Evolutionary Algorithm for Bilevel Optimization (BLEAQ)

Ankur Sinha, Pekka Malo and Kalyanmoy Deb

August 10, 2016

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 general bilevel problems. The code for the algorithms may be accessed from the website `http://bilevel.org`. This report only provides a basic introductory note on the algorithm. More information about the working of the algorithm and results can be found from the following papers. The algorithm implemented on the website might lead to slightly better results than what is given in the papers due to improvements made in the implementation.

1. Ankur Sinha, Pekka Malo, and Kalyanmoy Deb. Efficient evolutionary algorithm for single-objective bilevel optimization. arXiv preprint arXiv:1303.3901 (2013).
2. Ankur Sinha, Pekka Malo, and Kalyanmoy Deb. An improved bilevel evolutionary algorithm based on Quadratic Approximations. In 2014 IEEE Congress on Evolutionary Computation, 2014.
3. Ankur Sinha, Pekka Malo, and Kalyanmoy Deb. Evolutionary Algorithm for Bilevel Optimization using Approximations of the Lower Level Optimal Solution Mapping. European Journal of Operational Research, 2016 (In Press).

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. The hierarchical optimization structure may introduce difficulties such as non-convexity and disconnectedness even for simpler instances of bilevel optimization like bilevel linear programming problems. Bilevel linear programming is known to be strongly NP-hard [25], and it has been proven that merely evaluating a solution for optimality is also a NP-hard task [51]. This gives us an idea about the kind of challenges offered by bilevel problems with complex (non-linear, non-convex, discontinuous etc.) objective and constraint functions.

In the field of classical optimization, a number of studies have been conducted on bilevel programming [13, 50, 18]. 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 [9, 26], Branch-and-bound techniques [8], 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 [56, 53]. Most of the existing evolutionary procedures often involve enormous computational expense, which limits their utility to solving bilevel optimization problems 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) [37, 14, 10], economics (Stackelberg games, principal-agent problem, taxation, policy decisions) [23, 52, 12, 46, 45, 28], management (network facility location, coordination of multi-divisional firms) [29, 49, 5], engineering (optimal design, optimal chemical equilibria) [27, 48] etc [17, 7]. Complex practical problems are usually modified into a simpler single level optimization task, which is solved to arrive at a *satisficing*[1] instead of an optimal solution. For the complex bilevel problems, classical methods fail due to real world difficulties like non-linearity, discreteness, non-differentiability, non-convexity etc. Evolutionary methods are not very useful either because of their enormous computational expense. Under such a scenario, a hybrid strategy could be solution. Acknowledging the drawbacks associated with the two approaches, we propose a hybrid strategy that 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.

The remainder of the paper is organized as follows. In the next section, 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 that utilizes the proposed quadratic approximation principle within the evolutionary algorithm.

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. [36]. 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 are a popular approach to handle bilevel problems, where for every upper level vector a lower level optimization task is executed. However, they are computationally expensive and not feasible for large scale bilevel problems. Another nested approach was proposed in [56], where the lower level was handled using the Frank-Wolfe algorithm (reduced gradient method). The algorithm was successful in solving non-convex bilevel optimization problems, and the authors claimed it to be better than the classical methods. In 2005, Oduguwa and Roy [38] 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 [30], where the authors solve a

[1] Satisficing is a portmanteau of two words, satisfy and suffice. A satisficing solution need not be optimal but meets the needs of a decision maker.

bilevel application problem with linear objectives and constraints. Calvete et al. [11] proposed a genetic algorithm for linear bilevel problems. The authors claimed that their approach is also capable of handling quasi-concave bilevel problems with a linear lower level objective function.

Wang et al. [54] proposed an evolutionary algorithm based on a constraint handling scheme, where they successfully solve a number of standard test problems. 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. Later on, Wang et al. [55] provided an improved algorithm that was able to handle non-differentiable upper level objective function and non-convex lower level problem. The algorithm was shown to perform better than the one proposed in [54]. Another evolutionary algorithm proposed in [34] utilizes particle swarm optimization to solve 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 of variables. However, they do not report the computational expense of the nested procedure. A hybrid approach proposed in [32], 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 that 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. Other studies where authors have relied on a nested strategy include [46, 4]. In both of these studies an evolutionary algorithm has been used at both levels to handle bilevel problems.

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 [53, 31, 33]. 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 an 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 [24, 41, 16, 43, 40, 57].

3 Single-Objective Bilevel Problem

Bilevel optimization is a nested optimization problem that 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 sets of variables. In the following we provide two equivalent definitions of a bilevel optimization problem:

Definition 1 *At the upper-level the objective function and constraints are defined by* $F : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}$

and at the lower-level the objective function and constraints are defined by $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ *as follows:*

$$\begin{aligned} \underset{x_u \in X_U, x_l \in X_L}{\text{minimize}} \quad & F_0(x_u, x_l) \\ \text{subject to} \quad & x_l \in \operatorname{argmin}\{f_0(x_u, x_l) : f_j(x_u, x_l) \leq 0, \\ & \qquad\qquad\qquad\qquad j = 1, \ldots, J\} \\ & F_k(x_u, x_l) \leq 0, k = 1, \ldots, K \end{aligned}$$

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,*

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

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 bilevel optimization problem can be expressed as a general constrained optimization problem:*

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

where Ψ *can be interpreted as a parameterized range-constraint for the lower-level decision* x_l*.*

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

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

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

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

Similarly, the range is given by

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

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

Figure 1 describes the structure of a bilevel problem in terms of two components: (i) $\operatorname{gph} \Psi$, which gives the graph of the lower level decision-mapping Ψ as a subset of $X_U \times X_L$; and (ii) the plot of F_0 evaluated on $\operatorname{gph} \Psi$, 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 $\operatorname{gph} \Psi$ 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 Ψ is a single-valued mapping, i.e. there is a single lower level optimal vector corresponding to any upper level vector. Considering $\operatorname{gph} \Psi$ 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 Ψ 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 $\operatorname{gph} \Psi$ corresponds to the shaded region of F_0 for the upper level vectors between x_u^1 and x_u^2.

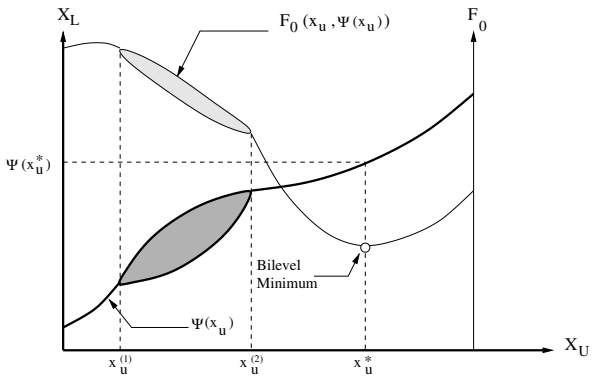

Figure 1: The Ψ-mapping for x_u and optimal x_l, and upper level objective function with respect to x_u when x_l is optimal.

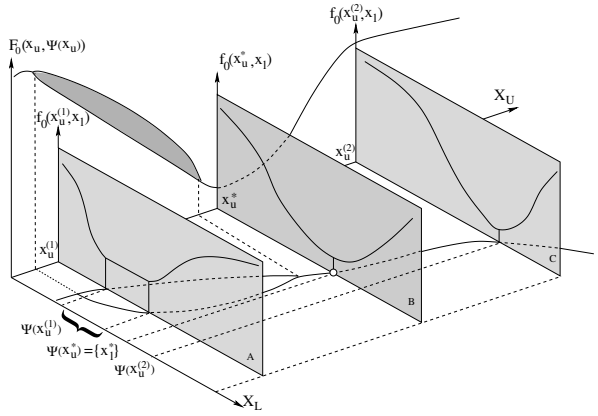

Figure 2: Graphical representation of a simple bilevel optimization problem.

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 $\mathrm{gph}\,\Psi$ is the domain of F_0. If we now consider the values of F_0 plotted against the upper level decision variables, we obtain a similar 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 $x_u^{(1)}$, $x_u^\star$ 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, Ψ must also 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 Ψ being single-valued at these points. The optimal upper level solution is indicated by point $(x_u^\star, x_l^\star)$, where $\Psi(x_u^\star) = \{x_l^\star\}$.

Example 3 *To provide a further insight into bilevel optimization, we consider a simple example with non-differentiable objective functions at upper and lower levels. The problem has a single upper and lower level variable, and does not contain any constraints at either of the two levels.*

$$\begin{aligned} \underset{x_u, x_l}{\text{minimize}} \quad & F_0(x_u, x_l) = |x_u| + x_l - 1 \\ \text{subject to} \quad & x_l \in \underset{x_l}{\operatorname{argmin}}\{f_0(x_u, x_l) = (x_u)^2 + |x_l - e^{x_u}|\} \end{aligned}$$

For a given upper level variable x_u, the optimum of the lower level problem in the above example is given by $x_l = e^{x_u}$. Therefore, this problem has a single-valued Ψ mapping, such that $\Psi(x_u) = \{e^{x_u}\}$. The Ψ mapping reduces this problem to a simple single-level optimization problem as a function of x_u. The optimal solution to the bilevel optimization problem is given by $(x_u^\star, x_l^\star) = (0, 1)$. In this example, the

lower level optimization problem is non-differentiable at the optimum for any given x_u, and the upper level objective function is non-differentiable at the bilevel optimum. Even though the problem involves simple objective functions at both levels, most of the KKT-based methods will face difficulties in handling such a problem. It is noteworthy that even though the objective functions at both levels are non-differentiable, the Ψ-mapping is continuous and smooth. For complex examples having disconnected or non-differentiable Ψ-mapping one can refer to [20].

4 Localization of the Lower Level Problem

Ideally, the analysis of a bilevel problem would be greatly simplified if the optimal solution mapping Ψ 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 exists 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 from Dontchev and Rockafellar [22]:

Definition 4 (Localization and single-valuedness) *For a given set-valued mapping $\Psi : X_U \rightrightarrows X_L$ and a pair $(x_u, x_l) \in \operatorname{gph} \Psi$, a graphical localization of Ψ at x_u for x_l is a set-valued mapping Ψ_{loc} such that*

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

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

$$\Psi_{\mathrm{loc}}(x_u) = \begin{cases} \Psi(x_u) \cap L & \text{for } x_u \in U, \\ \emptyset & \text{otherwise.} \end{cases}$$

If Ψ_{loc} is actually a function with domain U, it is indicated by referring to a single-valued localization $\psi_{\mathrm{loc}} : U \to X_L$ around x_u for x_l.

Obviously, graphical localizations of the above type can be defined for any lower level decision mapping. However, it is more interesting to ask when is the localization 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 Ψ mapping and its localization ψ_{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 lower level problem is of the form

$$\text{minimize } f_0(x_u, x_l) \text{ over all } x_l \in C, \tag{1}$$

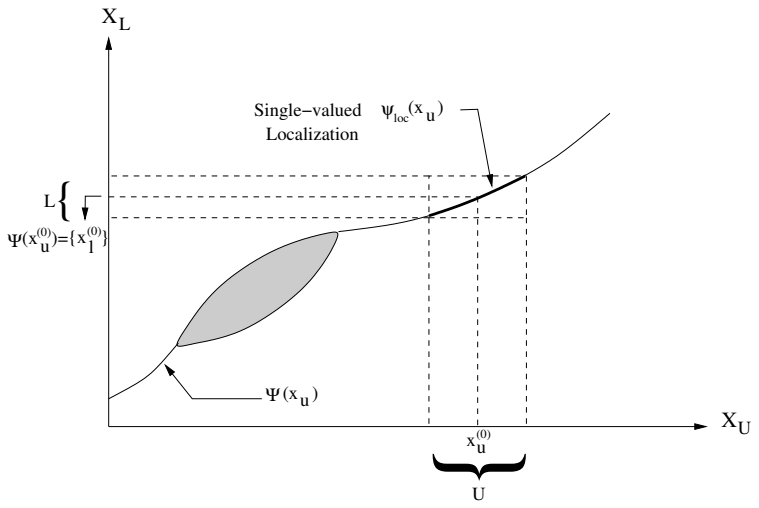

Figure 3: Localization around $x_u^{(0)}$ for $x_l^{(0)}$.

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

When the lower-level objective function f_0 is continuously differentiable, the necessary condition for x_l to be a local minimum for the lower level problem is given by the standard variational inequality

$$\nabla_l f_0(x_u, x_l) + N_C(x_l) \ni 0, \tag{2}$$

where

$$N_C(x_l) = \{v | \langle v, x_l' - x_l \rangle \leq 0, \text{ for all } x_l' \in C\} \tag{3}$$

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 [22], 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 lower level problem:

Definition 5 (Quasi-solution mapping) *The solution candidates (stationary points) to the lower level problem of form* (1) *can be identified by set-valued mapping* $\Psi^\star : X_U \rightrightarrows X_L$*,*

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

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

Whereas direct localization of Ψ 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 $\Psi^\star$ such that it is Lipschitz-continuous around a given pair $(x_u^\star, x_l^\star) \in \operatorname{gph} \Psi^\star$. 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 $x_u^\star$ there is a strong local minimum in the lower level problem (1).

Definition 6 (Lipschitz) *A single-valued localization $\psi_{\mathrm{loc}} : X_U \to X_L$ 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 $\gamma \geq 0$ such that*

$$|\psi_{\mathrm{loc}}(x_u) - \psi_{\mathrm{loc}}(x_u')| \leq \gamma |x_u - x_u'| \quad \textit{for all } x_u, x_u' \in U.$$

Theorem 7 (Localization of quasi-solution mapping) *Suppose in the lower-level optimization problem* (1)*, with $x_l^\star \in \Psi^\star(x_u^\star)$, 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 $(x_u^\star, x_l^\star)$, i.e. there exists $\gamma > 0$ such that

$$\langle \nabla_{ll} f_0(x_u^\star, x_l^\star) w, w \rangle \geq \gamma |w|^2, \quad \textit{for all } w \in C - C.$$

Then the quasi-solution mapping $\Psi^\star$ has a Lipschitz continuous single-valued localization ψ around $x_u^\star$ for $x_l^\star$.

Proof. When the lower-level objective function f_0 is twice continuously differentiable with the inequality $\langle \nabla_{ll} f_0(x_u^\star, x_l^\star) w, w \rangle \geq \gamma |w|^2$ holding for all $w \in C - C$, then by Proposition 2G.4 and Exercise 2G.5 in [22] the assumptions (a) and (b) in Theorem 2G.2 are satisfied, and this gives the rest. □

Corollary 8 (Localization with polyhedral constraints) *Suppose that in the setup of Theorem* (7) *C is polyhedral. Then the additional conclusion holds that, for all x_u in some neighborhood of $x_u^\star$, there is a strong local minimum at $x_l = \psi(x_u)$, i.e. for some $\varepsilon > 0$*

$$f_0(x_u, x_l') \geq f_0(x_u, x_l) + \frac{\varepsilon}{2} |x_l' - x_l|^2 \quad \textit{for all } x_l' \in C \textit{ near } x_l.$$

Proof. See Exercise 2G.5 and Theorem 2G.3 in [22]. □

It is worthwhile to note that second order conditions to ensure optimality of a quasi-solution also exist 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 lower level problems where the constraint set is not allowed to depend on the upper-level decisions. In practice, however, the lower level constraints are often dictated by the upper level 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 lower level problem of the form

$$\begin{aligned} \underset{x_l \in X_L}{\text{minimize}} \quad & f_0(x_u, x_l) \\ \text{subject to} \quad & f_j(x_u, x_l) \leq 0, j = 1, \ldots, J \end{aligned}$$

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

$$\begin{cases} f(x_u, x_l, y) = (\nabla_l L(x_u, x_l, y), -\nabla_y L(x_u, x_l, y)), \\ E = X_L \times \mathbb{R}^J_+ \end{cases}$$

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 ψ_{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 9 (Localization with Nonlinear Constraints) *For a given upper-level decision $x_u^\star$, let $(x_l^\star, y^\star)$ 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 $\nabla_l f_i(x_u, x_l)$, $i \in I$ are linearly independent, and

(ii) $\langle w, \nabla^2_{ll} L(x_u^\star, x_l^\star, y^\star) w \rangle > 0$ for every $w \neq 0$, $w \in M^+$,

where

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

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 $x_u^\star$ for $(x_l^\star, y^\star)$, $s(x_u) = (\psi_{\text{loc}}(x_u), y(x_u))$, 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 $x_u^\star$ the lower-level decision component $x_l = \psi_{\text{loc}}(x_u)$ gives a strong local minimum.

Proof. See e.g. Theorem 1 in [19] or Theorem 2G.9 in [22]. □

Despite the more complicated conditions required to establish the result for general non-linear constraints case, the eventual outcome of the Theorem (9) is essentially similar to Theorem (7) and Theorem (8). That is, in the vicinity of the current optimal solution, we can express the locally optimal lower level 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 lower level 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 $(x_u, x_l) \in \operatorname{gph} \Psi$ 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 γ-Lipschitz continuous function $\psi_{\mathrm{loc}} : U \to X_L$ such that $x_l' = \psi_{\mathrm{loc}}(x_u')$ is the unique local optimal solution of the lower level problem in L for each $x_u' \in U$. The existence of such ψ_{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 ε such that $x_u + \varepsilon \in U$, then the newly generated point $(x_u + \varepsilon, \psi_{\mathrm{loc}}(x_u + \varepsilon))$ gives another locally optimal solution where the change in the lower-level optimal decision vector is bounded by $|\psi_{\mathrm{loc}}(x_u+\varepsilon)-x_l| \leq \gamma|\varepsilon|$. In theory, this would allow us to reduce the bilevel optimization problem (2) to that of minimizing a locally Lipschitz function $F_0(x_u, \psi_{\mathrm{loc}}(x_u))$; see e.g. [21] for discussion on implicit function based techniques.

However, an essential difficulty for applying the result in practice follows from the fact that the mapping ψ_{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 ψ_{loc} are produced by using samples of currently known lower level optimal points. For simplicity, suppose that we want to find an estimate of ψ_{loc} around current best solution $(x_u, x_l) \in \operatorname{gph} \Psi$, and let

$$\mathcal{P} = \{(x_u^{(i)}, x_l^{(i)}) \in X_U \times X_L \mid x_l^{(i)} \in \Psi(x_u^{(i)}), i \in \mathcal{I}\} \subset \operatorname{gph} \Psi$$

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

Definition 10 (Learning of ψ_{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 $\mathcal{P}$, our goal is to choose a model $\hat{\psi} \in \mathcal{H}$ such that it minimizes the empirical error on the sample data-set, i.e.*

$$\hat{\psi} = \operatorname*{argmin}_{h \in \mathcal{H}} \sum_{i \in \mathcal{I}} L(h(x_u^{(i)}), x_l^{(i)}), \tag{4}$$

where $L : X_L \times X_L \to \mathbb{R}$ 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 $\mathcal{H}$ to consist of second-order polynomials. With these choices the empirical risk minimization problem (4) corresponds to an ordinary quadratic regression

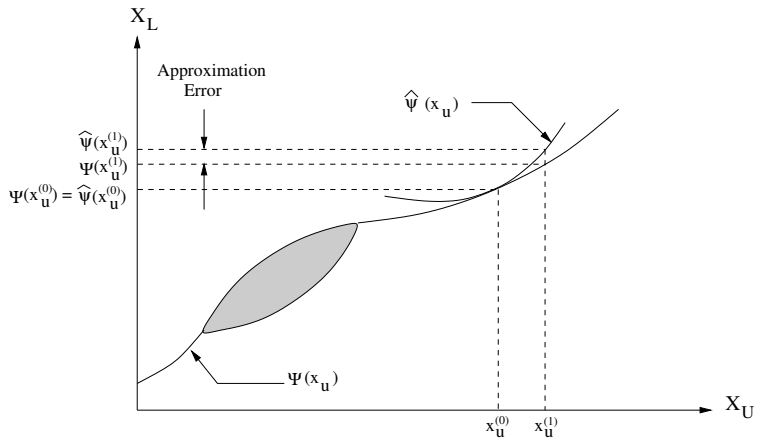

Figure 4: Approximation with localization around $x_u^{(0)}$.

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 $\mathcal{P}$, 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 be 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 optimum. 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 optimum. These functions can be useful for strategy development or decision making in bilevel problems appearing in fields like game theory, economics, science and engineering. Below we provide a step-by-step procedure for the algorithm.

S. 1 Initialize a random population of upper level variables of size N. Execute the lower level optimization problem to identify optimal lower level variables. Assign fitness based on upper level function value and constraints. Initialize generation number as $gen \leftarrow 0$.

S. 2 Tag all upper level members that have undergone a successful lower level optimization run as 1, and others as 0. Copy the tag 1 members to an archive.

S. 3 Increment generation number as $gen \leftarrow gen + 1$. Choose the best tag 1 member as one of the parents (index parent) from the population[2]. Randomly choose $2(\mu - 1)$ members from the population and perform a tournament selection based on upper level fitness to choose remaining $\mu - 1$ parents.

S. 4 Create λ offspring from the chosen μ parents, using crossover and polynomial mutation operators.

S. 5 If the number of tag 1 members in the population is greater than $\frac{N}{2}$ and archive size is greater than $\frac{(dim(x_u)+1)(dim(x_u)+2)}{2}+dim(x_u)$, then select $\frac{(dim(x_u)+1)(dim(x_u)+2)}{2}+dim(x_u)$ closest archive members[3] from the index parent. Construct quadratic functions $Q_t, t \in \{1, \ldots, dim(x_l)\}$ to represent lower level optimal variables as a function of upper level variables.

S. 6 If a quadratic approximation was performed in the previous step, find the lower level optimum for the offspring using the quadratic functions (Q_t). If the mean squared error e_{mse} is less than e_0(1e-3), the quadratic approximation is considered good and the offspring are tagged as 1, otherwise they are tagged as 0. If a quadratic approximation was not performed in the previous step, execute lower level optimization runs for each offspring. Tag the offspring as 1 for which a successful lower level optimization is performed.

S. 7 Copy the tag 1 offspring from the previous step (if any) to the archive. After finding the lower level variables for the offspring, choose r members from the parent population. A pool of chosen r members and λ offspring is formed. The best r members from the pool replace the chosen r members from the population.

S. 8 If gen is divisible by g_l and quadratic functions were generated at Step 5, then reduce the bilevel problem to a single level problem, such that optimal lower level solutions are given by the quadratic functions Q_t. Solve the single level optimization problem using local search (Refer to Subsection 5.3) around the best population member.

S. 9 If a local search is performed, test the upper level solution produced from the previous step by performing a lower level optimization at that point and evaluating the upper level fitness. If the newly produced point is better than the population best then replace the population best by the newly generated point.

S. 10 Perform a termination check. If the termination check is false, the algorithm moves to the next generation (Step 3).

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 $x_u^{(1)}$ and $x_u^{(2)}$ are close to each other, and therefore their corresponding optimal lower level members are also close. On the other hand $x_u^{(3)}$ is far away from $x_u^{(1)}$ and $x_u^{(2)}$. 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. 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.

[2]The choice of best tag 1 member as a parent makes the algorithm faster. However, for better exploration at upper level some other strategy may also be used.

[3]Please note that a quadratic fit in d dimensions requires at least $\frac{(d+1)(d+2)}{2}$ points. However, to avoid overfitting we use at least $\frac{(d+1)(d+2)}{2} + d$ points.

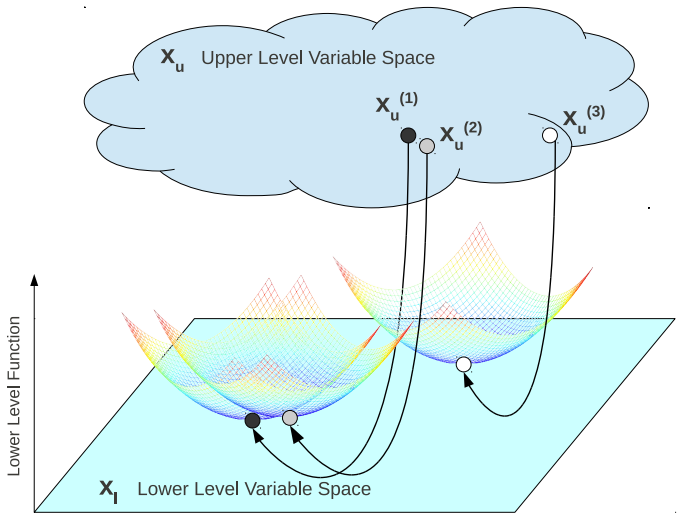

Figure 5: Lower level optimal solutions for three different upper level members.

5.2 Lower Level Optimization

At the lower level we attempt to automatically test if the lower level is a quadratic programming problem by creating a sample of points and approximating the functions and constraints. If the problem is actually a quadratic program, it is solved to optimum. If the problem is not a quadratic program, we still apply quadratic programming approach on the approximated functions. Thereafter, 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. The steps for the lower level optimization procedure are as follows:

5.2.1 Lower level quadratic programming

S. 1 Create $\frac{(dim(x_l)+1)(dim(x_l)+2)}{2} + dim(x_l)$ lower level points about $x_l^{(c)}$ using polynomial mutation.

S. 2 Construct a quadratic approximation for lower level objective function about $x_l^{(c)}$ using the created points. Construct linear approximations for the lower level 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 approximated objective function and the true lower level objective function. If the absolute difference is less than δ_{min} and the point is feasible with respect to the constraints, 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 already executed but unsuccessful, 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 2μ members randomly from the population, and perform a tournament selection to choose μ parents for crossover.

S. 3 The best parent among μ parents is chosen as the index parent and λ 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 λ 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 Local Search

To perform local search, we reduce the bilevel optimization problem to a single level optimization problem using the quadratic functions Q_t that approximate the lower level optimal solution for any given upper level vector. The auxiliary problem can be written as follows,

$$\begin{array}{ll} \underset{x \in X}{\text{Min}} & F(x_u, x_l), \\ \text{s.t.} & x_{l,t} = Q_t(x_u), \forall\, t \in \{1, \ldots, dim(x_l)\} \\ & G_j(x_u, x_l) \geq 0, j \in J. \end{array} \tag{5}$$

where Q_t is a quadratic function of the x_u variables. The above single level problem can be solved using sequential quadratic programming, if the functions are differentiable. The best upper level member in the population is used as a starting solution. If the functions $F(x_u, x_l)$ and $G_j(x_u, x_l)$ are non-differentiable, we approximate them with quadratic functions by sampling points around the best member in the population and then use sequential quadratic programming. Please note that Q_t represents a single-valued function between x_u and optimal x_l. It is not necessary that the mapping between x_u and optimal x_l is single-valued, rather there can be multiple optimal x_l for a given x_u. Therefore, the auxiliary problem should not be considered as a local approximation of the bilevel problem. The benefit in solving such a single level problem is that it is able to direct the BLEAQ approach into better regions in the search space.

5.4 Constraint handling

As we know by now that a bilevel problem has two levels of optimization tasks. There may be constraints at both levels. We modify any given bilevel problem such that lower level constraints belong only to the lower level optimization task. However, at the upper level we include both upper and lower level constraints. This is done to ensure 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 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' [15] 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.

5.5 Crossover Operator

The crossover operator used in Step 2 is similar to the PCX operator proposed in [47]. The operator creates a new solution from 3 parents as follows:

$$c = x^{(p)} + \omega_\xi d + \omega_\eta \frac{p^{(2)} - p^{(1)}}{2} \tag{6}$$

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 μ parents
- $p^{(1)}$ and $p^{(2)}$ are the other two parents
- $\omega_\xi = 0.1$ and $\omega_\eta = \frac{dim(x^{(p)})}{||x^{(p)} - g||_1}$ are the two parameters.

The two parameters ω_ξ and ω_η, 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.

5.6 Termination Criteria

The algorithm uses a variance based termination criteria at both levels. At the upper level, when the value of α_u described in the following equation becomes less than α_u^{stop}, the algorithm terminates.

$$\alpha_u = \frac{\sum_{i=1}^{n} \sigma^2(x^i_{u_T})}{\sum_{i=1}^{n} \sigma^2(x^i_{u_0})}, \tag{7}$$

where n is the number of upper level variables in the bilevel optimization problem, $x^i_{u_T} : \text{i} \in \{1, 2, \ldots, n\}$ represents the upper level variables in generation number T, and $x^i_{u_0} : \text{i} \in \{1, 2, \ldots, n\}$ represents the upper level variables in the initial random population. The value of α_u^{stop} should be kept small to ensure a high accuracy. Note that α_u is always greater than 0 and should be less than 1 most of the times.

A similar termination scheme is used for the lower level evolutionary algorithm. The value for α_l is given by:

$$\alpha_l = \frac{\sum_{i=1}^{m} \sigma^2(x^i_{l_t})}{\sum_{i=1}^{m} \sigma^2(x^i_{l_0})}, \tag{8}$$

where m is the number of variables at the lower level, $x^i_{l_t} : \text{i} \in \{1, 2, \ldots, m\}$ represents the lower level variables in generation number t, and $x^i_{l_0} : \text{i} \in \{1, 2, \ldots, m\}$ 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 α_l should be kept low. Inaccurate lower level solutions may mislead the algorithm in case of a conflict between the two levels.

5.7 Parameters and Platform

The parameters in the algorithm are fixed as $\mu = 3$, $\lambda = 2$, $r = 2$ and $g_l = 20$ at both levels. 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 α_u^{stop} and α_l^{stop} are fixed as $1e-4$ and $1e-5$ respectively at upper and the lower levels.

The code for the algorithm is written in MATLAB, and all the computations have been performed on a machine with 64 bit UNIX kernel, 2.6GHz quad-core Intel Core i7 processor and 8GB of 1600MHz DDR3 RAM.

6 Multiple Lower Level Optimization Problems

There can be situations where the lower level consists of multiple independent optimization problems instead of one. Implementation to handle such problems has not been included in the package, but still it can be made to handle such problems. If all the lower level optimization problems have their own independent variables, then we can replace all the lower level optimization problems with a single problem which is a summation of all the lower level problems. The bilevel optimal solution of this problem remains the same. This trick will allow one to use the BLEAQ algorithm directly.

References

[1] E. Aiyoshi and K. Shimizu. Hierarchical decentralized systems and its new solution by a barrier method. *IEEE Transactions on Systems, Man, and Cybernetics*, 11:444–449, 1981.

[2] E. Aiyoshi and K. Shimuzu. A solution method for the static constrained stackelberg problem via penalty method. *IEEE Transactions on Automatic Control*, AC-29(12):1112–1114, 1984.

[3] M. A. Amouzegar. A global optimization method for nonlinear bilevel programming problems. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 29(6):771–777, 1999.

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

[5] J. F. Bard. Coordination of multi-divisional firm through two levels of management. *Omega*, 11(5):457–465, 1983.

[6] J. F. Bard. *Practical Bilevel Optimization: Algorithms and Applications*. Kluwer Academic Publishers, 1998.

[7] J.F. Bard. *Practical Bilevel Optimization: Algorithms and Applications*. The Netherlands: Kluwer, 1998.

[8] J.F. Bard and J. Falk. An explicit solution to the multi-level programming problem. *Computers and Operations Research*, 9:77–100, 1982.

[9] L. Bianco, M. Caramia, and S. Giordani. A bilevel flow model for hazmat transportation network design. *Transportation Research Part C: Emerging technologies*, 17(2):175–196, 2009.

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

[11] Herminia I. Calvete, Carmen Gal, and Pedro M. Mateo. A new approach for solving linear bilevel problems using genetic algorithms. *European Journal of Operational Research*, 188(1):14 – 28, 2008.

[12] Mark Cecchini, Joseph Ecker, Michael Kupferschmid, and Robert Leitch. Solving nonlinear principal-agent problems using bilevel programming. *European Journal of Operational Research*, 230(2):364 – 373, 2013.

[13] B. Colson, P. Marcotte, and G. Savard. An overview of bilevel optimization. *Annals of Operational Research*, 153:235–256, 2007.

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

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

[16] 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.

[17] S. Dempe. Annotated bibliography on bilevel programming and mathematical programs with equilibrium constraints. *Optimization*, 52(3):339–359, 2003.

[18] S. Dempe, J. Dutta, and S. Lohse. Optimality conditions for bilevel programming problems. *Optimization*, 55(56):505–524, 2006.

[19] S. Dempe and S. Vogel. The subdifferential of the optimal solution in parametric optimization. Technical report, Fakultat fur Mathematik und Informatik, TU Bergakademie, 1997.

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

[21] Stephan Dempe. Bilevel programming: Implicit function approach. In Christodoulos A. Floudas and Panos M. Pardalos, editors, *Encyclopedia of Optimization*, pages 260–266. Springer, 2009.

[22] A.L. Dontchev and R.T. Rockafellar. *Implicit Functions and Solution Mappings: A View from Variational Analysis*. Springer, 1st edition, June 2009.

[23] D. Fudenberg and J. Tirole. *Game theory*. MIT Press, 1993.

[24] 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.

[25] P. Hansen, B. Jaumard, and G. Savard. New branch-and-bound rules for linear bilevel programming. *SIAM Journal on Scientific and Statistical Computing*, 13(5):1194–1217, 1992.

[26] J. Herskovits, A. Leontiev, G. Dias, and G. Santos. Contact shape optimization: A bilevel programming approach. *Struct Multidisc Optimization*, 20:214–221, 2000.

[27] C. Kirjner-Neto, E. Polak, and A.Der Kiureghian. An outer approximations approach to reliability-based optimal design of structures. *Journal of Optimization Theory and Applications*, 98(1):1–16, 1998.

[28] Raimund M. Kovacevic and Georg Ch. Pflug. Electricity swing option pricing by stochastic bilevel optimization: A survey and new approaches. *European Journal of Operational Research*, pages –, 2013.

[29] Hande Kkaydin, Necati Aras, and I. Kuban Altnel. Competitive facility location problem with attractiveness adjustment of the follower: A bilevel programming model and its solution. *European Journal of Operational Research*, 208(3):206 – 220, 2011.

[30] Francois Legillon, Arnaud Liefooghe, and El-Ghazali Talbi. Cobra: A cooperative coevolutionary algorithm for bi-level optimization. In *2012 IEEE Congress on Evolutionary Computation (CEC-2012)*. IEEE Press, 2012.

[31] Hecheng Li and Yuping Wang. A genetic algorithm for solving a special class of nonlinear bilevel programming problems. In Yong Shi, GeertDick Albada, Jack Dongarra, and PeterM.A. Sloot, editors, *Proceedings of the 7th international conference on Computational Science, Part IV: ICCS 2007*, volume 4490 of *Lecture Notes in Computer Science*, pages 1159–1162. Springer Berlin Heidelberg, 2007.

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

[33] Hecheng Li and Yuping Wang. An evolutionary algorithm with local search for convex quadratic bilevel programming problems. *Applied Mathematics and Information Sciences*, 5(2):139–146, 2011.

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

[35] B. D. Liu. Stackelberg-nash equilibrium for multi-level programming with multiple followers using genetic algorithms. *Computers and Mathematics with Applications*, 36(7):79–89, 1998.

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

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

[38] V. Oduguwa and R. Roy. Bi-level optimization using genetic algorithm. In *Proceedings of the 2002 IEEE International Conference on Artificial Intelligence Systems (ICAIS02)*, pages 322–327, 2002.

[39] J. V. Outrata. On the numerical solution of a class of stackelberg problems. *Zeitschrift Fur Operation Research*, AC-34(1):255–278, 1990.

[40] S. Ruuska and K. Miettinen. Constructing evolutionary algorithms for bilevel multiobjective optimization. In *2012 IEEE Congress on Evolutionary Computation (CEC-2012)*, 2012.

[41] 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.

[42] K. Shimizu and E. Aiyoshi. A new computational method for stackelberg and min-max problems by use of a penalty method. *IEEE Transactions on Automatic Control*, AC-26(2):460–466, 1981.

[43] 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.

[44] A. Sinha, P. Malo, and K. Deb. Unconstrained scalable test problems for single-objective bilevel optimization. In *2012 IEEE Congress on Evolutionary Computation (CEC-2012)*. IEEE Press, 2012.

[45] A. Sinha, P. Malo, A. Frantsev, and K. Deb. Multi-objective stackelberg game between a regulating authority and a mining company: A case study in environmental economics. In *2013 IEEE Congress on Evolutionary Computation (CEC-2013)*. IEEE Press, 2013.

[46] 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.

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

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

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

[50] L. N. Vicente and P. H. Calamai. Bilevel and multilevel programming: A bibliography review. *Journal of Global Optimization*, 5(3):291–306, 2004.

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

[52] F. J. Wang and J. Periaux. Multi-point optimization using gas and Nash/Stackelberg games for high lift multi-airfoil design in aerodynamics. In *Proceedings of the 2001 Congress on Evolutionary Computation (CEC-2001)*, pages 552–559, 2001.

[53] G. Wang, Z. Wan, X. Wang, and Y. Lv. Genetic algorithm based on simplex method for solving linear-quadratic bilevel programming problem. *Comput Math Appl*, 56(10):2550–2555, 2008.

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

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

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

[57] 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.