Bilevel Multi-Objective Optimization Problem Solving Using Evolutionary Algorithms

Kalyanmoy Deb kalyanmoy.deb@hse.fi
Department of Business Technology, Helsinki School of Economics, PO Box 1210,
FIN-00101, Helsinki, Finland

Ankur Sinha ankur.sinha@hse.fi
Department of Business Technology, Helsinki School of Economics, PO Box 1210,
FIN-00101, Helsinki, Finland

Abstract

Bilevel optimization problems require every feasible upper-level solution to satisfy optimality of a lower-level optimization problem. These problems commonly appear in many practical problem solving tasks including optimal control, process optimization, game-playing strategy development, transportation problems, and others. In the context of a bilevel single objective problem, there exists a number of theoretical, numerical, and evolutionary optimization results. However, there does not exist too many studies in the context of having multiple objectives in each level of a bilevel optimization problem. In this paper, we address bilevel multi-objective optimization issues and propose a viable algorithm based on evolutionary multi-objective optimization (EMO) principles. Proof-of-principle simulation results bring out the challenges in solving such problems and demonstrate the viability of the proposed EMO technique for solving such problems. This paper scratches the surface of EMO-based solution methodologies for bilevel multi-objective optimization problems and should motivate other EMO researchers to engage more into this important optimization task of practical importance.

Keywords

Genetic algorithms, evolutionary algorithms, bilevel optimization, multi-objective optimization, evolutionary programming hybrid evolutionary algorithms, sequential quadratic programming

1 Introduction

In evolutionary optimization, a few studies have considered solving bilevel programming problems in which an upper level solution is feasible only if it is one of the optimum of a lower level optimization problem. Such problems are found abundantly in practice, particularly in optimal control, process optimization, transportation problems, game playing strategies, reliability based design optimization, and others. In such problems, the lower level optimization task ensures a certain quality or certain physical properties which make a solution acceptable. Often, such requirements come up as equilibrium conditions, stability conditions, mass/energy balance conditions, which are mandatory for any solution to be feasible. For example. in reliability based design optimization, a feasible design must correspond to a certain specified reliability against failures. Solutions satisfying such conditions or requirements are not intuitive to obtain, rather they often demand an optimization problem to be solved. These essential tasks are posed as lower level optimization tasks in a bilevel optimization framework. The upper level optimization then must search among such reliable, equilibrium or stable solutions to find an optimal solution corresponding to one or more different (higher level) objectives.

Despite the importance of such problems in practice, the difficulty of searching and defining optimal solutions for bilevel optimization problems Dempe et al. (2006) exists. Despite the

©200X by the Massachusetts Institute of Technology

lack of theoretical results, there exists a plethora of studies related to bilevel single-objective optimization problems Calamai and Vicente (1994); Colson et al. (2007); Oduguwa and Roy (2002); Yin (2000) in which both upper and the lower level optimization tasks involve exactly one objective each. Despite having a single objective in the lower level task, usually in such problems the lower level optimization problem has more than one optimum. The goal of a bilevel optimization technique is then to first find the lower level optimal solutions and then search for the optimal solution for the upper level optimization task. In the context of bilevel multi-objective optimization studies, however, there does not exist too many studies using classical methods Eichfelder (2007) and none to our knowledge using evolutionary methods, probably because of the added complexities associated with solving each level. In such problems, every lower level optimization problem has a number of trade-off optimal solutions and the task of the upper level optimization algorithm is to focus its search on multiple trade-off solutions which are members of optimal trade-off solutions of lower level optimization problems.

In this paper, we suggest a viable evolutionary multi-objective optimization (EMO) algorithm for solving bilevel problems. We simulate the proposed algorithm on five different test problems, including a bilevel single-objective optimization problem. This proof-of-principle study shows viability of EMO for solving bilevel optimization problems and should encourage other EMO researchers to pay attention to this important class of practical optimization problems.

2 Description of Bilevel Multi-Objective Optimization Problem

A bilevel multi-objective optimization problem has two levels of multi-objective optimization problems such that the optimal solution of the lower level problem determines the feasible space of the upper level optimization problem. In general, the lower level problem is associated with a variable vector $\mathbf{x}_l$ and a fixed vector $\mathbf{x}_u$. However, the upper level problem usually involves all variables $\mathbf{x} = (\mathbf{x}_u, \mathbf{x}_l)$, but we refer here $\mathbf{x}_u$ exclusively as the upper level variable vector. A general bilevel multi-objective optimization problem can be described as follows:

$$\begin{array}{rl} \text{minimize}_{(\mathbf{x}_u,\mathbf{x}_l)} & \mathbf{F}(\mathbf{x}) = (F_1(\mathbf{x}), \ldots, F_M(\mathbf{x})), \\ \text{subject to} & \mathbf{x}_l \in \text{argmin}_{(\mathbf{x}_l)} \left\{ \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_m(\mathbf{x})) \,\middle|\, \mathbf{g}(\mathbf{x}) \geq \mathbf{0}, \mathbf{h}(\mathbf{x}) = \mathbf{0} \right\}, \\ & \mathbf{G}(\mathbf{x}) \geq \mathbf{0}, \mathbf{H}(\mathbf{x}) = \mathbf{0}, \\ & x_i^{(L)} \leq x_i \leq x_i^{(U)}, \quad i = 1, \ldots, n. \end{array} \tag{1}$$

In the above formulation, $F_1(\mathbf{x}), \ldots, F_M(\mathbf{x})$ are the upper level objective functions, and $\mathbf{G}(\mathbf{x})$ and $\mathbf{H}(\mathbf{x})$ are upper level inequality and equality constraints, respectively. The objectives $f_1(\mathbf{x}), \ldots, f_m(\mathbf{x})$ are the lower level objective functions, and functions $\mathbf{g}(\mathbf{x})$ and $\mathbf{h}(\mathbf{x})$ are lower level inequality and equality constraints, respectively. It should be noted that the lower level optimization problem is optimized only with respect to the variables $\mathbf{x}_l$ and the variable vector $\mathbf{x}_u$ is kept fixed. The Pareto-optimal solutions of a lower level optimization problem become feasible solutions to the upper level problem. The Pareto-optimal solutions of the upper level problem are determined by objectives $\mathbf{F}$ and constraints $\mathbf{G}$, and restricting the search among the lower level Pareto-optimal solutions.

3 Classical Approaches

Several studies exist in determining the optimality conditions for a Pareto-optimal solution to the upper level problem. The difficulty arises due to the existence of the lower level optimization problems. Usually the KKT conditions of the lower level optimization problems are used as constraints in formulating the KKT conditions of the upper level problem. As discussed in Dempe et al. (2006), although KKT optimality conditions can be written mathematically, the

presence of many lower level Lagrange multipliers and an abstract term involving coderivatives makes the procedure difficult to be applied in practice.

Fliege and Vicente Fliege and Vicente (2006) suggested a mapping concept in which a bilevel single-objective optimization problem can be converted to an equivalent four-objective optimization problem with a special cone dominance concept. Although the idea can be, in principle, extended for bilevel multi-objective optimization problems, the number of objectives to be considered is large and moreover handling constraints seems to introduce additional difficulties in obtaining resulting objectives. However, such an idea is interesting and can be pursued in the future.

In the context of bilevel single-objective optimization problems, there exists a number of studies, including some useful reviews Colson et al. (2007); Vicente and Calamai (2004), test problem generators Calamai and Vicente (1994), and even some evolutionary algorithm (EA) studies Oduguwa and Roy (2002); Mathieu et al. (1994); Yin (2000); Koh (2007); Wang et al. (2005). However, there does not seem to be too many studies on bilevel multi-objective optimization.

A recent study by Eichfelder Eichfelder (2007) suggested a refinement based strategy in which the algorithm starts with a uniformly distributed set of points on $\mathbf{x}_u$. Thereafter, for each $\mathbf{x}_u$ vector, the lower level Pareto-optimal solutions are found using a classical generating based optimization method. The set of such points obtained are said to be an approximation of the induced set. Non-dominated points with respect to the upper level problem are chosen from this set and they provide an approximate idea of the desired upper level Pareto-optimal front. Now, the chosen $\mathbf{x}_u$ vectors are refined in their vicinities and the lower level optimizations are repeated till a good approximation of the Pareto-optimal front is obtained. The difficulty with such a technique is that if the dimension of $\mathbf{x}_u$ is high, generating a uniformly spread $\mathbf{x}_u$ vectors and refining the resulting $\mathbf{x}_u$ vector will be computationally expensive. Definitely, an optimization algorithm for simultaneous selection of $\mathbf{x}_u$ and corresponding optimal $\mathbf{x}_l$ vectors will be more effective.

The greatest challenge in handling bilevel optimization problems seems to lie in the fact that unless a solution is optimal for the lower level problem, it cannot be feasible for the overall problem. This requirement, in some sense disallow any approximate optimization algorithm (including an EA or an EMO) to be used for solving the lower level task. But from all practical point of view near-optimal or near-Pareto-optimal solutions are often acceptable and it is in this spirit for which EA and EMO may have a great potential for solving bilevel optimization problems. EA or EMO has another advantage. Unlike the classical point-by-point approach, EA/EMO uses a population of points in their operation. By keeping two interacting populations, a coevolutionary algorithm can be developed so that instead of a serial and complete optimization of lower level problem for every upper level solution, both upper and lower level optimization tasks can be pursued simultaneously through iterations. In the following, we suggest one such procedure.

4 Test Problems

4.1 Existing Test Problems

4.1.1 Problem 1

Problem 1 has a total of three variables with x_1, x_2 belonging to $\mathbf{x}_l$ and y belonging to x_u and is taken from Eichfelder (2007):

$$\begin{aligned}
\text{minimize} \quad & \mathbf{F}(\mathbf{x}) = \left\{ \begin{array}{c} x_1 - y \\ x_2 \end{array} \right\}, \\
\text{subject to} \quad & (x_1, x_2) \in \text{argmin}_{(x_1,x_2)} \left\{ \mathbf{f}(\mathbf{x}) = \left(\begin{array}{c} x_1 \\ x_2 \end{array} \right) \middle| g_1(\mathbf{x}) = y^2 - x_1^2 - x_2^2 \geq 0 \right\}, \\
& G_1(\mathbf{x}) = 1 + x_1 + x_2 \geq 0, \\
& -1 \leq x_1, x_2 \leq 1, \quad 0 \leq y \leq 1.
\end{aligned} \tag{2}$$

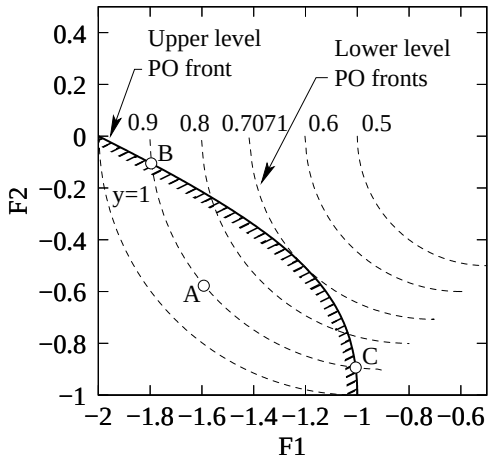


Figure 1: Pareto-optimal fronts of upper level (complete problem) and some representative lower level optimization tasks are shown for problem 1.

Both the lower and the upper level optimization tasks have two objectives each. A little consideration will reveal that for a fixed y value, the feasible region of the lower-level problem is the area inside a circle with center at origin ($x_1 = x_2 = 0$) and radius equal to y. The Pareto-optimal set for the lower-level optimization task for a fixed y is the bottom-left quarter of the circle:

$$\{(x_1, x_2) \in \mathbf{R}^2 \quad | \quad x_1^2 + x_2^2 = y^2, x_1 \leq 0, x_2 \leq 0\}.$$

The linear constraint in the upper level optimization task does not allow the entire quarter circle to be feasible for some y. Thus, at most a couple of points from the quarter circle belongs to the Pareto-optimal set of the overall problem. Eichfelder Eichfelder (2007) reported the following Pareto-optimal solutions for this problem:

$$\mathbf{x}^* = \left\{ (x_1, x_2, y) \in \mathbf{R}^3 \quad | \quad x_1 = -1 - x_2, x_2 = -\frac{1}{2} \pm \frac{1}{4}\sqrt{8y^2 - 4}, y \in \left[\frac{1}{\sqrt{2}}, 1 \right] \right\}. \tag{3}$$

The Pareto-optimal front in F_1-F_2 space is given in parametric form, as follows:

$$\left\{ (F_1, F_2) \in \mathbf{R}^2 \quad | \quad F_1 = -1 - F_2 - t, F_2 = -\frac{1}{2} \pm \frac{1}{4}\sqrt{8t^2 - 4}, t \in \left[\frac{1}{\sqrt{2}}, 1 \right] \right\}. \tag{4}$$

Figure 1 shows the Pareto-optimal front of problem 1. Lower level Pareto-optimal fronts of some representative y values are also shown on the figure, indicating that at most two such Pareto-optimal solutions (such as points B and C for $y = 0.9$) of a lower level optimization problem becomes the candidate Pareto-optimal solutions of the upper level problem. It is interesting to note that in this problem there exists a number of lower level Pareto-optimal solutions (such as solution A marked in the figure) which are infeasible to the upper level task. Thus,

if the lower level optimization is unable to find critical Pareto-optimal solutions (such as B or C) which correspond to the upper level Pareto-optimal solutions, but finds solutions like A in most occasions, the lower level task becomes useless. This makes the bilevel optimization task challenging and difficult.

4.1.2 Problem 2

Next, we create a simplistic bilevel two-objective optimization problem, having $\mathbf{x}_l = (x_1, x_2)$ and $\mathbf{x}_u = (y)$:

$$\begin{array}{ll} \text{minimize} & \mathbf{F}(\mathbf{x}) = \left\{ \begin{array}{c} (x_1-1)^2 + x_2^2 + y^2 \\ (x_1-1)^2 + x_2^2 + (y-1)^2 \end{array} \right\}, \\ \text{subject to} & (x_1, x_2) \in \text{argmin}_{(x_1,x_2)} \left\{ \mathbf{f}(\mathbf{x}) = \left(\begin{array}{c} x_1^2 + x_2^2 \\ (x_1-y)^2 + x_2^2 \end{array} \right) \right\}, \\ & -1 \leq x_1, x_2, y \leq 2. \end{array} \tag{5}$$

For a fixed value of y, the Pareto-optimal solutions of the lower level optimization problem are given as follows: $\{(x_1, x_2) \in \mathbf{R}^2 | x_1 \in [0, y], x_2 = 0\}$. For example, for $y = 0.75$, Figure 4 shows these solutions (points A through B) in the F_1-F_2 space. The points lie on a straight line and are not conflicting to each other. Thus, only one point (point A with $x_1 = y = 0.75$ and $x_2 = 0$) is a feasible solution to the upper level optimization task for a fixed $y = 0.75$. Interestingly, for a fixed y, the bottom-left boundary of the F_1-F_2 space corresponds to the upper bound of x_1 or $x_1 = 1$. However, solutions having $x_1 = 1$ till $x_1 = y$ are not Pareto-optimal for the overall problem. For $y = 0.75$, solutions on line CA (excluding A) are not Pareto-optimal to both lower and upper level problems. Similarly solutions from B upwards on the '$y = 0.75$' line are also not Pareto-optimal for both levels.

When we plot all solutions for which $x_1 = y$ and $x_2 = 0$, we obtain the dotted line marked with '$x_1 = y$' in the figure. Different lower level Pareto-optimal fronts (for different y values) are shown in the figure with dashed straight lines. It is interesting to note that all solutions on this '$x_1 = y$' curve are not Pareto-optimal to the overall problem. By investigating the figure, we observe that the Pareto-optimal solutions to the upper-level problem corresponds to following solutions: $\{(x_1, x_2, y) \in \mathbf{R}^3 | x_1 = y, x_2 = 0, y \in [0.5, 1.0]\}$. This problem does not have any constraint in its lower or upper level. If an algorithm fails to find true Pareto-optimal solutions of a lower level problem and ends up finding a solution below the '$x_1 = y$' curve, such as solution C, it can potentially dominate a true Pareto-optimal point (such as point A) thereby making the task of finding true Pareto-optimal solutions difficult.

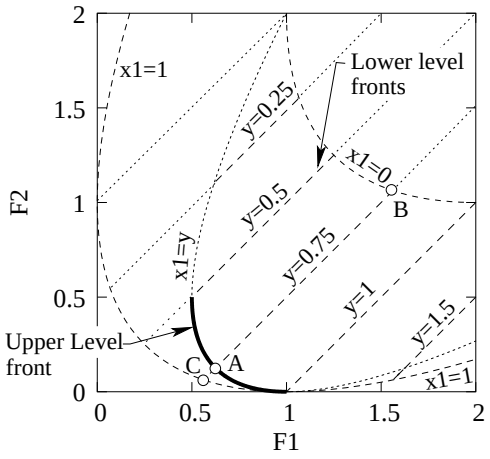


Figure 2: Pareto-optimal fronts of upper level (complete problem) and some representative lower level optimization tasks are shown for problem 1.

5 Principle of Bilevel Test Problem Development

Bilevel multi-objective optimization problems are different from single-level multi-objective optimization problems in that the Pareto-optimal solutions for the lower level multi-objective optimization problems are feasible solutions to the upper level problem. Thus, while developing a bilevel multi-objective test problem, we should have ways to test an algorithm's ability to handle complexities in both lower and upper level problems independently and collectively. In addition, we would also like to have have a knowledge of the exact location (and relationships) of Pareto-optimal points of each problem. We summarize in the following some properties which we would like to have in a bilevel multi-objective test problem:

1. Exact Pareto-optimal in both lower and upper level problems are easy to know. This will enable an user to evaluate the performance of an algorithm easily.

2. Problems are scalable with respect to number of variables. This will enable an user to test whether an algorithms scales well with number of variables in both lower and upper levels.

3. Problems are scalable with respect to number of objectives in both lower and upper level problems. This will enable an user to evaluate an algorithm's scalable performance with number of objectives.

4. Lower level problems are difficult to solve to Pareto-optimality. If lower level Pareto-optimal fronts are not found exactly, upper level solutions are not feasible. Therefore, these problems will test an algorithm's ability to converge to the correct Pareto-optimal front (we can borrow ideas here from single level EMO test problems). The shape (convex, non-convex, disjointedness) of the Pareto-optimal front plays an important role in this respect.

5. There is a conflict between lower and upper level problem solving tasks. In these problems, dominated lower level solutions may be allowed to dominate upper level Pareto-optimal points. These problems will test an algorithm's ability to converge to the lower level Pareto-optimal fronts, despite not enough emphasis by the upper level problem to do so.

6. Desirably, problems are scalable to multiple levels (at most up to three). On the other hand, it will be ideal to have the problems which degenerate to single level test problems as well.

7. Different lower level problems contribute differently to the upper level front. These test problems will test an algorithm's coordinating ability between upper and lower levels in finding the correct upper level Pareto-optimal front.

Various difference principles are possible to construct test problems following above guidelines. Here, we suggest a simple procedure for two-objective optimization problems in both lower and upper levels. Our approach is a bottom-up approach, in which both lower and upper level problems are designed starting with a pre-specified Pareto-optimal front. We begin the construction process from the lower level problem.

We choose a standard bi-objective test problem Zitzler et al. (2000); Deb et al. (2005) as a parameterized lower level problem here, which results in a parametric representation of the Pareto-optimal front $(f_1^*(t), f_2^*(t))$ for $t \in [t_{\min}, t_{\max}]$. For example, if ZDT1 is chosen, the Pareto-optimal front is given by: $(f_1^*(t) = t,\ f_2^*(t) = 1 - t^2)$ for $t \in [0, 1]$. Thus, the bi-objective front is always a one-parameter curve in the case of a bi-objective problem. Since all these solutions are feasible for the upper level problem and since only a subset (or none) of these solutions may qualify as the upper level Pareto-optimal solutions, we can now map these solutions to another one-dimensional non-dominated front in the upper level problem. We illustrate this aspect in Figure 3.

The lower level Pareto-optimal front (AB) in the f_1-f_2 space is mapped to a complete circle (A′B′) in the F_1-F_2 space. Different lower level Pareto-optimal fronts are mapped and placed at different locations on the upper level front by using a parameter v. The figure shows some such mappings at a few locations with circles. Many simple ideas for the mapping are possible and the figure demonstrates one such simple mapping.

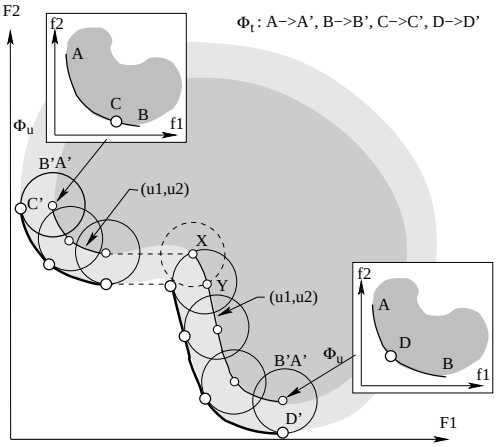


Figure 3: The construction principle of bilevel test problems.

Next, we choose a parameterized, trade-off relationship between two terms u_1 and u_2 as a function of v in the upper level objective space, such that (u_1^*, u_2^*) corresponds to the non-dominated relationship among u_1 and u_2. Many different values of v are possible and for every v, the lower level Pareto-optimal front is $(f_1(t(v)), f_2(t(v)))$ for $t \in [t_{\min}(v), t_{\max}(v)]$. These lower level solutions get mapped using ($\Phi_u : (f_1, f_2) \rightarrow (u_1, u_2)$) on a circle to the upper level objective space having the point v (on u_1-u_2 curve) as the center with a pre-specified radius. Another mapping Φ_t is used to map the parameter t to t'. The following is commutative diagram of both mappings:

$$\begin{array}{ccc} t(v) & \longrightarrow & (f_1^*, f_2^*)(v) \\ {\scriptstyle \Phi_t}\downarrow & & \downarrow{\scriptstyle \Phi_u} \\ t'(v) & \longrightarrow & (F_1, F_2)(v) \end{array} \tag{6}$$

The final upper level Pareto-optimal front can then be derived with the non-dominated solutions from all circles in the F_1-F_2 space, or mathematically

$$(F_1^*, F_2^*) = \text{ND}\left[\cup_{v \in \text{argmin}_{(v)}(u_1,u_2)}(F_1, F_2)(v)\right]. \tag{7}$$

Figure 3 depicts one such scenario, in which the lower level front is convex. The mapping Φ_u is such that the lower level Pareto-optimal points (A to B) in f_1-f_2 space is mapped to a full circle (A′ to B′) on the F_1-F_2 space. Thus only a portion of the lower level Pareto-optimal points are non-dominated in the upper level. The mapping Φ_t is chosen to be linear here (linearly mapping AB to to A′B′), but any other mapping can also be chosen to vary the density of points on the non-dominated part of the front in the upper level. The upper level trade-off information on v is non-convex (marked as u_1-u_2 relationship) in the example problem. It is now clear that every upper level Pareto-optimal front comes from a different v and is a different point in the corresponding lower level Pareto-optimal front. Interestingly, some non-dominated v values (in the range XY) do not lead to upper level Pareto-optimal front.

To solve the above problem to upper level Pareto-optimality, an algorithm must first find the exact lower level Pareto-optimal front for every value of the upper level variable v. The upper level optimization then must sort out and locate all possible v which contribute to the upper level Pareto-optimal front. Not only does the algorithm find all such v values, for each such value it should also preserve a well distributed set of lower level Pareto-optimal points

which are essential to span the upper level Pareto-optimal front. In the example of the figure, only one solution comes from each lower level front, but a different mapping Φ_u may cause a set of solutions from some lower level fronts to lie on the upper level Pareto-optimal front.

To make each level problem somewhat difficult, additional terms as a function of additional independent variables can be added in each level. The optimization algorithm must now find minimum values of the additional terms to be on the Pareto-optimal front. The approaches to such fronts can be made difficult by choosing suitable functional forms, thereby making the problem more difficult to solve.

5.1 Proposed Test Problems

5.1.1 Problem 3

The next problem has $2K$ variables with K real-valued variables each for $\mathbf{x}$ and $\mathbf{y}$:

$$
\begin{aligned}
&\text{minimize} \quad \mathbf{F}(\mathbf{x},\mathbf{y}) = \\
&\left\{ \begin{array}{l} (1+r-\cos(\alpha\pi y_1)) + \sum_{j=2}^{K}(y_j - \frac{j-1}{2})^2 \\ \quad +\tau\sum_{i=2}^{K}(x_i-y_i)^2 - r\cos\left(\frac{\pi}{2}\frac{x_1}{y_1}\right) \\ (1+r-\sin(\alpha\pi y_1)) + \sum_{j=2}^{K}(y_j - \frac{j-1}{2})^2 \\ \quad +\tau\sum_{i=2}^{K}(x_i-y_i)^2 - r\sin\left(\frac{\pi}{2}\frac{x_1}{y_1}\right) \end{array} \right\}, \\
&\text{subject to} \quad (\mathbf{x}) \in \text{argmin}_{(\mathbf{x})} \\
&\left\{ \mathbf{f}(\mathbf{x}) = \left(\begin{array}{l} x_1^2 + \sum_{i=2}^{K}(x_i-y_i)^2 \\ \quad +\sum_{i=2}^{K} 10(1-\cos(4\pi(x_i-y_i))) \\ \sum_{i=1}^{K}(x_i-y_i)^2 \\ \quad +\sum_{i=2}^{K} 10|\sin(4\pi(x_i-y_i)| \end{array} \right) \right\}, \\
&-K \le x_i \le K, \quad \text{for } i = 1,\ldots,K, \\
&1 \le y_1 \le 4, \quad -K \le y_j \le K, \quad \text{for } j = 2,\ldots,K.
\end{aligned} \tag{8}
$$

The lower level Pareto-optimal front for a given $\mathbf{y}$ vector corresponds to $x_i = y_i$ for $i = 2,\ldots,K$ and $x_1 \in [0, y_1]$. The objectives are related as follows: $f_2^* = (\sqrt{f_1^*} - y_1)^2$. Let us first consider $\tau = 1$. The upper level Pareto-optimal front corresponds to $y_i = (j-1)/2$ for $j = 2,\ldots,K$. The parametric functional relationship is $u_1 = 1 + r - \cos(\alpha\pi y_1)$ and $u_2 = 1 + r - \sin(\alpha\pi y_1)$. This is a circle of radius one and center at $((1+r),(1+r))$ in the F_1-F_2 space. Thus, the non-dominated portion is the third quadrant of this circle and this happens for $y_1 \in (2p+[0,0.5])/\alpha$, where p is an integer including zero. For $\alpha = 1$ and for $1 \le y_1 \le 4$, this happens for $y_1 \in [2, 2.5]$ and $y_1 = 4$. Accumulating the non-dominated portions of all circles of radius r at every optimal y_1, we have the overall upper level Pareto-optimal front defined as a circle of radius $(1+r)$ and center at $((1+r),(1+r))$, as shown in Figure 4.

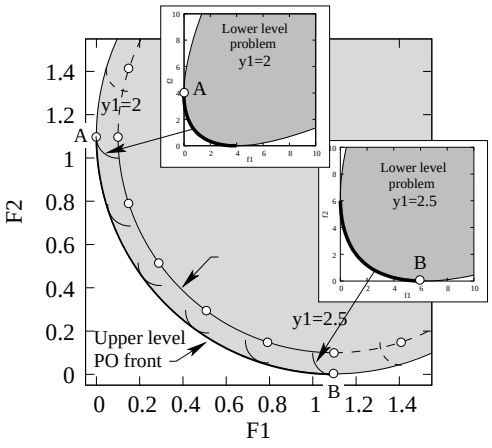


Figure 4: Pareto-optimal front for problem 3.

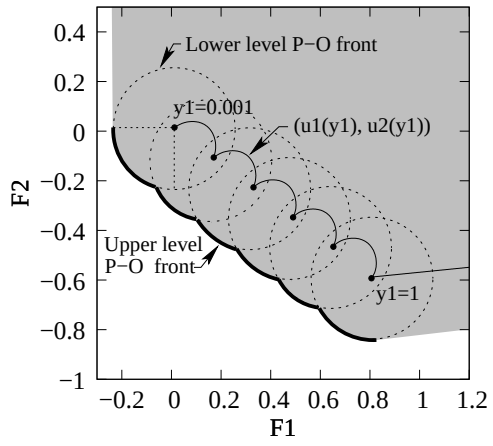


Figure 5: Pareto-optimal front for problem 4.

For this test problem, we use $K = 3$ (6-variable problem), $r = 0.1$, and $\alpha = 1$. By increasing K the number of variables can be increased without affecting the Pareto-optimal frontier. The effect of r is not that significant for a real-valued variable y_1. However, if a discrete y_1 is chosen, an increase in r will have a range of lower level Pareto-optimal solutions to qualify as upper level Pareto-optimal solutions. Also, an increase in α causes multi-modal y_1 solutions to appear on the upper level Pareto-optimal front. For example, for $\alpha = 2$, $y_1 \in [1, 1.25] \cup [2, 2.25] \cup [3, 3.25]$ will correspond to the upper level Pareto-optimal front.

An interesting scenario happens when $\tau = -1$ is set. Now, if any $\mathbf{x}$ is not Pareto-optimal to a lower level problem, some positive amount is deducted in the upper level objectives, thereby allowing points to exist below the Pareto-optimal curve of the upper level problem. These points are infeasible to the upper level due to their non-optimal property at the lower level problems and if allowed to exist in the upper level will dominate the true upper level Pareto-optimal points. Thus, this problem will be difficult to solve than the problem with $\tau = 1$, as the upper level problem does not motivate the solutions to become Pareto-optimal at the lower level and the overall procedure will end up finding infeasible solutions.

5.1.2 Problem 4

The next problem consists of $2K$ real-valued variables, but the upper level Pareto-optimal front corresponds to a few values of y_1:

$$\begin{aligned}
&\text{minimize} \quad \mathbf{F}(\mathbf{x}, \mathbf{y}) = \\
&\left\{ \begin{array}{l} u_1(y_1) + \sum_{j=2}^{K} \left[y_j^2 + 10(1 - \cos(4\pi y_i)) \right] \\ \quad -r \cos\left(2\pi \frac{x_1}{y_1} \right) \\ u_2(y_1) + \sum_{j=2}^{K} \left[y_j^2 + 10(1 - \cos(4\pi y_i)) \right] \\ \quad -r \sin\left(2\pi \frac{x_1}{y_1} \right) \end{array} \right\}, \\
&\text{subject to} \quad (\mathbf{x}) \in \\
&\quad \text{argmin}_{(\mathbf{x})} \left\{ \mathbf{f}(\mathbf{x}) = \left(\begin{array}{c} x_1^2 + \sum_{i=2}^{K} (x_i - y_i)^2 \\ \sum_{i=1}^{K} i(x_i - y_i)^2 \end{array} \right) \right\}, \\
&\quad -K \le x_i \le K, \quad \text{for } i = 1, \ldots, K, \\
&\quad 0.001 \le y_1 \le K, \quad -K \le y_j \le K, \quad \text{for } j = 2, \ldots, K,
\end{aligned} \tag{9}$$

where

$$u_1(y_1) = \left\{ \begin{array}{l} \cos(0.2\pi) y_1 + \sin(0.2\pi) \sqrt{|0.02 \sin(5\pi y_1)|}, \\ \quad \text{for } 0 \le y_1 \le 1, \\ y_1 - (1 - \cos(0.2\pi)), \quad y_1 > 1 \end{array} \right.$$

$$u_2(y_1) = \left\{ \begin{array}{l} -\sin(0.2\pi) y_1 + \cos(0.2\pi) \sqrt{|0.02 \sin(5\pi y_1)|}, \\ \quad \text{for } 0 \le y_1 \le 1, \\ 0.1(y_1 - 1) - \sin(0.2\pi), \quad \text{for } y_1 > 1. \end{array} \right. \tag{10}$$

For every $\mathbf{y}$ vector, the lower level front occurs for $x_i = y_i$ for $i = 2, \ldots, K$ and $x_1 \in [0, y_1]$ and the functional relationship is $f_2^* = (\sqrt{f_1^*} - y_1)^2$. Although a number of different values of y_1 correspond to a non-dominated trade-off with respect to (u_1, u_2), the upper level Pareto-optimal front corresponds to only six discrete values of y_1 (=0.001, 0.2, 0.4, 0.6, 0.8 and 1). Here we suggest using $r = 0.25$. In this problem, a lower level front is mapped to a circle which is centered at $(u_1(y_1), u_2(y_1))$ and has a radius of 0.25. The resulting Pareto-optimal front with six circular arcs are shown in Figure 5. It is clear that the upper level Pareto-optimal front is constructed only from different non-dominated regions from each of the six y_1 values. Interestingly, despite y_1 taking any real value within $[0.001, K]$, no other y_1 value contributes to the upper level Pareto-optimal frontier.

Like in Problem 3, we can add the term $\tau \sum_{i=2}^{K}(x_i - y_i)^2$ on both F_1 and F_2 with $\tau = -1$ in order to make the problem more difficult.

5.1.3 Problem 5

The next problem has a discrete variable y_1 and other variables are all real-valued:

$$
\begin{aligned}
&\text{minimize} \quad \mathbf{F}(\mathbf{x}, \mathbf{y}) = \\
&\left\{ \begin{array}{l} y_1 + \sum_{j=3}^{K}(y_j - j/2)^2 - R(y_1)\cos(4\tan^{-1}\left(\frac{y_2 - x_2}{y_1 - x_1}\right) \\ y_2 + \sum_{j=3}^{K}(y_j - j/2)^2 - R(y_1)\sin(4\tan^{-1}\left(\frac{y_2 - x_2}{y_1 - x_1}\right) \end{array} \right\}, \\
&\text{subject to} \quad (\mathbf{x}) \in \text{argmin}_{(\mathbf{x})} \\
&\left\{ \mathbf{f}(\mathbf{x}) = \left(\begin{array}{l} x_1 + \sum_{i=3}^{K}(x_i - y_i)^2 \\ x_2 + \sum_{i=3}^{K}(x_i - y_i)^2 \end{array} \right) \middle| \right. \\
&\qquad \left. g_1(\mathbf{x}) = (x_1 - y_1)^2 + (x_2 - y_2)^2 \leq r^2 \right\}, \\
&G(\mathbf{y}) = y_2 - (1 - y_1^2) \geq 0, \\
&-K \leq x_i \leq K, \quad \text{for } i = 1, \ldots, K, \\
&0 \leq y_j \leq K, \quad \text{for } j = 1, \ldots, K, \\
&y_1 \text{ is a multiple of 0.1.}
\end{aligned}
\tag{11}
$$

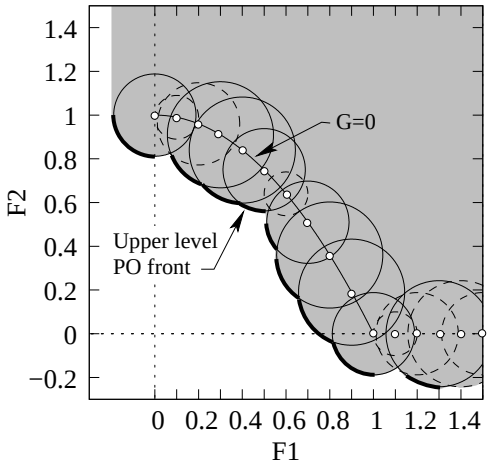


Figure 6: Pareto-optimal front for problem 5.

Here we suggest a periodically changing radius: $R(y_1) = 0.1 + 0.15|\sin(2\pi(y_1 - 0.1)|$ and $r = 0.2$. The corresponding Pareto-optimal solutions for the lower level problem are given as $\{(\mathbf{x}) \in \mathbf{R}^K | (x_1 - y_1)^2 + (x_2 - y_2)^2 = r^2, \quad x_1 \leq y_1, x_2 \leq y_2, x_i = y_i, \forall i \geq 3\}$

For the upper level Pareto-optimal points, $y_i = j/2$ for $j \leq 3$. The variables y_1 and y_2 take values satisfying constraint $G(\mathbf{y}) = 0$. For each such combination, variables x_1 and x_2 take values according to satisfying equation **??** and on the third quadrant of the circle in the F_1-F_2 space. Notice in Figure 6 how lower level Pareto-optimal solutions for $y_1 = 0.1$ and 0.2 mapped to corresponding circles in the upper level problem get dominated by that for $y_1 = 0$ and 0.3. Every lower level front has an unequal contribution to the upper level Pareto-optimal front.

Like before, we can add the term $\tau \sum_{i=3}^{K}(x_i - y_i)^2$ (with $\tau = -1$) on both objectives F_1 and F_2 to make the problem more difficult. It is interesting to note that although lower level dominated solutions may lie below the upper level Pareto-optimal front those solutions may still satisfy the upper level constraint $G()$. Thus, a negative value of τ will test an algorithm's ability to converge to the Pareto-optimal front of a lower level problem, despite the apparent emphasis of dominated solutions of the lower level problem by the upper level problem.

5.1.4 Problem 6

There are $K + L + 1$ variables in this problem.

$$
\begin{aligned}
&\text{minimize} \quad \mathbf{F}(\mathbf{x},\mathbf{y}) = \\
&\left\{ \begin{array}{l} (1-x_1)(1+\sum_{j=2}^{K} x_j^2)y_1 \\ x_1(1+\sum_{j=2}^{K} x_j^2)y_1 \end{array} \right\}, \\
&\text{subject to} \quad (\mathbf{x}) \in \text{argmin}_{(\mathbf{x})} \\
&\left\{ \mathbf{f}(\mathbf{x}) = \left(\begin{array}{l} (1-x_1)(1+\sum_{j=K+1}^{K+L} x_j^2)y_1 \\ x_1(1+\sum_{j=K+1}^{K+L} x_j^2)y_1 \end{array} \right) \right| \\
&\qquad g_1(\mathbf{x}) = (1-x_1)y_1 + \tfrac{1}{2}x_1y_1 - 1 \geq 0 \Big\}, \\
&-1 \leq x_1 \leq 1, \quad 1 \leq y_1 \leq 2, \\
&-(K+L) \leq x_i \leq (K+L), \quad \text{for } i = 2,\ldots,(K+L).
\end{aligned}
\tag{12}
$$

For a fixed y_1, the lower level Pareto-optimal front corresponds to the line $f_1 + f_2 = y_1$, as shown in Figure 7. The front is restricted at one end by the constraint and at the other end by x-axis. The constraint boundary does not allow the lower lower level front to continue further which otherwise would have extended until it intercepts the y-axis. At the lower level front, $x_i = 0$ for $i = K+1,\ldots,K+L$ and $x_1 \in [0, 2(1-\frac{1}{y_1})]$, the upper bound depends on the intersection point of the lower level front and the constraint boundary.

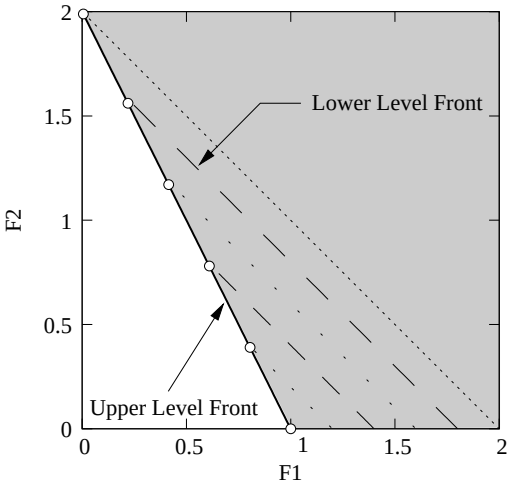


Figure 7: Pareto-optimal front for problem 6.

The functional relationship for the upper level front is $F_1 + \frac{1}{2}F_2 = 1$, which is the constraint boundary. For the upper level front $x_i = 0$ for $i = 2,\ldots,K$, every Pareto-optimal point corresponds to a different y_1 value in $[1,2]$ and a different x_1 value in $[0,1]$. By increasing K, the number of variables can be increased at the upper level and by increasing L the number of variables can be increased at the lower level. Thus, the complexity at each level can be increased independently by varying K and L. For this test problem we have taken $K = 3$ and $L = 2$ (6-variable test problem).

5.1.5 Problem 7

There are $K + L + 1$ variables in this problem.

$$
\begin{aligned}
&\text{minimize} \quad \mathbf{F}(\mathbf{x}, \mathbf{y}) = \\
&\left\{ \begin{array}{l} (1 - x_1)(1 + \sum_{j=2}^{K} x_j^2) y_1 \\ x_1 (1 + \sum_{j=2}^{K} x_j^2) y_1 \end{array} \right\}, \\
&\text{subject to} \quad (\mathbf{x}) \in \text{argmin}_{(\mathbf{x})} \\
&\left\{ \mathbf{f}(\mathbf{x}) = \left(\begin{array}{l} (1 - x_1)(1 + \sum_{j=K+1}^{K+L} x_j^2) y_1 \\ x_1 (1 + \sum_{j=K+1}^{K+L} x_j^2) y_1 \end{array} \right) \right| \\
&\qquad g_1(\mathbf{x}) = (1 - x_1) y_1 + \tfrac{1}{2} x_1 y_1 - 2 + \tfrac{1}{5} \left[5(1 - x_1) y_1 \right. \\
&\qquad \left. +0.2 \right] \geq 0 \Big\}, [\cdot] \text{ denotes greatest integer function,} \\
&-1 \leq x_1 \leq 1, \quad 1 \leq y_1 \leq 2, \\
&-(K + L) \leq x_i \leq (K + L), \quad \text{for } i = 2, \ldots, (K + L).
\end{aligned}
\tag{13}
$$

This test problem is similar to the test problem 1 except the constraint at the lower level. The lower level constraint in this test problem is a discontinuous step funtion which allows multiple points from each lower level front to enter the upper level front. In this problem also the lower level front is given by $f_1 + f_2 = y_1$. At the lower level front $x_i = 0$ for $i = K + 1, \ldots, K + L$ and $x_1 \in [0, a]$, where $a \in \{0.2, 0.4, 0.6, 0.8, 1\}$. The upper bound ($a$) of x_1 is determined by the intersection point of the lower level front and the constraint boundary.

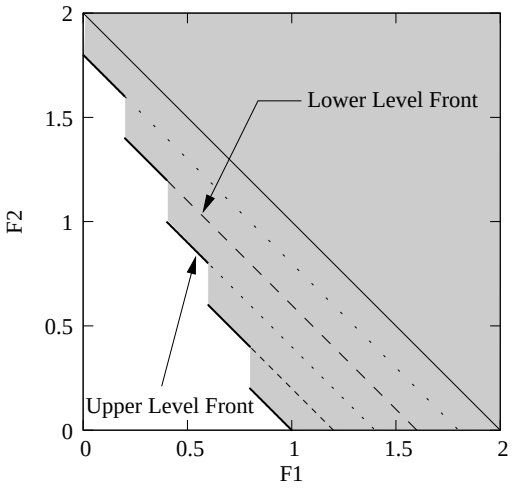


Figure 8: Pareto-optimal front for problem 7.

The functional relationship for the upper level front is $F_1 + \frac{1}{2} F_2 = 2 - \frac{1}{5} [F_1 + 0.2]$ which is the constraint boundary. For the upper level front, $x_i = 0$ for $i = 2, \ldots, K$ and every Pareto-optimal point corresponds to $y_1 \in \{0, 0.2, 0.4, 0.6, 0.8\}$ and a different x_1 value in $[0, 1]$. For this test problem we have taken $K = 3$ and $L = 2$ (6-variable test problem).

6 Existing Classical and Evolutionary Methodologies

7 Proposed Bilevel Evolutionary Multi-Objective Optimization

The proposed method uses the elitist non-dominated sorting GA or NSGA-II (Deb et al. (2002)), however any other EMO procedures can also be used instead. The upper level population (of size N_u) uses NSGA-II operations for a maximum of T_u generations or until the hypervolume based stopping criteria is met, with upper level objectives ($\mathbf{F}$) and constraints ($\mathbf{G}$) in determining non-dominated rank and crowding distance values of each population member.

The upper level execution calls a lower level NSGA-II simulation with a population size of N_l for T_l generations. Thereafter, a local search is done over a few (discussed later) rank one members of lower level until the lower level KKT criterion is met within a specified accuracy. The optimal members of the lower level undergoing local search are assigned an optimality tag

and become eligible to enter the archive at the upper level if it is non-dominated when compared to the existing members in the archive. The dominated members in the archive are eliminated.

The population has n_s subpopulations of size N_l each at the beginning. It is ensured that all the members in a subpopulation (or a lower level population) have the same $\mathbf{x}_u$. Initially, the population size of the lower level is set depending upon the lower and upper level variables, i.e. $\mathbf{x}_l$ and $\mathbf{x}_u$ respectively. The size of the subpopulation, N_l and the number of subpopulations, n_s is determined from the following equations:

$$\begin{array}{l} \frac{n_s^{min}}{N_l^{max}} = \frac{dim(\mathbf{x}_u)}{dim(\mathbf{x}_l)} \\ n_s^{min} N_l^{max} = N_u \\ \implies n_s^{min} = \sqrt{\frac{dim(\mathbf{x}_u)}{dim(\mathbf{x}_l)} N_u} \\ \text{and} \quad N_l^{max} = \sqrt{\frac{N_u}{\frac{dim(\mathbf{x}_u)}{dim(\mathbf{x}_l)}}} \end{array} \tag{14}$$

The above values are set for the first generation and then onwards the lower level population size is adjusted depending upon the proximity of $\mathbf{x}_u$ to the members in the archive. The maximum euclidean distance between the $\mathbf{x}_u$ vectors from the archive set is calculated and stored in δ_Y. For a generated $\mathbf{x}_u$ the eucleadean distance (only $\mathbf{x}_u$ is used to calculate the distance) from the closest archive member is stored in δ_y. The subpopulation size for the subsequent generations is calculated as follows:

$$\begin{array}{l} N_l = Integer(\frac{\delta_y}{\delta_Y} N_l^{max}) \\ \text{if} \quad N_l < 4 \quad \text{then} \quad N_l = 4 \\ \text{if} \quad N_l > N_l^{max} \quad \text{then} \quad N_l = N_l^{max} \end{array} \tag{15}$$

A hypervolume metric is calculated at the lower level to terminate the lower level run. Maximum (H_{vol}^{max}) and Minimum (H_{vol}^{min}) hypervolume is calculated from the lower level non-dominated set in every 10 generations of lower level run and the run is terminated when the following criteria is met.

$$\begin{array}{l} H_{metric} = \frac{H_l^{max} - H_l^{min}}{H_l^{max} + H_l^{min}} \\ \text{terminate if} \quad H_{metric} < \epsilon_l \end{array} \tag{16}$$

The number of lower level generations needed to meet the above criterion is calculated for each subpopulation during the first upper level generation and an average is taken. This average is set as the maximum generations, T_l^{max} for all the subsequent lower level runs and the hypervolume metric is no more calculated at the lower level. The lower level subsequent generations are calculated using T_l^{max}, δ_y and δ_Y as follows:

$$\begin{array}{l} T_l = Integer(\frac{\delta_y}{\delta_Y} T_l^{max}) \\ \text{if} \quad N_l > N_l^{max} \quad \text{then} \quad N_l = N_l^{max} \end{array} \tag{17}$$

The upper level population size N_u is fixed and is a user specified parameter. To start the proposed BLEMO, we create all solutions at random, but maintain the structure of having same $\mathbf{x}_u$ in a particular subpopulation. From thereon, the proposed operations ensure that the above-mentioned structure is maintained from one generation to another. In the following, we describe one iteration of the proposed BLEMO procedure. At the start of the upper level NSGA-II generation t, we have a population P_t of size N_u. Every population member has the following quantities computed from the previous iteration: (i) a non-dominated rank ND_u corresponding to $\mathbf{F}$ and $\mathbf{G}$, (ii) a crowding distance value CD_u corresponding to $\mathbf{F}$, (iii) a non-dominated rank

ND_l corresponding to $\mathbf{f}$and $\mathbf{g}$, and (iv) a crowding distance value CD_l using $\mathbf{f}$. In addition to these, for the members stored in the archive, we have (v) a crowding distance value CD_a corresponding to $\mathbf{F}$and (vi) a non-dominated rank ND_a corresponding to $\mathbf{F}$ and $\mathbf{G}$.

7.1 Step-by-Step Procedure

Step 1: Apply a pair of binary tournament selections on members ($\mathbf{x} = (\mathbf{x}_u, \mathbf{x}_l)$) of P_t using ND_u and CD_u lexicographically. Also, apply a pair of binary tournament selections, using ND_a and CD_a lexicographically, on members randomly chosen from the archive A_t. Choose two such winners from A_t and two winners from P_t. The members from A_t participate as parents with a probability of $\frac{|A_t|}{|A_t|+|P_t|}$ otherwise the members from P_t become the parents for crossover. The upper level variable vectors $\mathbf{x}_u$ of two selected parents are then recombined using the SBX operator (Deb and Agrawal (1995)) to obtain two new vectors of which one is chosen at random. The chosen solution is mutated by the polynomial mutation operator with a probability of $1/|\mathbf{x}_u|$ (Deb (2001)) to obtain a child vector (say, $\mathbf{x}_u^{(1)}$).

The lower level population size for the vector $\mathbf{x}_u^{(1)}$ is calculated as suggested before. If the lower level population size is equal to N_l^{max} then a new subpopulation is created. N_l^{max} new lower level variable vectors $\mathbf{x}_l^{(i)}$ by applying selection-recombination-mutation operations on members of P_t and A_t choosing a random member from P_t with a probability of $\frac{|A_t|}{|A_t|+|P_t|}$ else accepting a random member from A_t. Thereafter, N_l child solutions are created by concatenating upper and lower level variables vectors together, as follows: $c_i = (\mathbf{x}_u^{(1)}, \mathbf{x}_l^{(i)})$ for $i = 1, \ldots, N_l$. Thus, for the new upper level variable vector, a subpopulation of N_l^{max} lower level variable vectors are created by genetic operations from P_t and A_t.

If the lower level population size is less than N_l^{max} then the member (say, $\mathbf{x}_u^{(a)}$) from the archive, closest to $\mathbf{x}_u^{(1)}$ is chosen. New lower level variable vectors in this case are not created instead the lower level variable vectors from the subpopulation to which $\mathbf{x}_u^{(a)}$ belongs is accepted as new lower level vectors for the subpopulation of $\mathbf{x}_u^{(1)}$. Child solutions are created by concatenating upper and lower level variables vectors together, as follows: $c_i = (\mathbf{x}_u^{(1)}, \mathbf{x}_l^{(a_i)})$ for $i = 1, \ldots, min(N_l, N_l^a$ where N_l^a is the subpopulation size to which $\mathbf{x}_u^{(a)}$ belongs. If $N_l^a < N_l$, then the remaining members ($N_l - N_l^a$) are created by the mechanism suggested in the previous paragraph.

Step 2: For each subpopulation, we now perform a NSGA-II procedure using lower level objectives ($\mathbf{f}$) and constraints ($\mathbf{g}$) for T_l generations. It is interesting to note that in each lower level NSGA-II, the upper level variable vector $\mathbf{x}_u$ is not changed. For every mating, the solutions are chosen as usual by the binary tournament selection using ND_l and CD_l lexicographically, if the subpopulation has no member which has already entered the archive. If the subpopulation has a member or members which have already entered the archive then only these solutions are chosen for mating using the mentioned binary tournament selection. The mutation is performed as usual. After the lower level NSGA-II simulation is performed for a subpopulation, the resulting solutions are marked with their non-dominated rank (ND_l) and crowding distance value (CD_l). From the members in the subpopulation which have $ND_l = 1$ and best ND_u, the ones which dominate any of the archive members or the ones with a distance less than $\delta_Y \frac{N_l}{N_l^{max}}$ from any of the archive members, are chosen for a SQP based local search. These members are assigned an optimal tag if they meet the specified KKT criterion. These members are then compared with the archive members. If these members are non-dominated when compared with the members in the archive then

they become eligible to be added into the archive. The dominated members in the archive are eliminated.

All the members from each subpopulation are then combined together in one population (the child population, Q_t). It is interesting to note that in Q_t, there are at least n_s members having $ND_l = 1$ (at least one coming from each subpopulation). Also, in Q_t, there are exactly n_s different $\mathbf{x}_u$ variable vectors.

Step 3: Each member of Q_t is now evaluated with $\mathbf{F}$ and $\mathbf{G}$. Populations P_t and Q_t are combined together to form R_t. The combined population R_t is then ranked according to non-domination and members within an identical non-dominated rank are assigned a crowding distance computed in the $\mathbf{F}$ space. Thus, each member of Q_t gets an upper level non-dominated rank ND_u and a crowding distance value CD_u.

Step 4: From the combined population R_t of size $2N_u$, half of its members are chosen in this step. First, the members of rank $ND_u = 1$ are considered. From them, solutions having $ND_l = 1$ are noted one by one in the order of reducing crowding distance CD_u, for each such solution the entire N_l subpopulation from its source population (either P_t or Q_t) is copied in an intermediate population S_t. If a subpopulation is already copied in S_t and a future solution from the same subpopulation is found to have $ND_u = ND_l = 1$, the subpopulation is not copied again. When all members of $ND_u = 1$ are considered, a similar consideration is continued with $ND_u = 2$ and so on till exactly n_s subpopulations are copied in S_t.

Step 5: Each subpopulation of S_t which are not the immediate offsprings of the current generation are modified using the lower level NSGA-II procedure applied with $\mathbf{f}$ and $\mathbf{g}$ for T_l generations. This step helps progress each lower level populations towards their individual Pareto-optimal frontiers.

Step 6: Finally, all subpopulations obtained after the lower level NSGA-II simulations are combined together to form the next generation population P_{t+1}.

The evaluation of the initial population is similar to the above. First, members of P_0 are created at random with n_s subpopulations of size N_l^{max} each. Every subpopulation has an identical $\mathbf{x}_u$ vector for all its members. Thereafter, each subpopulation is sent for an update of $\mathbf{x}_l$ vectors to the lower level NSGA-II (with $\mathbf{f}$ and $\mathbf{g}$). The lower level NSGA-II, in the first upper level generation, stops with a hypervolume based stopping criteria discussed before. Every member is assigned corresponding ND_l and CD_l values. The resulting subpopulations (from NSGA-II) are combined into one population (renamed as P_0) and evaluated using $\mathbf{F}$ and $\mathbf{G}$. Every member is then assigned a non-dominated rank ND_u and a crowding distance value CD_u.

The lower level optimal solutions of every subpoplation are saved in an archive (A_t). Initially, the archive A_0 is an empty set. Thereafter, at the end of every lower level run, solutions which have an optimal tag assigned to it are saved in the archive A_t. The non-dominated solutions (with $\mathbf{F}$ and $\mathbf{G}$) of the archive are kept in A_t and rest members are deleted from the archive.

The stopping criteria for the upper level is either the maximum number of generations specified by the user or the Hypervolume based metric calculated, this time with the objective functions of the upper level non dominated set. Maximum (H_{vol}^{max}) and Minimum (H_{vol}^{min}) hypervolume is calculated from the upper level non-dominated set in every 10 generations of upper level run and the run is terminated when the following criteria is met.

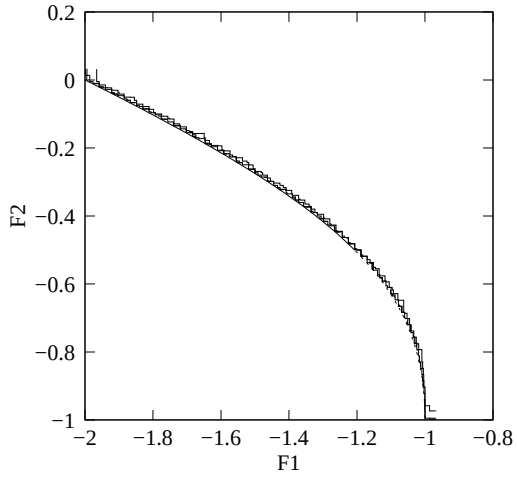


Figure 9: Attainment surfaces (100%, 50% and 0%) for problem 1 from 21 runs. The hypervolumes for the obtained attainment surfaces are 0.3583, 0.3678 and 0.3700

$$\begin{array}{l} H_{metric} = \frac{H_u^{max} - H_u^{min}}{H_u^{max} + H_u^{min}} \\ \text{terminate if} \quad H_{metric} < \epsilon_u \\ \text{or} \quad T_u = T_u^{max} \end{array} \tag{18}$$

7.2 Algorithmic Complexity

In the above BLEMO, the maximum number of function evaluations, until the termination criteria is met depends on N_u, T_u and T_l^{max}. The maximum number of function evaluations would be $N_u(2T_u + 1)(T_l^{max} + 1) + |SQP|$. Here $|SQP|$ is the number of function evaluations required for the Sequential Quadratic Programming method. It will be evident from the results below that the actual number of function evaluations required by the algorithm is much less when compared with the maximum.

Lower level NSGA-II is able to bring the members close to the pareto optimal front which makes the function evaluations required by SQP much less than the set maximum. Moreover the lower level NSGA-II generations is not T_l^{max} for each lower level call and can be as small as 1 depending upon the proximity of the upper level variable of the subpopulation to the closest upper level variable of an archive member. The given expression just represents an upper bound to the number of function evaluations and the evaluations would get close to the mentioned expression only in the case the algorithm is not able to find even a single feasible solution.

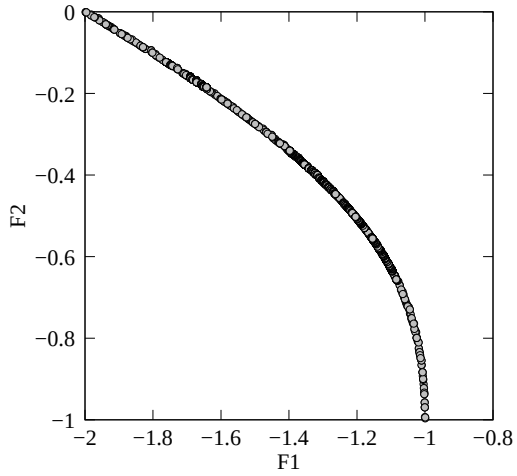


Figure 10: Best run for problem 1 from 21 runs. The hypervolume for the obtained non-dominated front is 0.3688

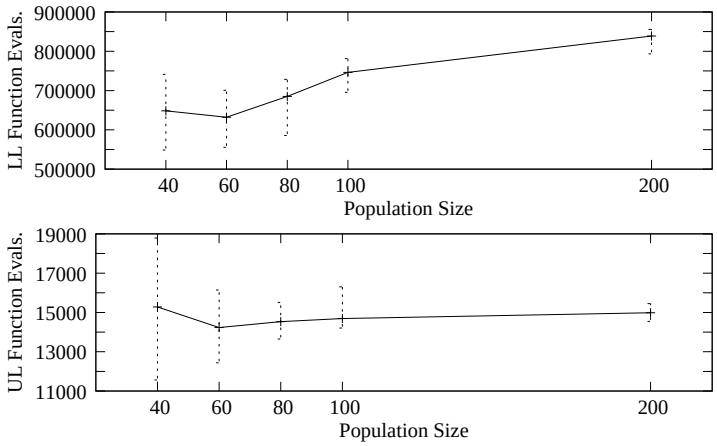


Figure 11: Average lower and upper level function evaluations with different population sizes for Problem 1. The average has been taken for 21 runs.

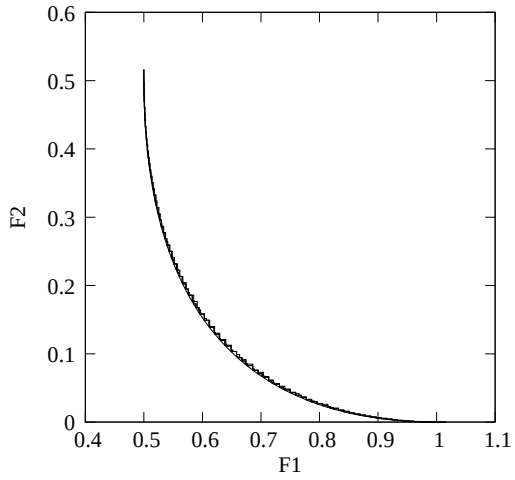


Figure 12: Attainment surfaces (100%, 50% and 0%) for problem 2 from 21 runs. The hypervolumes for the obtained attainment surfaces are 0.8561, 0.8582 and 0.8589

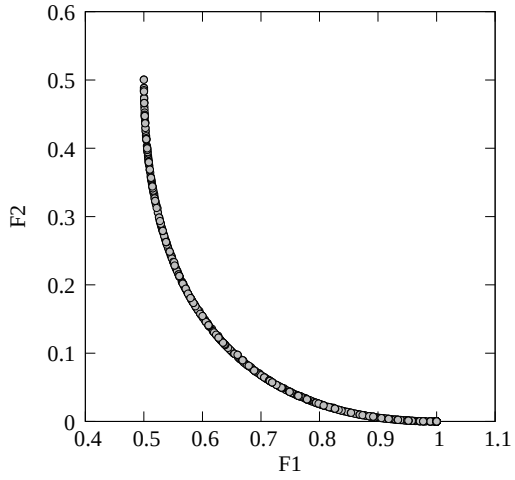


Figure 13: Best run for problem 2 from 21 runs. The hypervolume for the obtained non-dominated front is 0.8580

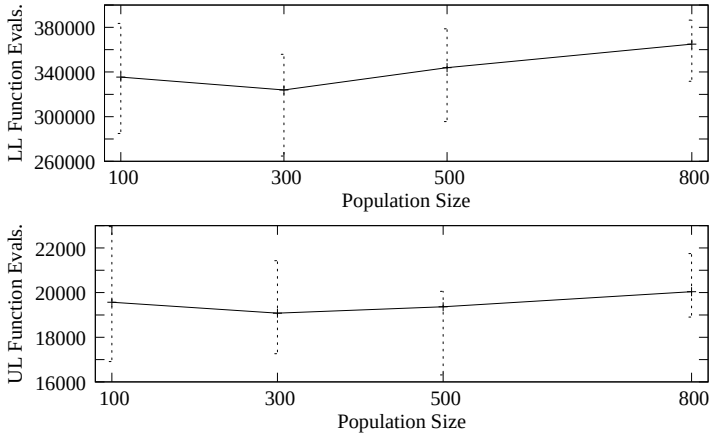


Figure 14: Average lower and upper level function evaluations with different population sizes for Problem 2. The average has been taken for 21 runs.

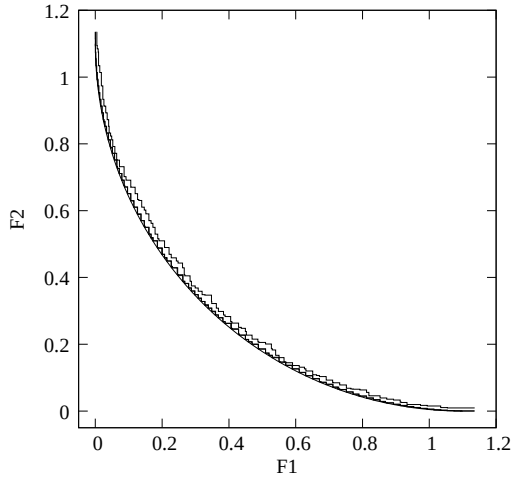


Figure 15: Attainment surfaces (100%, 50% and 0%) for problem 3 from 21 runs. The hypervolumes for the obtained attainment surfaces are 0.7812, 0.7984 and 0.7992

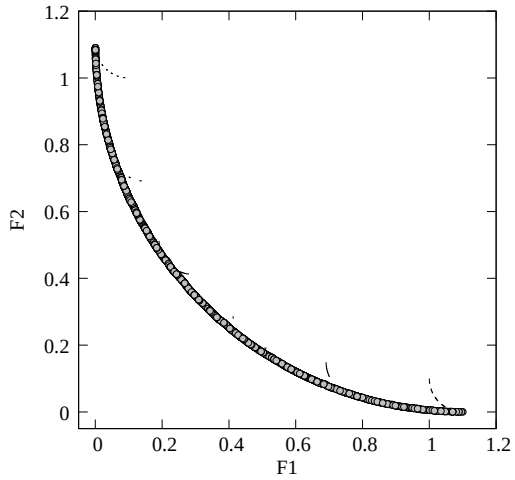


Figure 16: Best run for problem 3 from 21 runs. The hypervolume for the obtained non-dominated front is 0.7983

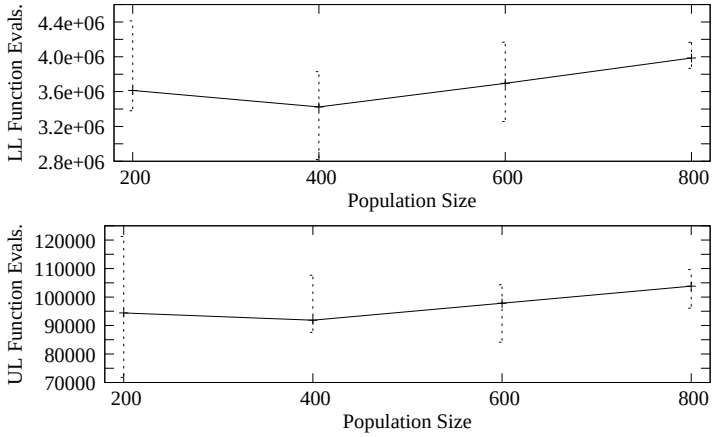


Figure 17: Average lower and upper level function evaluations with different population sizes for Problem 3. The average has been taken for 21 runs.

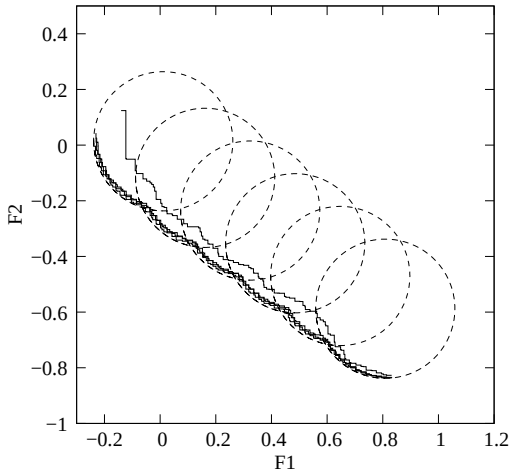


Figure 18: Attainment surfaces (100%, 75%, 50% and 0%) for problem 4 from 21 runs. The hypervolumes for the obtained attainment surfaces are 0.5776, 0.6492, 0.6542 and 0.6629

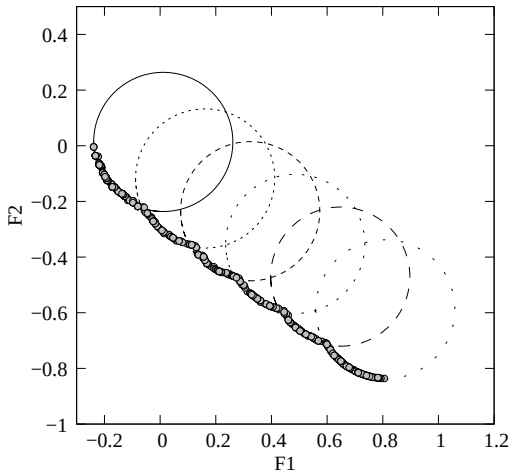


Figure 19: Best run for problem 4 from 21 runs. The hypervolume for the obtained non-dominated front is 0.6612

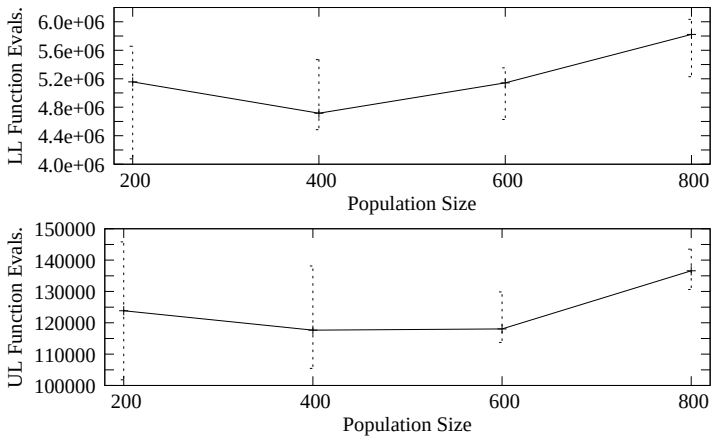


Figure 20: Average lower and upper level function evaluations with different population sizes for Problem 4. The average has been taken for 21 runs.

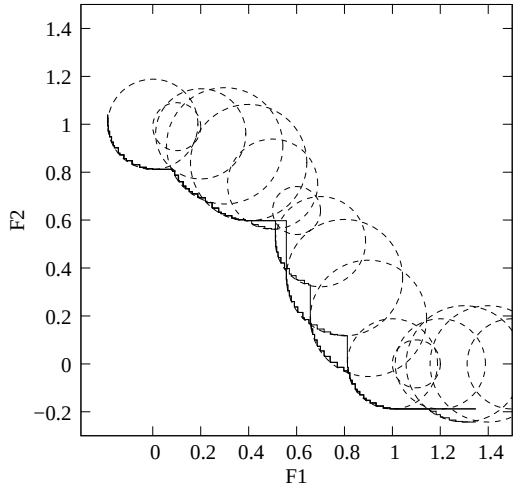


Figure 21: Attainment surfaces (100%, 50% and 0%) for problem 5 from 21 runs. The hypervolumes for the obtained attainment surfaces are 0.5528, 0.5705 and 0.5759

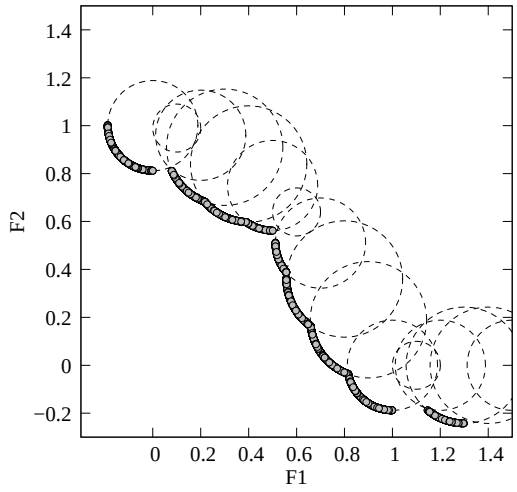


Figure 22: Best run for problem 5 from 21 runs. The hypervolume for the obtained non-dominated front is 0.5718

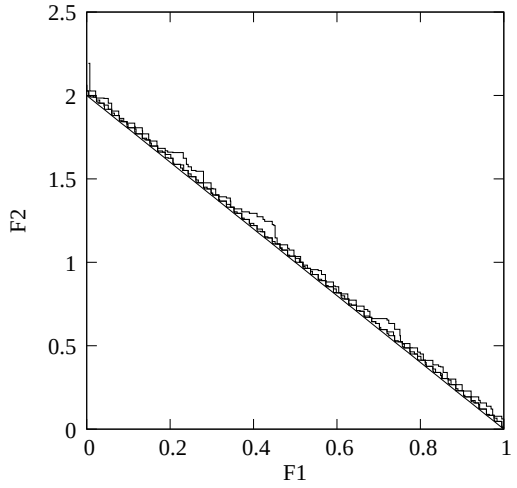


Figure 23: Attainment surfaces (100%, 50% and 0%) for problem 6 from 21 runs. The hypervolumes for the obtained attainment surfaces are 0.5077, 0.5241 and 0.5264

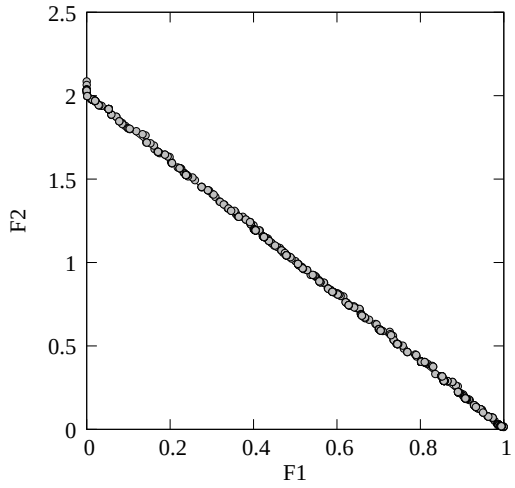


Figure 24: Best run for problem 6 from 21 runs. The hypervolume for the obtained non-dominated front is 0.5248

8 Results on Test Problems

8.1 Problem 1

8.2 Problem 2

8.3 Problem 3

8.4 Problem 4

8.5 Problem 5

8.6 Problem 6

8.7 Problem 7

9 Results on Other Numerical Problems

9.1 Problem 8

9.2 Problem 9

Table 1: SQP function evaluations for lower level.

Pr. No.	Lower Level Variables	Upper Level Variables	Population Size	Average SQP Evaluations	SQP Evaluations per Call
1	2	1	60	91382.16	19.03
2	14	1	300	232426.94	108.59
3	10	10	400	1996769.64	77.9
4	10	10	400	2620456.7	97.79
5	10	10	400	2867432.28	75.43
6	3	3	120	517876.02	37.34
7	3	3	120	801013.44	50.73
8	2	1	60	80234.5	20.5
9	3	2	60	504084.15	43.93

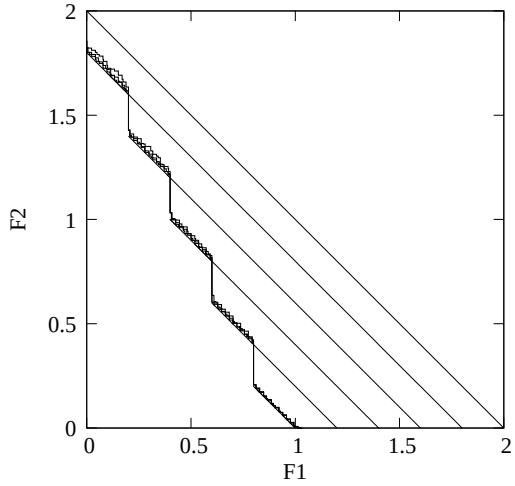


Figure 25: Attainment surfaces (100%, 50% and 0%) for problem 7 from 21 runs. The hypervolumes for the obtained attainment surfaces are 0.5216, 0.5281 and 0.5308

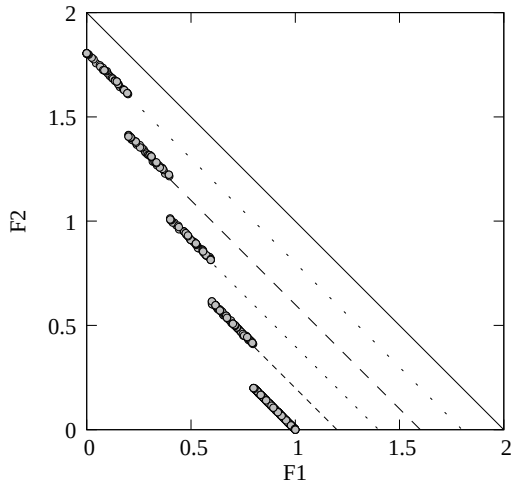


Figure 26: Best run for problem 7 from 21 runs. The hypervolume for the obtained non-dominated front is 0.5273

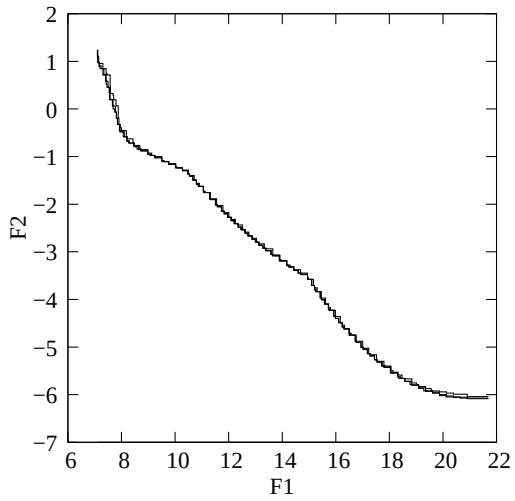


Figure 27: Attainment surfaces (100%, 50% and 0%) for problem 8 from 21 runs. The hypervolumes for the obtained attainment surfaces are 0.6342, 0.6362 and 0.6363

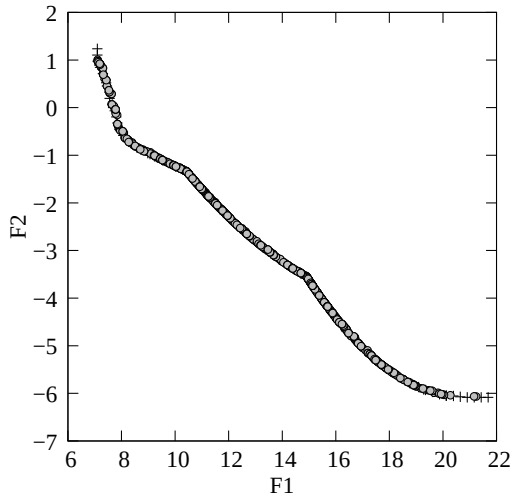


Figure 28: Best run for problem 8 from 21 runs. The hypervolume for the obtained non-dominated front is 0.6362

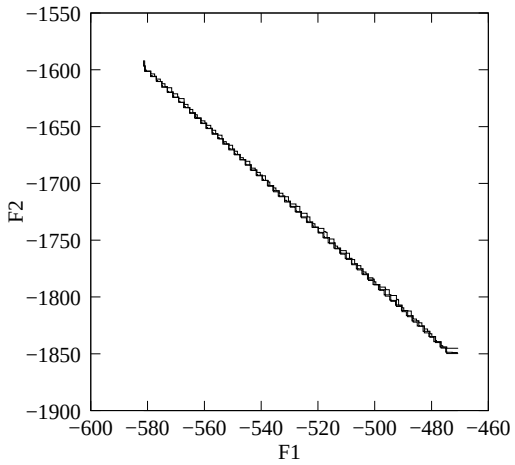


Figure 29: Attainment surfaces (100%, 50% and 0%) for problem 9 from 21 runs. The hypervolumes for the obtained attainment surfaces are 0.5239, 0.5255 and 0.5260

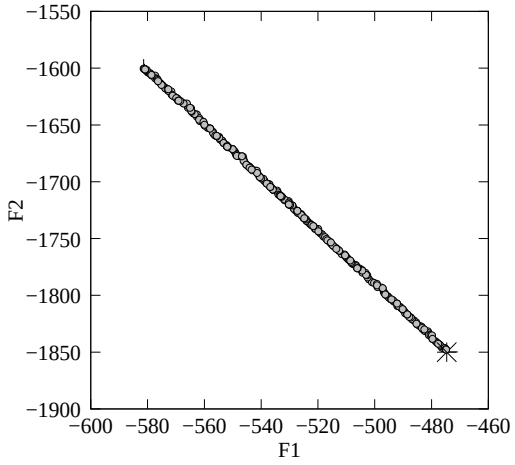


Figure 30: Best run for problem 9 from 21 runs. The hypervolume for the obtained non-dominated front is 0.5253

Table 2: Total function evaluations for the upper and lower level (21 runs). The lower level function evaluations include the evaluations of SQP

Pr. No.	**Best**		**Median**		**Worst**	
	Total LL Function Evaluations	Total UL Function Evaluations	Total LL Function Evaluations	Total UL Function Evaluations	Total LL Function Evaluations	Total UL Function Evaluations
1	555410	12438	629590	14163	700634	16146
2	264675	17262	319499	18733	355904	21429
3	2819770	87582	3423544	91852	3829812	107659
4	4484580	105439	4695352	116605	5467633	138107
5	3970411	112560	4725596	118848	5265074	125438
6	813702	32047	869459	34419	916982	38555
7	905912	33534	996633	34281	1136518	40475
8	210794	10513	261030	11367	304654	11634
9	1624658	37218	1927022	39960	2036208	41570

Table 3: Comparison of function evaluations for $\tau = -1$ and $\tau = +1$ cases.

Pr. No.	**Best**		**Median**		**Worst**	
	Total LL Function Evaluations	Total UL Function Evaluations	Total LL Function Evaluations	Total UL Function Evaluations	Total LL Function Evaluations	Total UL Function Evaluations
3($\tau = +1$)	2819770	87582	3423544	91852	3829812	107659
3($\tau = -1$)	3139381	92624	3597090	98934	4087557	113430
4($\tau = +1$)	4484580	105439	4695352	116605	5467633	138107
4($\tau = -1$)	4796131	112563	4958593	122413	5731016	144428

10 Results on Single-Objective Bilevel Problems

10.1 Problem 1

10.2 Problem 2

10.3 Problem 3

11 Parametric Study

12 Discussions

13 Conclusions

14 Acknowledgements

Authors wish to thank Academy of Finland and Foundation of Helsinki School of Economics for their support of this study.

References

Calamai, P. H. and Vicente, L. N. (1994). Generating quadratic bilevel programming test problems. *ACM Trans. Math. Software*, 20(1):103–119.

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

Deb, K. (2001). *Multi-objective optimization using evolutionary algorithms*. Chichester, UK: Wiley.

Deb, K. and Agrawal, R. B. (1995). Simulated binary crossover for continuous search space. *Complex Systems*, 9(2):115–148.

Deb, K., Agrawal, S., Pratap, A., and Meyarivan, T. (2002). A fast and elitist multi-objective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197.

Deb, K., Thiele, L., Laumanns, M., and Zitzler, E. (2005). Scalable test problems for evolutionary multi-objective optimization. In Abraham, A., Jain, L., and Goldberg, R., editors, *Evolutionary Multiobjective Optimization*, pages 105–145. London: Springer-Verlag.

Dempe, S., Dutta, J., and Lohse, S. (2006). Optimality conditions for bilevel programming problems. *Optimization*, 55(5âĂŞ6):505âĂŞ–524.

Eichfelder, G. (2007). Soving nonlinear multiobjective bilevel optimization problems with coupled upper level constraints. Technical Report Preprint No. 320, Preprint-Series of the Institute of Applied Mathematics, Univ. Erlangen-NÃijrnberg, Germany.

Fliege, J. and Vicente, L. N. (2006). Multicriteria approach to bilevel optimization. *Journal of Optimization Theory and Applications*, 131(2):209–225.

Koh, A. (2007). Solving transportation bi-level programs with differential evolution. In *2007 IEEE Congress on Evolutionary Computation (CEC-2007)*, pages 2243–2250. IEEE Press.

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

Oduguwa, V. and Roy, R. (2002). Bi-level optimisation using genetic algorithm. In *Proceedings of the 2002 IEEE International Conference on Artificial Intelligence Systems (ICAISâĂŹ02)*, pages 322–327.

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

Wang, Y., Jiao, Y.-C., and Li, H. (2005). An evolutionary algorithm for solving nonlinear bilevel programming based on a new constraint-handling scheme. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 35(2):221–232.

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

Zitzler, E., Deb, K., and Thiele, L. (2000). Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation Journal*, 8(2):125–148.