and the performance gain is observed to be significant.
1 Introduction

Bilevel optimization is a branch of optimization, which contains a nested optimization problem within the constraints of the outer optimization problem. The outer optimization task is usually referred as the upper level task, and the nested inner optimization task is referred as the lower level task. The lower level problem appears as a constraint, such that only an optimal solution to the lower level optimization problem is a possible feasible candidate to the upper level optimization problem. Such a requirement makes bilevel optimization problems difficult to handle and have kept researchers and practitioners busy alike. In the field of classical optimization, a number of studies have been conducted on bilevel programming [4, 25, 8]. Approximate solution techniques are commonly employed to handle bilevel problems with simplifying assumptions like smoothness, linearity or convexity. Some of the classical approaches commonly used to handle bilevel problems include the Karush-Kuhn-Tucker approach [3, 13], Branch-and-bound techniques [2] and the use of penalty functions [1]. Most of these solution methodologies are rendered inapplicable, as soon as the bilevel optimization problem becomes complex. Heuristic procedures such as evolutionary algorithms have also been developed for handling bilevel problems with higher levels of complexity [28, 26]. Most of the existing evolutionary procedures often involve enormous computational expense, which limit their utility to solving bilevel optimization problems only with smaller number of variables.

There are a number of practical problems which are bilevel in nature. They are often encountered in transportation (network design, optimal pricing), economics (Stackelberg games, principal-agent problem, taxation, policy decisions), management (network facility location, coordination of multi-divisional firms), engineering (optimal design, optimal chemical equilibria) etc [7]. Complex practical problems are usually modified into a simpler single level optimization task, which is solved to arrive at a satisfying solution instead of an optimal solution. For the complex bilevel problems, classical methods fail because of real world difficulties (like non-linearity, discreetness, non-differentiability, non-convexity etc), and evolutionary methods are not very useful because of their enormous computational expense, a hybrid strategy could be solution. In cognizance of the drawbacks associated with the two approaches, we propose a hybrid strategy in this paper, which utilizes principles from classical optimization, within an evolutionary algorithm to quickly approach a bilevel optimum. The proposed method is a bilevel evolutionary algorithm based on quadratic approximations (BLEAQ) of the lower level optimal variables as a function of upper level variables.

To begin with, we provide a review of the past work on bilevel optimization using evolutionary algorithms, followed by description of a general bilevel optimization problem. Thereafter, we provide a supporting evidence that a strategy based on iterative quadratic approximations of the lower level optimal variables with respect to the upper level variables could be used to converge towards the bilevel optimal solution. This is followed by the description of the methodology, which utilizes the proposed quadratic approximation principle within the
evolutionary algorithm. The proposed ideas are further supported by results on a number of test problems. Firstly, BLEAQ is evaluated on recently proposed unconstrained SMD test problems [23], where the method is shown to successfully handle test problems with 10 dimensions. A performance comparison is performed against a nested evolutionary strategy, and the efficiency gain is provided. Secondly, BLEAQ is evaluated on a set of standard test problems chosen from the literature. These are constrained test problems with smaller number of variables. For comparing the results obtained using BLEAQ, we choose the algorithm proposed in [27], which is able to successfully solve all the test problems.

2 Past research on Bilevel Optimization using Evolutionary Algorithms

Evolutionary algorithms for bilevel optimization have been proposed as early as in the 1990s. One of the first evolutionary algorithm for handling bilevel optimization problems was proposed by Mathieu et al. [18]. The proposed algorithm was a nested strategy, where the lower level was handled using a linear programming method, and the upper level was solved using a genetic algorithm (GA). Nested strategies could be one of the approaches to handle bilevel problems, where for every upper level vector a lower level optimization task is executed. However, a nested approach is computationally expensive and is not a feasible approach for large scale bilevel problems. Another nested approach was proposed in [28], where the lower level was handled using the Frank-Wolfe algorithm (reduced gradient method). The algorithm was successful in handling non-convex bilevel optimization problems, and the authors claimed it to be better than the classical methods. In 2005, Oduguwa and Roy [19] proposed a co-evolutionary approach for finding optimal solution for bilevel optimization problems. Their approach utilizes two populations. The first population handles upper level vectors, and the second population handles lower level vectors. The two populations interact with each other to converge towards the optimal solution. An extension of this study can be found in [11], where the authors solve a bilevel application problem with linear objectives and constraints.

Wang et al. [27] proposed an evolutionary algorithm based on a constraint handling scheme, where they successfully solve a number of standard test problems. The results produced by their approach finds better solutions for a number of test problems, as compared to what is reported in the literature. The algorithm is able to handle non-differentiability at the upper level objective function. However, the method may not be able to handle non-differentiability in the constraints, or the lower level objective function. Given the robustness of the approach in handling a variety of standard test problems, we choose this algorithm as one of the benchmarks. Another evolutionary algorithm proposed in [17] utilizes particle swarm optimization to handle the bilevel problems. Even this approach is nested as it solves the lower level optimization problem for
each upper level vector. The authors show that the approach is able to handle a number of standard test problems with smaller number or variables. However, they do not report the computational expense of the nested procedure. A hybrid approach proposed in [15], which is also nested, utilizes simplex-based crossover strategy at the upper level, and solves the lower level using one of the classical approaches. This method successfully solves a number of standard test problems, however given the nested nature of the algorithm it is not scalable for large number of variables. The authors report the number of generations and population sizes required by the algorithm, which may be used to compute the function evaluations at the upper level, but they do not explicitly report the total number of function evaluations required at the lower level.

Researchers in the field of evolutionary algorithms have also tried to convert the bilevel optimization problem into a single level optimization problem using the Karush-Kuhn-Tucker (KKT) conditions [26, 14, 16]. However, such conversions are possible only for those bilevel problems, where the lower level is smooth and the KKT conditions can be easily produced.

Recently, there has also been interest in multi-objective bilevel optimization using evolutionary algorithms. Some of the studies in the direction of solving multi-objective bilevel optimization problems using evolutionary algorithms are [12, 21, 6, 22, 20, 29].

3 Single-Objective Bilevel Problem

Bilevel optimization is a nested optimization problem which involves two levels of optimization tasks. The structure of a bilevel optimization problem demands that the optimal solutions to the lower level optimization problem may only be considered as feasible candidates for the upper level optimization problem. The problem contains two classes of variables, the upper level variables \( x_u \in X_U \subset \mathbb{R}^n \), and the lower level variables \( x_l \in X_L \subset \mathbb{R}^m \). For the lower level problem, the optimization task is performed with respect to variables \( x_l \), and the variables \( x_u \) act as parameters. A different \( x_u \) leads to a different lower level optimization problem, whose optimal solution needs to be determined. The upper level problem usually involves all variables \( x = (x_u, x_l) \), and the optimization is expected to be performed with respect to both the sets of variables. In the following we provide two equivalent definitions of a bilevel optimization problem:

**Definition 1** For the upper-level objective function \( F : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R} \) and lower-level objective function \( f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R} \)

\[
\begin{align*}
\text{minimize} & \quad F_0(x_u, x_l) \\
\text{subject to} & \quad x_l \in \arg\min \{ f_0(x_u, x_l) : f_j(x_u, x_l) \leq 0, j = 1, \ldots, J \} \\
& \quad F_k(x_u, x_l) \leq 0, k = 1, \ldots, K
\end{align*}
\]
The above definition can be stated in terms of set-valued mappings as follows:

**Definition 2** Let $\Psi : \mathbb{R}^n \rightrightarrows \mathbb{R}^m$ be a set-valued mapping,

$$\Psi(x_u) = \text{argmin}\{f_0(x_u, x_l) : f_j(x_u, x_l) \leq 0, j = 1, \ldots, J\},$$

which represents the constraint defined by the lower-level optimization problem, i.e. $\Psi(x_u) \subset X_L$ for every $x_u \in X_U$. Then the bi-level optimization problem can be expressed as a general constrained optimization problem:

$$\begin{align*}
\text{minimize} & \quad F_0(x_u, x_l) \\
\text{subject to} & \quad x_l \in \Psi(x_u) \\
& \quad F_k(x_u, x_l) \leq 0, k = 1, \ldots, K
\end{align*}$$

where $\Psi$ can be interpreted as a parameterized range-constraint for the lower-level decision $x_l$.

The graph of the optimal-decision mapping is interpreted as a subset of $X_U \times X_L$

$$\text{gph } \Psi = \{(x_u, x_l) \in X_U \times X_L \mid x_l \in \Psi(x_u)\},$$

which displays the connections between the upper-level decisions and corresponding optimal lower-level decisions. The domain of $\Psi$, which is obtained as a projection of gph $\Psi$ on the upper-level decision space $X_U$, represents all the points $x_u \in X_U$, where the lower-level (follower’s) problem has at least one optimal solution, i.e.

$$\text{dom } \Psi = \{x_u|\Psi(x_u) \neq \emptyset\}.$$ 

Similarly, the range is given by

$$\text{rge } \Psi = \{x_l|x_l \in \Psi(x_u) \text{ for some } x_u\},$$

which corresponds to the projection of gph $\Psi$ on the lower-level decision space $X_L$.

Figure 1 describes the structure of a bilevel problem in terms of two components: (i) gph $\Psi$, which gives the graph of the optimal decision-mapping $\Psi$ as a subset of $X_U \times X_L$; and (ii) the plot of $F_0$ evaluated on gph $\Psi$, which shows the upper level objective function with respect to upper level variables $x_u$, when the lower level is optimal $x_l \in \Psi(x_u)$. The shaded area of gph $\Psi$ shows the regions where there are multiple lower level optimal vectors corresponding to any upper level vector. On the other hand, the non-shaded parts of the graph represent the regions where $\Psi$ is a single-valued mapping, i.e. there is a single lower level optimal vector corresponding to any upper level vector. Considering gph $\Psi$ as the domain of $F_0$ in the figure, we can interpret $F_0$ entirely as a function of $x_u$, i.e. $F_0(x_u, \Psi(x_u))$. Therefore, whenever $\Psi$ is multi-valued, we can also see a shaded region in the plot of $F_0$ which shows the different upper level function values for any upper level vector with multiple lower level optimal solutions.
For instance, in Figure 1, the shaded region of $\text{gph } \Psi$ corresponds to the shaded region of $F_0$ for the upper level vectors between $x_u^1$ and $x_u^2$.

For a more detailed illustration, see the 3-dimensional graph in Figure 2, where the values of the upper and lower level objective functions $F_0$ and $f_0$ are plotted against the decision space $X_U \times X_L$. For simplicity, let us again assume that $\text{gph } \Psi$ is the domain of $F_0$. If we now consider the values of $F_0$ plotted against the plane of decision variables, we obtain the same information as before. However, as an addition to the previous figure, we have also described how the lower level objective function $f_0$ depends on the upper level decisions. In the figure, the shaded planes marked as A, B and C represent three different lower level optimization problems parameterized by $x_u^{(1)}$, $x_u^*$ and $x_u^{(2)}$, respectively. Once an upper-level decision vector has been fixed, the lower level objective can be interpreted entirely as a function of $x_l$. Hence each shaded plane shows a single-variable plot of $f_0$ against $X_L$ given a fixed $x_u$. Consider, for example, the plane A corresponding to the upper level decision $x_u^{(1)}$. From the shape of the lower level objective function it is easy to see that there are multiple optimal solutions at the lower level. Therefore, also $\Psi$ must be set-valued at this point, and the collection of optimal lower level solutions is given by $\Psi(x_u^{(1)})$. For the other two shaded planes B and C, there is only a single lower level optimal solution for the given $x_u$, which corresponds to $\Psi$ being single-valued at these points. The optimal upper level solution is indicated by point $(x_u^*, x_l^*)$, where $\Psi(x_u^*) = \{x_l^*\}$.
Figure 2: Graphical representation of a simple bilevel optimization problem.

4 Localization of the Follower’s Problem

Ideally, the analysis of a bilevel problem would be greatly simplified if the optimal solution mapping $\Psi$ could be treated as if it were an ordinary function. In particular, for the design of an efficient bilevel algorithm, it would be valuable to identify the circumstances under which single-valued functions can be used to construct local approximations for the optimal solution mapping. Given that our study of solution mappings is closely related to sensitivity analysis in parametric optimization, there already exist considerable research on the regularity properties of solution mappings, and especially on the conditions for obtaining single-valued localizations of general set-valued mappings. To formalize the notions of localization and single-valuedness in the context of set-valued mappings, we have adopted the following definition by Dontchev and Rockafellar [10]:

**Definition 3 (Localization and single-valuedness)** For a given set-valued mapping $\Psi : X_U \rightharpoonup X_L$ and a pair $(x_u, x_l) \in \text{gph } \Psi$, a graphical localization of $\Psi$ at $x_u$ for $x_l$ is a set-valued mapping $\Psi_{\text{loc}}$ such that

$$\text{gph } \Psi_{\text{loc}} = (U \times L) \cap \text{gph } \Psi$$

for some upper-level neighborhood $U$ of $x_u$ and lower-level neighborhood $L$ of $x_l$, i.e.

$$\Psi_{\text{loc}}(x_u) = \begin{cases} \Psi(x_u) \cap L & \text{for } x_u \in U, \\ \emptyset & \text{otherwise.} \end{cases}$$
If $\Psi_{loc}$ is actually a function with domain $U$, it is indicated by referring to a single-valued localization $\psi_{loc}: U \rightarrow X_L$ around $x_u$ for $x_l$.

Obviously, graphical localizations of the above type can be defined for any optimal-decision mapping. However, it is more interesting to ask when the localization is not only a single-valued function but also possesses convenient properties such as continuity and certain degree of smoothness. In this section, our plan is to study how the lower-level solution mappings behave under small perturbations, and clarify the regularity conditions in which the solution mapping can be locally characterized by a Lipschitz-continuous single-valued function. We begin the discussion from simple problems with convex set-constraints in section 4.1 and gradually extend the results to problems with general non-linear constraints in section 4.2.

As an example, Figure 3 shows the $\Psi$ mapping and its localization $\psi_{loc}$ around $x_u^{(0)}$ for $x_l^{(0)}$. The notations discussed in the previous definition are shown in the figure to develop a graphical insight for the theory discussed above.

### 4.1 Localization with Convex Constraints

To motivate the development, suppose that the follower’s problem is of form

\[
\text{minimize } f_0(x_u, x_l) \text{ over all } x_l \in C,
\]

where lower-level decisions are restricted by a simple set-constraint $C$ which is assumed to be non-empty, convex, and closed in $X_L$. 

![Figure 3: Localization around $x_u^{(0)}$ for $x_l^{(0)}$.](image_url)
When the lower-level objective function $f_0$ is continuously differentiable, the necessary condition for $x_l$ to be a local minimum for the follower’s problem is given by the standard variational inequality

$$\nabla_l f_0(x_u, x_l) + N_C(x_l) \ni 0,$$

where

$$N_C(x_l) = \{ v | \langle v, x'_l - x_l \rangle \leq 0, \text{ for all } x'_l \in C \}$$

is the normal cone to $C$ at $x_l$ and $\nabla_l f_0$ denotes the gradient of $f_0$ with respect to the lower-level decision vector. The solutions to the inequality are referred to as stationary points with respect to minimizing over $C$, regardless of whether or not they correspond to local or global minimum. Of course, in the special case when $f_0$ is also convex, the inequality yields a sufficient condition for $x_l$ to be a global minimum, but in other cases the condition is not sufficient to guarantee lower-level optimality of $x_l$. Rather the inequality is better interpreted as a tool for identifying quasi-solutions, which can be augmented with additional criteria to ensure optimality. In particular, as discussed by Dontchev and Rockafellar [10], significant efforts have been done to develop tools that help to understand the behavior of solutions under small perturbations which we are planning to utilize while proposing our solution for solving the bilevel problem. With this purpose in mind, we introduce the following definition of a quasi-solution mapping for the follower’s problem:

**Definition 4 (Quasi-solution mapping)** The solution candidates (stationary points) to the follower’s problem of form (1) can be identified by set-valued mapping $\Psi^* : X_U \rightrightarrows X_L$,

$$\Psi^*(x_u) = \{ x_l | \nabla_l f_0(x_u, x_l) + N_C(x_l) \ni 0 \},$$

which represents the set of stationary lower-level decisions for the given upper-level decision $x_u$. When $f_0$ is convex, $\Psi^*$ coincides to the optimal solution mapping, i.e. $\Psi^* = \Psi$.

Whereas direct localization of $\Psi$ is difficult, it is easier to obtain a well-behaved localization for the quasi-solution mapping first, and then establish the conditions under which the obtained solutions furnish a lower-level local minimum. The approach is motivated by the fact that for variational inequalities of the above type, there are several variants of implicit function theorem that can be readily applied to obtain localizations with desired properties. Below, we present two localization-theorems for quasi-solution mappings. The first theorem shows the circumstances in which there exists a single-valued localization of $\Psi^*$ such that it is Lipschitz-continuous around a given pair $(x_u^*, x_l^*) \in \text{gph} \Psi^*$. The second theorem elaborates the result further under the additional assumption that $C$ is polyhedral, which is sufficient to guarantee that for all points in the neighborhood of $x_u^*$ there is a strong local minimum in the follower’s problem (1).
Definition 5 (Lipschitz) A single-valued localization \( \psi_{\text{loc}} : X_U \to X_L \) is said to be Lipschitz continuous around an upper-level decision \( \bar{x}_u \) when there exists a neighborhood \( U \) of \( \bar{x}_u \) and a constant \( \mu \geq 0 \) such that
\[
|\psi_{\text{loc}}(x_u) - \psi_{\text{loc}}(x'_u)| \leq \mu |x_u - x'_u| \text{ for all } x_u, x_u' \in U.
\]

Theorem 6 (Localization of quasi-solution mapping) Suppose in the lower-level optimization problem (1), with \( x^*_u \in \Psi^*(x^*_l) \), that
(i) \( C \) is non-empty, convex, and closed in \( X_L \), and
(ii) \( f_0 \) is twice continuously differentiable with respect to \( x_l \), and has a strong convexity property at \((x^*_u, x^*_l)\), i.e. there exists \( \mu > 0 \) such that
\[
\langle \nabla_{ll} f_0(x^*_u, x^*_l), w \rangle \geq \mu |w|^2, \text{ for all } w \in C - C.
\]

Then the quasi-solution mapping \( \Psi^* \) has a Lipschitz continuous single-valued localization \( \psi \) around \( x^*_u \) for \( x^*_l \).

Proof. When the lower-level objective function \( f_0 \) is twice continuously differentiable with the inequality \( \langle \nabla_{ll} f_0(x^*_u, x^*_l), w \rangle \geq \mu |w|^2 \) holding for all \( w \in C - C \), then by Proposition 2G.4 and Exercise 2G.5 in [10] the assumptions (a) and (b) in Theorem 2G.2 are satisfied, and this gives the rest. \( \square \)

Corollary 7 (Localization with polyhedral constraints) Suppose that in the setup of Theorem (6) \( C \) is polyhedral. Then the additional conclusion holds that, for all \( x_u \) in some neighborhood of \( x^*_u \), there is a strong local minimum at \( x_l = \psi(x_u) \), i.e. for some \( \varepsilon > 0 \)
\[
f_0(x_u, x'_l) \geq f_0(x_u, x_l) + \frac{\varepsilon}{2}|x'_l - x_l|^2 \text{ for all } x'_l \in C \text{ near } x_l.
\]

Proof. See Exercise 2G.5 and Theorem 2G.3 in [10]. \( \square \)

It is worthwhile to note that second order conditions to ensure that a quasi-solution corresponds to an actual local minimum exist also for other than polyhedral sets, but the conditions are generally not available in a convenient form.

4.2 Localization with Nonlinear Constraints

Until now, we have considered the simple class of follower’s problems where the constraint set is not allowed to depend on the upper-level decisions. In practice, however, the follower’s constraints are often dictated by the leader’s choices. Therefore, the above discussion needs to be extended to cover the case of general non-linear constraints that are allowed to be functions of both \( x_l \) and \( x_u \). Fortunately, this can be done by drawing upon the results available for constrained parametric optimization problems.
Consider the follower’s problem of form

\[
\begin{align*}
\text{minimize} & \quad f_0(x_u, x_l) \\
\text{subject to} & \quad f_j(x_u, x_l) \leq 0, j = 1, \ldots, J
\end{align*}
\]

where the functions \( f_0, f_1, \ldots, f_J \) are assumed to be twice continuously differentiable. Let \( L(x_u, x_l, y) = f_0(x_u, x_l) + y_1 f_1(x_u, x_l) + \cdots + y_J f_J(x_u, x_l) \) denote the Lagrangian function. Then the necessary first-order optimality condition is given by the following variational inequality

\[
f(x_u, x_l, y) + N_E(x_l, y) \ni (0, 0),
\]

where

\[
\begin{align*}
E = X_L \times \mathbb{R}_+^J
\end{align*}
\]

The pairs \((x_l, y)\) which solve the variational inequality are called the Karush-Kuhn-Tucker pairs corresponding to the upper-level decision \( x_u \). Now in the similar fashion as done in Section 4.1, our interest is to establish the conditions for the existence of a mapping \( \psi_{\text{loc}} \) which can be used to capture the behavior of the \( x_l \) component of the Karush-Kuhn-Tucker pairs as a function of the upper-level decisions \( x_u \).

**Theorem 8 (Localization with Nonlinear Constraints)** For a given upper-level decision \( x_u^* \), let \((x_l^*, y^*)\) denote a corresponding Karush-Kuhn-Tucker pair. Suppose that the above problem for twice continuously differentiable functions \( f_i \) is such that the following regularity conditions hold

(i) the gradients \( \nabla f_i(x_u, x_l) \), \( i \in I \) are linearly independent, and

(ii) \( \langle w, \nabla^2 f_i(x_u^*, x_l^*, y^*) w \rangle > 0 \) for every \( w \neq 0, w \in M^+ \),

where

\[
\begin{align*}
I &= \{ i \in [1, J] \mid f_i(x_u^*, x_l^*) = 0 \}, \\
M^+ &= \{ w \in \mathbb{R}^n \mid w \perp \nabla f_i(x_u^*, x_l^*) \text{ for all } i \in I \}.
\end{align*}
\]

Then the Karush-Kuhn-Tucker mapping \( S : X_U \rightrightarrows X_L \times \mathbb{R}^J \),

\[
S(x_u) := \{(x_l, y) \mid f(x_u, x_l, y) + N_E(x_l, y) \ni (0, 0)\},
\]

has a Lipschitz-continuous single-valued localization \( s \) around \( x_u^* \) for \((x_l^*, y^*)\), \( s(x_u) = (\psi_{\text{loc}}(x_u), y(x_u)) \), where \( \psi_{\text{loc}} : X_U \to X_L \) and \( y : X_U \to \mathbb{R}^J \). Moreover, for every \( x_u \) in some neighborhood of \( x_u^* \) the lower-level decision component \( x_l = \psi_{\text{loc}}(x_u) \) gives a strong local minimum.

**Proof.** See e.g. Theorem 1 in [9] or Theorem 2G.9 in [10].

Despite the more complicated conditions required to establish the result for general non-linear constraints case, the eventual outcome of the Theorem (8) is essentially similar to Theorems (6) and (7). That is, in the vicinity of the current optimal solution, we can express the locally optimal follower’s decisions as a relatively smooth function of the upper-level decisions.
4.3 Approximation with Localizations

The single-valued localizations of solution mappings can be used as effective tools for alleviating computational burden in a greedy evolutionary algorithm. Instead of solving the follower’s problem from scratch for every upper-level decision, we can use localizations to obtain an “educated guess” on the new optimal lower-level decision. The intuition for using the above results is as follows.

Suppose that \((x_u, x_l) \in \text{gph } \Psi\) is a known lower-level optimal pair, i.e. \(x_l \in \Psi(x_u)\), and the lower-level problem satisfies the regularity conditions in the previous theorems. Then for some open neighborhoods \(U \subset X_U\) of \(x_u\) and \(L \subset X_L\) of \(x_l\), there exists a uniquely determined \(\mu\)-Lipschitz continuous function \(\psi_{loc} : U \rightarrow L\) such that \(x'_l = \psi_{loc}(x'_u)\) is the unique local optimal solution of the follower’s problem in \(L\) for each \(x'_u \in U\). The existence of such \(\psi_{loc}\) leads to a direct strategy for generating new solution candidates from the existing ones. If the currently known upper level decision \(x_u\) is perturbed by a small random vector \(\varepsilon\) such that \(x_u + \varepsilon \in U\), then the newly generated point \((x_u + \varepsilon, \psi_{loc}(x_u + \varepsilon))\) gives another locally optimal solution where the change in the lower-level optimal decision vector is bounded by \(|\psi_{loc}(x_u + \varepsilon) - x_l| \leq \mu|\varepsilon|\). In theory, this would allow us to reduce the bilevel optimization problem (2) to that of minimizing a locally Lipschitz function \(F_0(x_u, \psi_{loc}(x_u))\); see e.g. Dempe et al. for discussion on implicit function based techniques.

However, an essential difficulty for applying the result in practice follows from the fact that the mapping \(\psi_{loc}\) is not known with certainty, except for a few special cases. To resolve this problem in an efficient way, we consider embedding the above results within an evolutionary framework, where estimates of \(\psi_{loc}\) are produced by using samples of currently known optimal points. For simplicity, suppose that we want to find an estimate of \(\psi_{loc}\) around current best solution \((x_u, x_l) \in \text{gph } \Psi\), and let

\[
P = \{(x_u^{(i)}, x_l^{(i)}) \in X_U \times X_L \mid x_l^{(i)} \in \Psi(x_u^{(i)}), i \in I\} \subset \text{gph } \Psi
\]

be a sample from the neighborhood of \((x_u, x_l)\). Then the task of finding a good estimator for \(\psi_{loc}\) can be viewed as an ordinary supervised learning problem:

**Definition 9 (Learning of \(\psi_{loc}\))** Let \(\mathcal{H}\) be the hypothesis space, i.e. the set of functions that can be used to predict the optimal lower-level decision from the given upper-level decision. Given the sample \(P\), our goal is to choose a predictor \(\hat{\psi} \in \mathcal{H}\) such that it minimizes the empirical error on the sample data-set, i.e.

\[
\hat{\psi} = \arg\min_{h \in \mathcal{H}} \sum_{i \in I} L(h(x_u^{(i)}), x_l^{(i)}),
\]

where \(L : X_L \times X_L \rightarrow \mathbb{R}\) denotes the empirical risk function.

For a graphical illustration, see Figure 4 showing one example of a local approximation \(\hat{\psi}\) around \(x_u^{(0)}\) for \(x_l\) using a quadratic function. When the exact form of the underlying mapping is unknown, the approximation generally leads to an error as one moves away from the point around which the localization is
performed. In the figure, the approximation error is shown for a point $x_u^{(1)}$ in the neighborhood of $x_u^{(0)}$. This error may not be significant in the vicinity of $x_u^{(0)}$, and could provide a good guess for the lower-level optimal solution for a new $x_u$ close to $x_u^{(0)}$.

In this paper, we have chosen to use the squared prediction error as the empirical risk function when performing the approximations, i.e.

$$L(h(x_u^{(i)}), x_l^{(i)}) = |x_l^{(i)} - h(x_u^{(i)})|^2,$$

and at the same time we have restricted the hypothesis space $H$ to consist of second-order polynomials. With these choices the empirical risk minimization problem (4) corresponds to an ordinary quadratic regression problem. Therefore, as long as the estimation problem is kept light enough and the evolutionary framework is such that the solution population can be used to construct a training dataset $P$, the use of the estimation approach can be expected to enhance the algorithm’s overall performance by reducing the number of times the lower-level optimization problem needs to solved.

## 5 Algorithm description

In this section, we provide a description for the bilevel evolutionary algorithm based on quadratic approximations (BLEAQ). The optimization strategy is based on approximation of the lower level optimal variables as a function of upper level variables. To begin with, an initial population of upper level members with random upper level variables is initialized. For each member, the lower
level optimization problem is solved using a lower level optimization scheme, and optimal lower level members are noted. Based on the lower level optimal solutions achieved, a quadratic relationship between the upper level variables and each lower level optimal variable is established. If the approximation is good (in terms of mean squared error) then it can be used to predict the optimal lower level variables for any given set of upper level variables. This eliminates the requirement to solve a lower level optimization problem. However, one needs to be cautious while accepting the optimal solutions from the quadratic approximations as even a single poor solution might lead to a wrong bilevel optimum.

At each generation of the algorithm, a new quadratic approximation is generated which goes on improving as the population converges towards the true optima. At the termination of the procedure, the algorithm not only provides the optimal solutions to a bilevel problem, but also acceptably accurate functions representing the relationship between upper and lower variables at the optima. These functions can be useful for strategy developments or making decisions in bilevel problems appearing in fields like game theory, economics, science or engineering. In the following we provide a step-by-step procedure for the algorithm.

S. 1 Initialization: The algorithm starts with a random population of size $N$, which is initialized by generating the required number of upper level variables, and then executing the lower level evolutionary optimization procedure to determine the corresponding optimal lower level variables. Fitness is assigned based on upper level function value and constraints.

S. 2 Tagging: Tag all the upper level members as 1 which have undergone a lower level optimization run, and others as 0.

S. 3 Selection of upper level parents: Given the current population, we randomly choose $2(\mu - 1)$ number of members from the population and perform a tournament selection based on upper level function value. This produces $\mu - 1$ number of parents.

S. 4 Evolution at the upper level: Choose the best tag 1 member as the index parent, and the members from the previous step as other parents. Then, create $\lambda$ number of offsprings from the chosen $\mu$ parents, using crossover and polynomial mutation operators.

S. 5 Quadratic approximation: If the number of tag 1 members in the population is greater than $\frac{(\text{dim}(x_u)+1)\text{dim}(x_u)+2}{2}$, then select all the tag 1 upper level members to construct quadratic functions to represent each of the lower level optimal variables as a function of upper level variables. If the number of tag 1 members is less than $\frac{(\text{dim}(x_u)+1)\text{dim}(x_u)+2}{2}$ + dim$(x_u)$ then a quadratic approximation is not performed.

S. 6 Lower level optimum: If a quadratic approximation was performed in the previous step then find the lower level optima for the offsprings using the quadratic approximation. If the mean squared error $e_{mse}$ is less than
$e_0(1e^{-3})$, then the quadratic approximation is considered good and the offsprings are tagged as 1, otherwise they are tagged as 0. If a quadratic approximation was not performed in the previous step then execute lower level optimization runs for each of the offsprings. To execute lower level optimization for an offspring member, the closest tag 1 parent is determined. From the closest tag 1 parent, the lower level optimal member $(x^{(c)}_{i})$ is copied (Refer Section 5.1). Thereafter, a lower level optimization run is performed which first tries to optimize the problem using a quadratic programming approach, and if unsuccessful, it uses an evolutionary optimization algorithm. The copied lower level member from the closest upper level parent is used in the lower level optimization run. Tag the offspring as 1 for which lower level optimization is performed.

S. 7 Population Update: After finding the lower level variables for the offsprings, $r$ members are chosen from the parent population. A pool of chosen $r$ members and $\lambda$ offsprings is formed. The best $r$ members from the pool replace the chosen $r$ members from the population. A termination check is performed and the algorithm moves to the next generation (Step 3) if the termination check is false.

5.1 Property of two close upper level members

For two close upper level members, it is often expected that the lower level optimal solutions will also lie close to each other. This scenario is explained in Figure 5, where $x^{(1)}_u$ and $x^{(2)}_u$ are close to each other, and therefore their corresponding optimal lower level members are also close. On the other hand $x^{(3)}_u$ is far away from $x^{(1)}_u$ and $x^{(2)}_u$. Therefore, its optimal lower level member is not necessarily close to the other two lower level members. This property of the bilevel problems is utilized in the proposed algorithm, such that if a lower level optimization has been performed for one of the upper level members, then the corresponding lower level member is utilized while performing a lower level optimization task for another upper level member in the vicinity of the previous member.

5.2 Lower Level Optimization

At the lower level we first use a quadratic programming approach to find the optima. If the procedure is unsuccessful we use a global optimization procedure using an evolutionary algorithm to find the optimum. The lower level optimization using evolutionary algorithm is able to handle complex optimization problems with multimodality. The fitness assignment at this level is performed based on lower level function value and constraints. Following are the steps for the lower level optimization procedure:
5.2.1 Lower level quadratic programming

S. 1 Create \( \dim(x_l)+1 \) lower level points about \( x_l^{(c)} \) using polynomial mutation.

S. 2 Construct a quadratic function about \( x_l^{(c)} \) using the created points. Construct linear approximations for the constraints.

S. 3 Optimize the quadratic function with linear constraints using a sequentially quadratic programming approach.

S. 4 Compute the value of the optimum using the quadratic function and the lower level function. If the absolute difference is less than \( \delta_{\min} \) then accept the solution as lower level optimum, otherwise perform an evolutionary optimization search.

5.2.2 Lower level evolutionary optimization

S. 1 If lower level evolutionary optimization is executed directly without quadratic programming, then randomly initialize \( n \) lower level members. If quadratic programming is executed, then use the solution obtained using quadratic
programming as one of the population members and randomly initialize other \( n - 1 \) lower level members. The upper level variables are kept fixed for all the population members.

**S. 2** Choose \( 2\mu \) members randomly from the population, and perform a tournament selection to choose \( \mu \) parents for crossover.

**S. 3** The best parent among \( \mu \) parents is chosen as the index parent and \( \lambda \) number of offsprings are produced using the crossover and mutation operators.

**S. 4** A population update is performed by choosing \( r \) random members from the population. A pool is formed using \( r \) chosen members and \( \lambda \) offsprings, from which the best \( r \) members are used to replace the \( r \) chosen members from the population.

**S. 5** Next generation (Step 2) is executed if the termination criteria is not satisfied.

### 5.3 Constraint handling

As we know by now that a bilevel problem has two levels of optimization tasks. There may be constraints at the lower level and also constraints at the upper level. We modify any given bilevel problems such that lower level constraints belong only to the lower level optimization task, however, at the upper level we include both the upper and lower level constraints. This is done to ensure at the upper level that a solution which is not feasible at the lower level cannot be feasible at the upper level, no matter whether the lower level optimization task is performed or not. While computing the overall constraint violation for a solution at the upper level, it is not taken into account whether the lower level variables are optimal or not. We use a separate tagging scheme in the algorithm to account for optimality or non-optimality of lower level variables.

The algorithm uses similar constraint handling scheme at both the levels, where the overall constraint violation for any solution is the summation of the violations of all the equality and inequality constraints. A solution \( x^{(i)} \) is said to ‘constraint dominate’ [5] a solution \( x^{(j)} \) if any of the following conditions are true:

1. Solution \( x^{(i)} \) is feasible and solution \( x^{(j)} \) is not.
2. Solution \( x^{(i)} \) and \( x^{(j)} \) are both infeasible but solution \( x^{(i)} \) has a smaller overall constraint violation.
3. Solution \( x^{(i)} \) and \( x^{(j)} \) are both feasible but the objective value of \( x^{(i)} \) is less than that of \( x^{(j)} \).
5.4 Crossover Operator

The crossover operator used in Step 2 is similar to the PCX operator proposed in [24]. The operator creates a new solution from 3 parents as follows:

\[ c = x^{(p)} + \omega_\xi d + \omega_\eta \frac{p^{(2)} - p^{(1)}}{2} \]  

(5)

The terms used in the above equation are defined as follows:

- \( x^{(p)} \) is the index parent
- \( d = x^{(p)} - g \), where \( g \) is the mean of \( \mu \) parents
- \( p^{(1)} \) and \( p^{(2)} \) are the other two parents
- \( \omega_\xi = 0.1 \) and \( \omega_\eta = \frac{\text{dim}(x^{(p)})}{||x^{(p)} - g||} \) are the two parameters.

The two parameters \( \omega_\xi \) and \( \omega_\eta \), describe the extent of variations along the respective directions. At the upper level, a crossover is performed only with the upper level variables and the lower level variables are determined from the quadratic function or by lower level optimization call. At the lower level, crossover is performed only with the lower level variables and the upper level variables are kept fixed as parameters.

5.5 Termination Criteria

The algorithm uses a variance based termination criteria at both the levels. At the upper level, when the value of \( \alpha_u \) described in the following equation becomes less than \( \alpha_{u \text{ stop}} \), the algorithm terminates.

\[ \alpha_u = \sum_{i=1}^{M} \frac{\sigma^2(x_{uT}^i)}{\sigma^2(x_{u0}^i)}. \]  

(6)

From the above equation, \( \alpha_u \) is always greater than 0 and should be less than 1 most of the times. \( M \) is the number of upper level variables in the bilevel optimization problem, \( x_{uT}^i : i \in \{1, 2, \ldots, M\} \) represents the upper level variables in generation number \( T \), and \( x_{u0}^i : i \in \{1, 2, \ldots, M\} \) represents the upper level variables in the initial random population. The value of \( \alpha_{u \text{ stop}} \) should be kept small to ensure a high accuracy.

A similar termination scheme is used for the lower level evolutionary algorithm. The value for \( \alpha_l \) is determined as follows.

\[ \alpha_l = \sum_{i=1}^{m} \frac{\sigma^2(x_{lT}^i)}{\sigma^2(x_{l0}^i)}. \]  

(7)

In the above equation, \( m \) is the number of variables at the lower level, \( x_{lT}^i : i \in \{1, 2, \ldots, m\} \) represents the lower level variables in generation number \( T \), and \( x_{l0}^i : i \in \{1, 2, \ldots, m\} \) represents the lower level variables in the initial random population for a particular lower level optimization run. A high accuracy is desired particularly at the lower level, therefore the value for \( \alpha_l \) should be kept low. Inaccurate lower level solutions may mislead the algorithm in case of a conflict between the two levels.
Table 1: Overview of test-problem framework components

Panel A: Decomposition of decision variables

<table>
<thead>
<tr>
<th>Vector</th>
<th>Purpose</th>
<th>Vector</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_{u1}$</td>
<td>Complexity on upper-level</td>
<td>$x_{l1}$</td>
<td>Complexity on lower-level</td>
</tr>
<tr>
<td>$x_{u2}$</td>
<td>Interaction with lower-level</td>
<td>$x_{l2}$</td>
<td>Interaction with upper-level</td>
</tr>
</tbody>
</table>

Panel B: Decomposition of objective functions

<table>
<thead>
<tr>
<th>Component</th>
<th>Purpose</th>
<th>Component</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F_0^1(x_{u1})$</td>
<td>Difficulty in convergence</td>
<td>$f_0^1(x_{u1}, x_{u2})$</td>
<td>Functional dependence</td>
</tr>
<tr>
<td>$F_0^2(x_{l1})$</td>
<td>Conflict / co-operation</td>
<td>$f_0^2(x_{l1})$</td>
<td>Difficulty in convergence</td>
</tr>
<tr>
<td>$F_0^3(x_{u2}, x_{l2})$</td>
<td>Difficulty in interaction</td>
<td>$f_0^3(x_{u2}, x_{l2})$</td>
<td>Difficulty in interaction</td>
</tr>
</tbody>
</table>

5.6 Parameters

The parameters in the algorithm are fixed as $\mu = 3$, $\lambda = 2$ and $r = 2$ at both the level. Crossover probability is fixed at 0.9 and the mutation probability is 0.1. The upper level population size $N$ and the lower level population size $n$ are fixed at 50 for all the problems. The values for $\alpha_u^{stop}$ and $\alpha_l^{stop}$ are fixed as $1e-5$ at both the upper and the lower levels.

6 SMD problems

A set of test problems (SMD) for single objective bilevel optimization has recently been proposed in [23]. The proposed problems are scalable in terms of number of decision variables. In this section, we discuss about the construction of these test problems in brief and provide a description of the proposed SMD test problems. The construction of the test problem involves splitting the upper and lower level functions into three components. Each of the components is specialized for inducing a certain kind of difficulty into the bilevel test problem. The functions are chosen based on the difficulties required in terms of convergence at the two levels, and interaction between the two level. A generic bilevel test problem is stated as follows:

$$F_0(x_u, x_l) = F_0^1(x_{u1}) + F_0^2(x_{l1}) + F_0^3(x_{u2}, x_{l2})$$

$$f_0(x_u, x_l) = f_0^1(x_{u1}, x_{u2}) + f_0^2(x_{l1}) + f_0^3(x_{u2}, x_{l2})$$

where

$$x_u = (x_{u1}, x_{u2}) \quad \text{and} \quad x_l = (x_{l1}, x_{l2})$$

The above equations contain three terms at both the levels. Table 1 provides a summary for the role played by each term in the equations. The upper level and lower level variables are broken into two smaller vectors (refer Panel A in Table 1), such that, vectors $x_{u1}$ and $x_{l1}$ are used to induce complexities at the upper and lower levels independently, and vectors $x_{u2}$ and $x_{l2}$ are responsible to induce complexities because of interaction. The upper and lower level functions
are decomposed such that each of the components is specialized for a certain purpose only (refer Panel B in Table 1). The term \( F_1(x_{u1}) \), at the upper level, is responsible for inducing difficulty in convergence solely at the upper level, and the term \( f_2(x_{l1}) \), at the lower level, is responsible for inducing difficulty in convergence solely at the lower level. The term \( F_2(x_{l1}) \) decides if there is a conflict or a cooperation between the two levels. The terms \( F_3(x_{l2}, x_{u2}) \) and \( f_3(x_{l2}, x_{u2}) \) induce difficulties because of interaction at the two levels, though \( F_3(x_{l2}, x_{u2}) \) may also be used to induce a cooperation or a conflict. Finally, \( f_1(x_{u1}, x_{u1}) \) is a fixed term at the lower level which does not induce any difficulties, rather helps to create a functional dependence between lower level optimal solution(s) and the upper level variables.

### 6.1 SMD1

SMD1 is a test problem with cooperation between the two levels. The lower level optimization problem is a simple convex optimization task. The upper level is convex with respect to upper level variables and optimal lower level variables.

\[
\begin{align*}
F_0^1 & = \sum_{i=1}^{p} (x_{u1}^i)^2 \\
F_0^2 & = \sum_{j=1}^{q} (x_{l1}^j)^2 \\
F_0^3 & = \sum_{i=1}^{p} (x_{u2}^i)^2 + \sum_{i=1}^{r} (x_{u2}^i - \tan x_{l2}^i)^2 \\
f_0^1 & = \sum_{i=1}^{p} (x_{u1}^i)^2 \\
f_0^2 & = \sum_{j=1}^{q} (x_{l1}^j)^2 \\
f_0^3 & = \sum_{i=1}^{r} (x_{u2}^i - \tan x_{l2}^i)^2 
\end{align*}
\]

The range of variables is as follows,

\[
\begin{align*}
x_{u1}^i & \in [-5, 10], \quad \forall \ i \in \{1, 2, \ldots, p\} \\
x_{u2}^i & \in [-5, 10], \quad \forall \ i \in \{1, 2, \ldots, r\} \\
x_{l1}^i & \in [-5, 10], \quad \forall \ i \in \{1, 2, \ldots, q\} \\
x_{l2}^i & \in \left(-\frac{\pi}{2}, \frac{\pi}{2}\right), \quad \forall \ i \in \{1, 2, \ldots, r\}
\end{align*}
\]

Relationship between upper level variables and lower level optimal variables is given as follows,

\[
\begin{align*}
x_{l1}^i & = 0, \quad \forall \ i \in \{1, 2, \ldots, p\} \\
x_{l2}^i & = \tan^{-1} x_{u2}^i, \quad \forall \ i \in \{1, 2, \ldots, r\}
\end{align*}
\]

The values of the variables at the optima are \( x_u = 0 \) and \( x_l \) is obtained by the relationship given above. Both the upper and lower level functions are equal to 0 at the optima. Figure 6 shows the contours of the upper level function with respect to the upper and lower level variables for a four variable test problem with \( \text{dim}(x_{u1}) = \text{dim}(x_{u2}) = \text{dim}(x_{l1}) = \text{dim}(x_{l2}) = 1 \). Figure 6 shows the contours of the upper level function at each \( x_u \) in Sub-figure P assuming that the lower level variables are optimal. Sub-figure S shows the behavior of the upper level function with respect to \( x_l \) at optimal \( x_u \).
6.2 SMD2

SMD2 is a test problem with a conflict between the upper level and lower level optimization tasks. The lower level optimization problem is a convex optimization task. An inaccurate lower level optimum may lead to upper level function value better than the true optimum for the bilevel problem. The upper level is convex with respect to upper level variables and optimal lower level variables.

\[
\begin{align*}
F_0^1 &= \sum_{i=1}^{p} (x_{u1}^i)^2 \\
F_0^2 &= -\sum_{i=1}^{q} (x_{l1}^i)^2 \\
F_0^3 &= \sum_{i=1}^{r} (x_{u2}^i)^2 - \sum_{i=1}^{r} (x_{u2}^i - \log x_{l2}^i)^2 \\
F_0^4 &= \sum_{i=1}^{r} (x_{l2}^i)^2 \\
f_0^1 &= \sum_{i=1}^{p} (x_{u1}^i)^2 \\
f_0^2 &= \sum_{i=1}^{q} (x_{l1}^i)^2 \\
f_0^3 &= \sum_{i=1}^{r} (x_{u2}^i - \log x_{l2}^i)^2
\end{align*}
\]  

(12)

The range of variables is as follows,

\[
\begin{align*}
x_{u1}^i &\in [-5, 10], \quad \forall \ i \in \{1, 2, \ldots, p\} \\
x_{u2}^i &\in [-5, 1], \quad \forall \ i \in \{1, 2, \ldots, r\} \\
x_{l1}^i &\in [-5, 10], \quad \forall \ i \in \{1, 2, \ldots, q\} \\
x_{l2}^i &\in (0, e], \quad \forall \ i \in \{1, 2, \ldots, r\}
\end{align*}
\]  

(13)

Relationship between upper level variables and lower level optimal variables is given as follows,

\[
\begin{align*}
x_{l1}^i &= 0, \quad \forall \ i \in \{1, 2, \ldots, q\} \\
x_{l2}^i &= \log^{-1} x_{u2}^i, \quad \forall \ i \in \{1, 2, \ldots, r\}
\end{align*}
\]  

(14)
The values of the variables at the optima are $x_u = 0$ and $x_l$ is obtained by the relationship given above. Both the upper and lower level functions are equal to 0 at the optima. Figure 7 represents the same information as in Figure 6 for a four variable bilevel test problem.

### 6.3 SMD3

SMD3 is a test problem with a cooperation between the two levels. The difficulty introduced is in terms of multi-modality at the lower level which contains the Rastrigin’s function. The upper level is convex with respect to upper level variables and optimal lower level variables.

\[
\begin{align*}
F_0^1 &= \sum_{i=1}^{p} (x_{u1}^i)^2 \\
F_0^2 &= \sum_{i=1}^{q} (x_{u2}^i)^2 \\
F_0^3 &= \sum_{i=1}^{r} (x_{u2}^i)^2 + \sum_{i=1}^{r} ((x_{u2}^i)^2 - \tan x_{l2}^i)^2 \\
f_0^1 &= \sum_{i=1}^{p} (x_{u1}^i)^2 \\
f_0^2 &= q + \sum_{i=1}^{q} (x_{l1}^i)^2 - \cos 2\pi x_{l1}^i \\
f_0^3 &= \sum_{i=1}^{r} ((x_{u2}^i)^2 - \tan x_{l2}^i)^2
\end{align*}
\]

The range of variables is as follows,

\[
\begin{align*}
x_{u1}^i &\in [-5, 10], \quad \forall \ i \in \{1, 2, \ldots, p\} \\
x_{u2}^i &\in [-5, 10], \quad \forall \ i \in \{1, 2, \ldots, r\} \\
x_{l1}^i &\in [-5, 10], \quad \forall \ i \in \{1, 2, \ldots, q\} \\
x_{l2}^i &\in \left(-\frac{\pi}{2}, \frac{\pi}{2}\right), \quad \forall \ i \in \{1, 2, \ldots, r\}
\end{align*}
\]
Relationship between upper level variables and lower level optimal variables is given as follows,

\[
\begin{align*}
x^*_1 &= 0, \quad \forall\ i \in \{1, 2, \ldots, q\} \\
x^*_2 &= \tan^{-1}(x^*_u)^2, \quad \forall\ i \in \{1, 2, \ldots, r\}
\end{align*}
\]  

(17)

The values of the variables at the optima are \(x_u = 0\) and \(x_l\) is obtained by the relationship given above. Both the upper and lower level functions are equal to 0 at the optima. Figure 8 shows the contours of the upper level function at each \(x_u\) in Sub-figure P assuming that the lower level variables are optimal. Sub-figure S shows the behavior of the lower level function at optimal \(x_u\).

### 6.4 SMD4

SMD4 is a test problem with a conflict between the two levels. The difficulty is in terms of multi-modality at the lower level which contains the Rastrigin’s function. The upper level is convex with respect to upper level variables and optimal lower level variables.

\[
\begin{align*}
F^1_0 &= \sum_{i=1}^{p} (x^1_i)^2 \\
F^2_0 &= -\sum_{i=1}^{q} (x^1_i)^2 \\
F^3_0 &= \sum_{i=1}^{p} (x^2_i)^2 - \sum_{i=1}^{r} (|x^1_i| - \log(1 + x^1_i)) \\
f^1_0 &= \sum_{i=1}^{p} (x^2_i)^2 \\
f^2_0 &= q + \sum_{i=1}^{q} \left( (x^1_i)^2 - \cos 2\pi x^1_i \right) \\
f^3_0 &= \sum_{i=1}^{r} (|x^2_i| - \log(1 + x^2_i))^2
\end{align*}
\]  

(18)
Figure 9: Upper and lower level function contours for a four variable SMD4 test problem.

The range of variables is as follows,

\[
x_{u1}^i \in [-5, 10], \quad \forall \ i \in \{1, 2, \ldots, p\}
\]
\[
x_{u2}^i \in [-1, 1], \quad \forall \ i \in \{1, 2, \ldots, r\}
\]
\[
x_{l1}^i \in [-5, 10], \quad \forall \ i \in \{1, 2, \ldots, q\}
\]
\[
x_{l2}^i \in [0, e], \quad \forall \ i \in \{1, 2, \ldots, r\}
\]

(19)

Relationship between upper level variables and lower level optimal variables is given as follows,

\[
x_{l1}^i = 0, \quad \forall \ i \in \{1, 2, \ldots, q\}
\]
\[
x_{l2}^i = \log^{-1}|x_{u2}^i| - 1, \quad \forall \ i \in \{1, 2, \ldots, r\}
\]

(20)

The values of the variables at the optima are \(x_u = 0\) and \(x_l\) is obtained by the relationship given above. Both the upper and lower level functions are equal to 0 at the optima. Figure 9 represents the same information as in Figure 8 for a four variable bilevel problem.

6.5 SMD5

SMD5 is a test problem with a conflict between the two levels. The difficulty introduced is in terms of multi-modality and convergence at the lower level. The lower level problem contains the banana function such that the global optimum lies in a long, narrow, flat parabolic valley. The upper level is convex with
respect to upper level variables and optimal lower level variables.

\[
\begin{align*}
F_0^1 & = \sum_{i=1}^{p} (x_{u1}^i)^2 \\
F_0^2 & = -\sum_{i=1}^{q} \left( (x_{l1}^i + 1)^2 + (x_{l1}^i - 1)^2 \right) \\
F_0^3 & = \sum_{i=1}^{r} (x_{u2}^i)^2 - \sum_{i=1}^{r} (|x_{u2}^i| - (x_{l2}^i)^2) \\
f_0^1 & = \sum_{i=1}^{p} (x_{u1}^i)^2 \\
f_0^2 & = \sum_{i=1}^{q} (x_{l1}^i)^2 + \sum_{i=q+1}^{q+s} (x_{l1}^i)^2 \\
f_0^3 & = \sum_{i=1}^{r} (x_{u2}^i)^2 - \sum_{i=1}^{r} (x_{u2}^i - x_{l2}^i)^2 \\
f_0^4 & = \sum_{i=1}^{r} (x_{u2}^i)^2 - \sum_{i=q+1}^{q+s} (x_{l1}^i - x_{l1}^{i+1})^2 \\
f_0^5 & = \sum_{i=1}^{r} (x_{u2}^i - x_{l2}^i)^2
\end{align*}
\]

(21)

The range of variables is as follows,

\[
\begin{align*}
x_{u1}^i & \in [-5, 10], \quad \forall \ i \in \{1, 2, \ldots, p\} \\
x_{u2}^i & \in [-5, 10], \quad \forall \ i \in \{1, 2, \ldots, r\} \\
x_{l1}^i & \in [-5, 10], \quad \forall \ i \in \{1, 2, \ldots, q\} \\
x_{l2}^i & \in [-5, 10], \quad \forall \ i \in \{1, 2, \ldots, r\}
\end{align*}
\]

(22)

Relationship between upper level variables and lower level optimal variables is given as follows,

\[
\begin{align*}
x_{l1}^i & = 1, \quad \forall \ i \in \{1, 2, \ldots, q\} \\
x_{l2}^i & = \sqrt{|x_{u2}^i|}, \quad \forall \ i \in \{1, 2, \ldots, r\}
\end{align*}
\]

(23)

The values of the variables at the optima are \( x_u = 0 \) and \( x_l \) is obtained by the relationship given above. Both the upper and lower level functions are equal to 0 at the optima.

### 6.6 SMD6

SMD6 is a test problem with a conflict between the two levels. The problem contains infinitely many global solutions at the lower level, for any given upper level vector. Out of the entire global solution set, there is only a single lower level point which corresponds to the best upper level function value.

\[
\begin{align*}
F_0^1 & = \sum_{i=1}^{p} (x_{u1}^i)^2 \\
F_0^2 & = -\sum_{i=1}^{q} (x_{l1}^i)^2 + \sum_{i=q+1}^{q+s} (x_{l1}^i)^2 \\
F_0^3 & = \sum_{i=1}^{r} (x_{u2}^i)^2 - \sum_{i=1}^{r} (x_{u2}^i - x_{l2}^i)^2 \\
f_0^1 & = \sum_{i=1}^{p} (x_{u1}^i)^2 \\
f_0^2 & = \sum_{i=1}^{q} (x_{l1}^i)^2 + \sum_{i=q+1, i=i+1}^{q+s} (x_{l1}^i - x_{l1}^{i+1})^2 \\
f_0^3 & = \sum_{i=1}^{r} (x_{u2}^i - x_{l2}^i)^2
\end{align*}
\]

(24)

The range of variables is as follows,

\[
\begin{align*}
x_{u1}^i & \in [-5, 10], \quad \forall \ i \in \{1, 2, \ldots, p\} \\
x_{u2}^i & \in [-5, 10], \quad \forall \ i \in \{1, 2, \ldots, r\} \\
x_{l1}^i & \in [-5, 10], \quad \forall \ i \in \{1, 2, \ldots, q+s\} \\
x_{l2}^i & \in [-5, 10], \quad \forall \ i \in \{1, 2, \ldots, r\}
\end{align*}
\]

(25)

Relationship between upper level variables and lower level optimal variables is given as follows,

\[
\begin{align*}
x_{l1}^i & = 0, \quad \forall \ i \in \{1, 2, \ldots, q\} \\
x_{l2}^i & = x_{u2}^i, \quad \forall \ i \in \{1, 2, \ldots, r\}
\end{align*}
\]

(26)
The values of the variables at the optima are $x_u = 0$ and $x_l$ is obtained by the relationship given above. Both the upper and lower level functions are equal to 0 at the optima.

7 Standard test problems

Tables 2 and 3 provide a set of 8 standard test problems collected from the literature. The dimensions of the upper and lower level variables are given in the first column, and the problem formulation is defined in the second column. The third column provides the best known solution available in the literature for the chosen test problems.

8 Results

In this section, we provide the results obtained on the SMD test problems and other test problems using the BLEAQ approach. The section is divided into two parts. The first part contains the results obtained using the BLEAQ approach, which has been compared with the results obtained using a nested procedure. It was difficult to identify an approach from the existing literature, which can efficiently handle the SMD test problems with 10 variables. Therefore, we have chosen the nested approach as the benchmark for this study. The second part contains the results obtained on 8 standard constrained test problems, which have been studied by others in the past. We compare our method against the approach proposed by Wang et al. (2005), which is able to handle all the test problems successfully.

8.1 Results for SMD test problems

We report the results obtained by the nested procedure and the BLEAQ approach in this sub-section on 10-dimensional unconstrained SMD test problems. For test problems SMD1 to SMD5 we choose $p = 3$, $q = 3$ and $r = 2$, and for SMD6 we choose $p = 3$, $q = 1$, $r = 2$ and $s = 2$. Tables 4 and 5 provide the results obtained using the nested approach on these test problems. The nested approach uses an evolutionary algorithm at both the levels. For every upper level member a lower level optimization problem is solved in this approach [23]. The approach successfully handles all the test problems. However, as can be observed from Table 4, the number of function evaluation required are very high at the lower level. We use the nested approach as a benchmark to assess the savings obtained from the BLEAQ approach. Tables 6 and 7 provide the results obtained using the BLEAQ approach. Fourth and Fifth column in Table 6 provides the median function evaluations required at the upper and lower levels respectively. The numbers in the brackets provide a ratio of the function evaluations required using nested approach against the function evaluations required
Table 2: Description of the selected standard test problems (TP1-TP4).

<table>
<thead>
<tr>
<th>Problem</th>
<th>Formulation</th>
<th>Best Known Sol.</th>
</tr>
</thead>
<tbody>
<tr>
<td>TP1</td>
<td>Minimize $F_0(x_u, x_i) = (x_u^1 - 30)^2 + (x_u^2 - 20)^2 - 20x_i^1 + 20x_i^2$, s.t. $y \in \arg\min_{(y)} \left{ f_0(x_u, x_i) = (x_u^1 - x_i^1)^2 + (x_u^2 - x_i^2)^2 \right}$, $x_u^1 + 2x_u^2 \geq 30, x_u^1 + x_u^2 \leq 25, x_u^2 \leq 15$</td>
<td>$F_0 = 225.0$ $f_0 = 100.0$</td>
</tr>
<tr>
<td>TP2</td>
<td>Minimize $F_0(x_u, x_i) = 2x_u^1 + 2x_u^2 - 3x_i^1 - 3x_i^2 - 60$, s.t. $y \in \arg\min_{(y)} \left{ f_0(x_u, x_i) = (x_i^1 - x_u^1 + 20)^2 + (x_i^2 - x_u^2 + 20)^2 \right}$, $x_i^1 - 2x_i^1 \geq 10, x_i^2 - 2x_i^2 \geq 10$, $-10 \geq y_i \geq 20, i = 1, 2$, $x_i^1 + x_i^2 + x_i^2 - 2x_i^2 \leq 40$, $0 \leq x_i \leq 50, i = 1, 2$</td>
<td>$F_0 = 0.0$ $f_0 = 100.0$</td>
</tr>
<tr>
<td>TP3</td>
<td>Minimize $F_0(x_u, x_i) = -(x_u^1)^2 - 3(x_u^2)^2 - 4x_i^1 + (x_i^2)^2$, s.t. $y \in \arg\min_{(y)} \left{ f_0(x_u, x_i) = 2(x_u^1)^2 + (x_i^1)^2 - 5x_i^2 \right}$ $x_u^2 + 3x_i^1 - 4x_i^2 \geq 4$, $0 \leq y_i, i = 1, 2$ $(x_i^1)^2 + 2x_i^2 \leq 4$, $0 \leq x_i, i = 1, 2$</td>
<td>$F_0 = -18.6787$ $f_0 = -1.0156$</td>
</tr>
<tr>
<td>TP4</td>
<td>Minimize $F_0(x_u, x_i) = -8x_u^1 - 4x_u^2 + 4x_i^1 - 40x_i^2 - 4x_i^3$, s.t. $y \in \arg\min_{(y)} \left{ f_0(x_u, x_i) = x_u^1 + 2x_u^2 + x_i^1 + x_i^2 + 2x_i^2 \right}$, $x_i^2 + x_i^3 - x_i^1 \leq 1$, $2x_u^2 - x_i^1 + 2x_i^2 - 0.5x_i^3 \leq 1$, $2x_u^2 + 2x_i^1 - x_i^2 - 0.5x_i^3 \leq 1$, $0 \leq y_i, i = 1, 2$</td>
<td>$F_0 = -29.2$ $f_0 = 3.2$</td>
</tr>
</tbody>
</table>
Table 3: Description of the selected standard test problems (TP5-TP8).

<table>
<thead>
<tr>
<th>Problem</th>
<th>Formulation</th>
<th>Best Known Sol.</th>
</tr>
</thead>
<tbody>
<tr>
<td>TP5</td>
<td>Minimize $F_0(x_u, x_l) = rt(x_u)x_u - 3x_1^2 - 4x_2^2 + 0.5t(x_l)x_l$, s.t. $y \in \arg\min_{(y)} \begin{cases} f_0(x_u, x_l) = 0.5t(x_l)hx_l - t(b(x_u))x_l \ -0.333x_1^2 + x_2^2 - 2 \leq 0 \ x_1^2 - 0.333x_2^2 - 2 \leq 0 \ 0 \leq y_i, \ i = 1, 2 \end{cases}$, where $h = \begin{pmatrix} 1 \ 3 \ 10 \end{pmatrix}$, $b(x) = \begin{pmatrix} -1 \ 2 \ -3 \end{pmatrix}$, $x, r = 0.1$</td>
<td>$F_0 = -3.6$ $f_0 = -2.0$</td>
</tr>
<tr>
<td>TP6</td>
<td>Minimize $F_0(x_u, x_l) = (x_u^1 - 1)^2 + 2x_1^2 - 2x_1^1$, s.t. $y \in \arg\min_{(y)} \begin{cases} f_0(x_u, x_l) = (2x_1^1 - 4)^2 + (2x_1^2 - 1)^2 + x_1^1x_1^2 \ 4x_2^1 + 5x_1^1 + 4x_2^2 \leq 12 \ 4x_1^1 - 4x_2^1 - 5x_1^2 \leq -4 \ 4x_1^1 - 4x_1^2 + 5x_2^2 \leq 4 \ 4x_1^1 - 4x_2^1 + 5x_2^2 \leq -4 \end{cases}$, $0 \leq x_1^1$</td>
<td>$F_0 = -1.2091$ $f_0 = 7.6145$</td>
</tr>
<tr>
<td>TP7</td>
<td>Minimize $F_0(x_u, x_l) = -\frac{(x_u^1 + x_1^2)(x_1^2 + x_2^2)}{1 + x_1^2 x_1^2 + x_2^2 x_2^2}$, s.t. $y \in \arg\min_{(y)} \begin{cases} f_0(x_u, x_l) = \frac{(x_u^1 + x_1^2)(x_1^2 + x_2^2)}{1 + x_1^2 x_1^2 + x_2^2 x_2^2} \ 0 \leq y_i \leq x_i, \ i = 1, 2 \ (x_1^1)^2 + (x_1^2)^2 \leq 100 \end{cases}$</td>
<td>$F_0 = -1.98$ $f_0 = 1.98$</td>
</tr>
<tr>
<td>TP8</td>
<td>Minimize $F_0(x_u, x_l) =</td>
<td>2x_1^1 + 2x_2^1 - 3x_1^2 - 3x_2^2 - 60</td>
</tr>
</tbody>
</table>
using BLEAQ. The nested approach requires 2 to 5 times more function evaluations at the upper level, and more than 10 times function evaluations at the lower level as compared to BLEAQ. Both the approaches are able to successfully handle all the test problems. However, BLEAQ dominates the nested approach by a significant margin, particularly in terms of the function evaluations at the lower level.

Table 4: Function evaluations (FE) for the upper level (UL) and the lower level (LL) from 11 runs of nested approach.

<table>
<thead>
<tr>
<th>Pr. No.</th>
<th>Best LL FE</th>
<th>Best UL FE</th>
<th>Median LL FE</th>
<th>Median UL FE</th>
<th>Worst LL FE</th>
<th>Worst UL FE</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMD1</td>
<td>834478</td>
<td>1195</td>
<td>1628136</td>
<td>2473</td>
<td>2037784</td>
<td>3258</td>
</tr>
<tr>
<td>SMD2</td>
<td>1001588</td>
<td>1495</td>
<td>1500725</td>
<td>2239</td>
<td>2095710</td>
<td>3341</td>
</tr>
<tr>
<td>SMD3</td>
<td>868658</td>
<td>1232</td>
<td>1412300</td>
<td>2231</td>
<td>1790398</td>
<td>2684</td>
</tr>
<tr>
<td>SMD4</td>
<td>530263</td>
<td>730</td>
<td>1106389</td>
<td>1634</td>
<td>1337435</td>
<td>1988</td>
</tr>
<tr>
<td>SMD5</td>
<td>1257659</td>
<td>1666</td>
<td>1947316</td>
<td>2875</td>
<td>2561079</td>
<td>3528</td>
</tr>
<tr>
<td>SMD6</td>
<td>1243533</td>
<td>1624</td>
<td>2326072</td>
<td>3076</td>
<td>2946779</td>
<td>3812</td>
</tr>
</tbody>
</table>

Table 5: Accuracy for the upper and lower levels, and the lower level calls from 11 runs of nested approach.

<table>
<thead>
<tr>
<th>Pr. No.</th>
<th>Median UL Accuracy</th>
<th>Median LL Accuracy</th>
<th>Median LL Calls</th>
<th>LL Evaluations</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMD1</td>
<td>0.0051</td>
<td>0.0016</td>
<td>2473</td>
<td>647.02</td>
</tr>
<tr>
<td>SMD2</td>
<td>0.0015</td>
<td>0.0005</td>
<td>2239</td>
<td>653.82</td>
</tr>
<tr>
<td>SMD3</td>
<td>0.0087</td>
<td>0.0026</td>
<td>2231</td>
<td>649.59</td>
</tr>
<tr>
<td>SMD4</td>
<td>0.0078</td>
<td>0.0027</td>
<td>1634</td>
<td>694.91</td>
</tr>
<tr>
<td>SMD5</td>
<td>0.0012</td>
<td>0.0032</td>
<td>2875</td>
<td>717.13</td>
</tr>
<tr>
<td>SMD6</td>
<td>0.0096</td>
<td>0.0071</td>
<td>3076</td>
<td>767.03</td>
</tr>
</tbody>
</table>

8.2 Convergence Study

In this sub-section we provide the results on the convergence of the algorithm towards the bilevel optimal solution. We performed a convergence study for the first two SMD test problems with 10-dimensions, and the results are presented in Figures 10 and 11 for one particular run. The figures show the progress of the elite member at each of the generations. The upper plot shows the convergence towards the upper level optimal function value, and the lower plot shows the convergence towards the lower level optimal function value. The algorithm
Table 6: Function evaluations (FE) for the upper level (UL) and the lower level (LL) from 11 runs of the proposed BLEAQ.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LL (Savings)</td>
<td>UL (Savings)</td>
<td></td>
</tr>
<tr>
<td>SDM1</td>
<td>99315</td>
<td>610</td>
<td>110716 (14.71)</td>
</tr>
<tr>
<td>SDM2</td>
<td>70032</td>
<td>376</td>
<td>91023 (16.49)</td>
</tr>
<tr>
<td>SDM3</td>
<td>110701</td>
<td>620</td>
<td>125546 (11.25)</td>
</tr>
<tr>
<td>SDM4</td>
<td>61326</td>
<td>410</td>
<td>81434 (13.59)</td>
</tr>
<tr>
<td>SDM5</td>
<td>102868</td>
<td>330</td>
<td>126371 (15.41)</td>
</tr>
<tr>
<td>SDM6</td>
<td>5558</td>
<td>658</td>
<td>8393 (277.14)</td>
</tr>
</tbody>
</table>

Table 7: Accuracy for the upper and lower levels, and the lower level calls from 11 runs of the proposed BLEAQ.

<table>
<thead>
<tr>
<th>Pr. No.</th>
<th>Median UL Accuracy</th>
<th>Median LL Accuracy</th>
<th>Median LL Calls</th>
<th>Median LL Evals</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDM1</td>
<td>0.006828</td>
<td>0.003521</td>
<td>528</td>
<td>202.47</td>
</tr>
<tr>
<td>SDM2</td>
<td>0.003262</td>
<td>0.002745</td>
<td>522</td>
<td>176.09</td>
</tr>
<tr>
<td>SDM3</td>
<td>0.009122</td>
<td>0.004364</td>
<td>603</td>
<td>225.91</td>
</tr>
<tr>
<td>SDM4</td>
<td>0.006957</td>
<td>0.002716</td>
<td>574</td>
<td>146.39</td>
</tr>
<tr>
<td>SDM5</td>
<td>0.004103</td>
<td>0.003773</td>
<td>513</td>
<td>243.68</td>
</tr>
<tr>
<td>SDM6</td>
<td>0.000000</td>
<td>0.000000</td>
<td>167</td>
<td>50.24</td>
</tr>
</tbody>
</table>

preserves the elite member at the upper level, so we observe a continuous improvement in the upper plot. However, in a bilevel problem an improvement in the elite at the upper level does not guarantee a continuous improvement in the lower level function value. Therefore, we observe that the lower level convergence plot is not a continuously reducing plot rather contains small humps.

Next, we evaluate the algorithm’s performance in approximating the actual \( \Psi \) mapping. For this we choose the problem SMD1, which has the following \( \Psi \) mapping.

\[
\begin{align*}
    x_{l1}^i &= 0, \quad \forall \ i \in \{1, 2, \ldots, p\} \\
    x_{l2}^i &= \tan^{-1} x_{u2}^i, \quad \forall \ i \in \{1, 2, \ldots, r\}
\end{align*}
\]  

We set \( p = 3 \), \( q = 3 \) and \( r = 2 \) to get a 10-dimensional SMD1 test problem. The \( \Psi \) mapping for SMD1 test problem is variable separable. Therefore, in order to show the convergence of the approximation on a 2-d plot, we choose the variable \( x_{l2}^i \), and show its approximation with respect to \( x_{l2}^i \). The true relationship between the variables is shown in Figure 12, along with the quadratic approximations generated at various generations. It can be observed that the
approximations in the vicinity of the true bilevel optimum improves with increasing number of generations.

8.3 Results for constrained test problems

In this sub-section, we provide results for 8 standard constrained test problems chosen from the literature. We compare our results against the approach proposed by Wang et al. (2005). The reason for the choice of this approach as a benchmark is that the approach was successful in solving all the chosen constrained test problems. The results obtained using the BLEAQ and WJL [27] approach has been provided in Tables 8, 9 and 10. Table 8 provides the minimum, median and maximum function evaluations required to solve the chosen problems using the BLEAQ approach. Table 9 provides the accuracy obtained at both the levels, in terms of absolute difference from the best known solution to a particular test problem. The numbers in the brackets provide the best solution known from the literature for each of the test problems. The table also provides the median number of lower level calls, and the average number of lower level function evaluations required per lower level call. Table 10 compares the mean function evaluations at the upper and lower levels required by BLEAQ against that required by WJL. In their paper, Wang et al. have reported the function evaluations as sum of function evaluations for the two levels. The reported metric in their paper is the average function evaluations from multiple runs of their algorithm. We have computed a similar metric for BLEAQ, and the results are reported in the table. It can be observed that WJL requires close to an order of magnitude times more function evaluations for most of the test problems. This clearly demonstrates the efficiency gain obtained using the BLEAQ approach. It also suggests that the mathematical insights used along with the evolutionary principles in the BLEAQ approach is helpful in converging quickly towards the bilevel optimal solution.
9 Conclusions

In this paper, we have propose an efficient bilevel evolutionary algorithm (BLEAQ) which works by approximating the optimal solution mapping between the lower level optimal solutions and the upper level variables. The algorithm not only converges towards the the optimal solution of the bilevel optimization problem, but also provides the optimal solution mapping at the optima. This provides
Table 9: Accuracy for the upper and lower levels, and the lower level calls from 11 runs of the proposed BLEAQ.

<table>
<thead>
<tr>
<th>Pr.</th>
<th>Median UL Accuracy</th>
<th>Median LL Accuracy</th>
<th>Median LL Calls</th>
<th>Median LL Evals</th>
</tr>
</thead>
<tbody>
<tr>
<td>TP1</td>
<td>0.000000 (225.0)</td>
<td>0.000000 (100.0)</td>
<td>206</td>
<td>72.76</td>
</tr>
<tr>
<td>TP2</td>
<td>0.012657 (0.0)</td>
<td>0.000126 (100.0)</td>
<td>235</td>
<td>61.79</td>
</tr>
<tr>
<td>TP3</td>
<td>0.000000 (-18.678711)</td>
<td>0.000000 (-1.015625)</td>
<td>112</td>
<td>40.00</td>
</tr>
<tr>
<td>TP4</td>
<td>0.040089 (-29.2)</td>
<td>0.007759 (3.2)</td>
<td>255</td>
<td>60.00</td>
</tr>
<tr>
<td>TP5</td>
<td>0.008053 (-3.6)</td>
<td>0.040063 (-2.0)</td>
<td>203</td>
<td>78.50</td>
</tr>
<tr>
<td>TP6</td>
<td>0.000099 (-1.2091)</td>
<td>0.000332 (7.6145)</td>
<td>233</td>
<td>75.23</td>
</tr>
<tr>
<td>TP7</td>
<td>0.093192 (-1.98)</td>
<td>0.093192 (1.98)</td>
<td>3862</td>
<td>95.42</td>
</tr>
<tr>
<td>TP8</td>
<td>0.001819 (0.0)</td>
<td>0.000064 (100.0)</td>
<td>200</td>
<td>61.50</td>
</tr>
</tbody>
</table>

Table 10: Comparison of BLEAQ against the results achieved by WJL approach.

<table>
<thead>
<tr>
<th>Pr. No.</th>
<th>Mean LL Func. Evals.</th>
<th>BLEAQ</th>
<th>WJL</th>
<th>Savings</th>
</tr>
</thead>
<tbody>
<tr>
<td>TP1</td>
<td>16228</td>
<td>85499</td>
<td>5.27</td>
<td></td>
</tr>
<tr>
<td>TP2</td>
<td>17210</td>
<td>256227</td>
<td>14.89</td>
<td></td>
</tr>
<tr>
<td>TP3</td>
<td>5256</td>
<td>92526</td>
<td>17.60</td>
<td></td>
</tr>
<tr>
<td>TP4</td>
<td>16690</td>
<td>291817</td>
<td>17.48</td>
<td></td>
</tr>
<tr>
<td>TP5</td>
<td>16738</td>
<td>77302</td>
<td>4.62</td>
<td></td>
</tr>
<tr>
<td>TP6</td>
<td>17717</td>
<td>163701</td>
<td>9.24</td>
<td></td>
</tr>
<tr>
<td>TP7</td>
<td>261387</td>
<td>1074742</td>
<td>4.11</td>
<td></td>
</tr>
<tr>
<td>TP8</td>
<td>14105</td>
<td>213522</td>
<td>15.14</td>
<td></td>
</tr>
</tbody>
</table>

valuable information, as to how the lower level optimal solution changes based on changes in the upper level variable vector in the vicinity of the bilevel optima. The algorithm has been tested on two sets of test problems. The first set of test problems are recently proposed SMD test problems which are scalable in terms of number of variables. The method has been tested on a 10-dimensional instance of these test problems. The second set of test problems are 8 standard test problems chosen from the literature. These problems are constrained and have relatively smaller number of variables. The results obtained using the BLEAQ approach has been compared against a nested strategy for the SMD test problems, and against the WJL approach for the standard constrained test problems. For both the sets, BLEAQ offers significant improvement in the number of function evaluations at both the levels.
10 Acknowledgements

Ankur Sinha and Pekka Malo would like to acknowledge the support provided by Liikesivistysrahasto and Helsinki School of Economics foundation. Authors of the paper would like to acknowledge the conversations with Prof. Yu-Ping Wang.

References


