An Efficient and Accurate Solution Methodology for Bilevel Multi-Objective Programming Problems Using a Hybrid Evolutionary-Local-Search Algorithm

Kalyanmoy Deb deb@iitk.ac.in
Finland Distinguished Professor (FiDiPro), Department of Mechanical Engineering, Indian Institute of Technology Kanpur, PIN 208016, India, and Department of Business Technology, Aalto University School of Economics, PO Box 21220, 00076, Helsinki, Finland

Ankur Sinha ankur.sinha@hse.fi
Department of Business Technology, Aalto University School of Economics, PO Box 21220, 00076, Helsinki, Finland

Abstract

Bilevel optimization problems involve two optimization tasks (upper and lower level), in which every feasible upper level solution must correspond to an optimal solution to a lower level optimization problem. These problems commonly appear in many practical problem solving tasks including optimal control, process optimization, game-playing strategy developments, transportation problems, and others. However, they are commonly converted into a single level optimization problem by using an approximate solution procedure to replace the lower level optimization task. Although there exist a number of theoretical, numerical, and evolutionary optimization studies involving single-objective bilevel programming problems, not many studies look at the context of multiple conflicting objectives in each level of a bilevel programming problem. In this paper, we address certain intricate issues related to solving multi-objective bilevel programming problems, present challenging test problems, and propose a viable and hybrid evolutionary-cum-local-search based algorithm as a solution methodology. The hybrid approach performs better than a number of existing methodologies and scales well up to 40-variable difficult test problems used in this study. The population sizing and termination criteria are made self-adaptive, so that no additional parameters need to be supplied by the user. The study indicates a clear niche of evolutionary algorithms in solving such difficult problems of practical importance compared to their usual solution by a computationally expensive nested procedure. The study opens up many issues related to multi-objective bilevel programming and hopefully this study will motivate EMO and other researchers to pay more attention to this important and difficult problem solving activity.

Keywords

Bilevel optimization, evolutionary multi-objective optimization, NSGA-II, test problem development, problem difficulties, hybrid evolutionary algorithms, self-adaptive algorithm, sequential quadratic programming.

1 Introduction

Optimization problems are usually considered to have a single level consisting of one or more objective functions to be optimized, and constraints, if present, must be satisfied (Reklaitis et al., 1983; Rao, 1984). Optimization problems come in many different forms and complexities involving the type and size of the variable vector, objective and constraint functions, nature of problem parameters, modalities of objective functions, interactions among objectives, extent of feasible search region, computational burden, and inherent noise in evaluating solutions, and so on (Deb, 2001). While these factors are keeping optimization researchers and practitioners busy in devising efficient solution procedures, the practice always seems to have more to offer than what the researchers have been able to comprehend and implement in the realm of optimization studies.

Bilevel programming problems have two levels of optimization problems—upper and lower levels (Colson et al., 2007; Vicente and Calamai, 2004). In the upper level optimization task, a solution, in addition to satisfying its own constraints, must also be an optimal solution to another optimization problem, called the lower level optimization problem. Although the concept is intriguing, bilevel programming problems commonly appear in many practical optimization problems (Bard, 1998). Thinking simply, the bilevel scenario occurs when a solution in an (upper level) optimization task must be a physically or a functionally acceptable solution, such as being a stable solution or being a solution in equilibrium or being a solution that must satisfy certain conservation principles, and so on. Satisfaction of one or more of these conditions can then be posed as another (lower level) optimization task. However, often in practice (Bianco et al., 2009; Dempe, 2002; Pakala, 1993), such problems are not usually treated as bilevel programming problems. Instead some approximate methodologies are used to replace the lower level problem. In many scenarios it is observed that approximate solution methodologies are not available or they are practically and functionally unacceptable. Ideally such problems involving an assurance of a physically or functionally viable solution can be posed as bilevel programming problems and solved.

Bilevel programming problems involving a single objective function in upper and lower levels have received some attention from theory (Dempe et al., 2006), algorithm development and application (Alexandrov and Dennis, 1994; Vicente and Calamai, 2004), and even using evolutionary algorithms (Yin, 2000; G. Wang et al., 2008). However, apart from a few recent studies (Eichfelder, 2007, 2008; Halter and Mostaghim, 2006; Shi and Xia, 2001) and our recent evolutionary multi-objective optimization (EMO) studies (Deb and Sinha, 2009a,b; Sinha and Deb, 2009), multi-objective bilevel programming studies are scarce in both classical and evolutionary optimization fields. The lack of interest for handling multiple conflicting objectives in a bilevel programming context is not due to lack of practical problems, but more due to the need for searching and storing multiple trade-off lower level solutions for a single upper level solution and due to the complex interactions which upper and lower level optimization tasks can provide. In this paper, we take a closer look at the intricacies of multi-objective bilevel programming problems, present a set of difficult test problems by using an extended version of our earlier proposed test problem construction procedure, and propose and evaluate a hybrid EMO-cum-local-search bilevel programming algorithm (H-BLEMO).

In the remainder of this paper, we briefly outline a generic multi-objective bilevel optimization problem and then provide an overview of existing studies both on single

and multi-objective bilevel programming. Past evolutionary methods are particularly highlighted. Thereafter, we list a number of existing multi-objective bilevel test problems and then discuss an extension of our recent suggestion. The proposed hybrid and self-adaptive bilevel evolutionary multi-objective optimization algorithm (H-BLEMO) is then described in detail by providing a step-by-step procedure. Simulation results on eight different problems are shown. A comparison of the performance of the proposed algorithm with our previously-proposed methodology and with a nested optimization strategy is made. The test problem construction procedure allowed us to create problems that exhibit conflicting goals between lower and upper level optimization tasks. The use of local search and self-adaptive methodologies enabled us to solve such difficult problems using H-BLEMO, whereas these same problems are found to be unsolvable by our earlier methods. Further, a scalability study of the proposed algorithm is made by considering different problem sizes ranging from 10 to 40 decision variables. Finally, conclusions of the study are presented.

2 Multi-Objective Bilevel Optimization Problems

A multi-objective bilevel optimization problem has two levels of multi-objective optimization problems such that a feasible solution of the upper level problem must be a member of the Pareto-optimal set of the lower level optimization problem. A general multi-objective bilevel optimization problem can be described as follows:

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

In the above formulation, $F_1(\mathbf{x}), \ldots, F_M(\mathbf{x})$ are upper level objective functions and $f_1(\mathbf{x}), \ldots, f_m(\mathbf{x})$ are lower level objective functions. The functions $\mathbf{g}(\mathbf{x})$ and $\mathbf{h}(\mathbf{x})$ determine the feasible space for the lower level problem. The decision vector is $\mathbf{x}$ which is composed of two smaller vectors $\mathbf{x}_u$ and $\mathbf{x}_l$, such that $\mathbf{x} = (\mathbf{x}_u, \mathbf{x}_l)$. It should be noted that the lower level optimization problem is optimized only with respect to the variables $\mathbf{x}_l$ and the variables $\mathbf{x}_u$ act as fixed parameters for the problem. Therefore, the solution set of the lower level problem can be represented as a function of $\mathbf{x}_u$, or as $\mathbf{x}_l^*(\mathbf{x}_u)$. This means that the upper level variables ($\mathbf{x}_u$), act as a parameter to the lower level problem and hence the lower level optimal solutions ($\mathbf{x}_l^*$) are a function of the upper level vector $\mathbf{x}_u$. The functions $\mathbf{G}(\mathbf{x})$ and $\mathbf{H}(\mathbf{x})$ along with the Pareto-optimality to the lower level problem determine the feasible space for the upper level optimization problem. Both sets $\mathbf{x}_l$ and $\mathbf{x}_u$ are decision variables for the upper level problem.

2.1 Practical Importance

Bilevel multi-objective optimization problems arise from hierarchical problems in practice in which a strategy for solving the overall system depends on optimal strategies of solving a number of subsystems. Let us consider two different examples to illustrate these problems.

Many engineering design problems in practice involve an upper level optimization problem requiring that a feasible solution to the problem must satisfy certain physical conditions, such as satisfying a network flow balance or satisfying stability

conditions or satisfying some equilibrium conditions. If simplified mathematical equations for such conditions are easily available, often they are directly used as constraints and the lower level optimization task is avoided. But in many problems establishing whether a solution is stable or in equilibrium can be done by ensuring that the solution is an optimal solution to a derived optimization problem. Such a derived optimization problem can then be formulated as a lower level problem in a bilevel optimization problem. A common source of bilevel problems is in chemical process optimization problems, in which the upper level problem optimizes the overall cost and quality of product, whereas the lower level optimization problem optimizes error measures indicating how closely the process adheres to different theoretical process conditions, such as mass balance equations, cracking, or distillation principles (Dempe, 2002).

The bilevel problems are also similar in principle to the Stackelberg games (Fudenberg and Tirole, 1993; J. F. Wang and Periaux, 2001) in which a leader makes the first move and a follower then maximizes its move considering the leader's move. The leader has an advantage in that it can control the game by making its move in a way so as to maximize its own gain knowing that the follower will always maximize its own gain. For example (Zhang et al., 2007), a company CEO (leader) may be interested in maximizing net profits and quality of products, whereas heads of branches (followers) may maximize their own net profit and worker satisfaction. The CEO knows that for each person's strategy, the heads of branches will optimize their own objectives. The CEO must then adjust his or her own decision variables so that the CEO's own objectives are maximized. Stackelberg's game model and its solutions are used in many different problem domains, including engineering design (Pakala, 1993), security applications (Paruchuri et al., 2008), and others.

3 Existing Classical and Evolutionary Methodologies

The importance of solving bilevel optimization problems, particularly problems having a single objective in each level, has been recognized amply in the optimization literature. The research has been focused in both theoretical and algorithmic aspects. However, there has been lukewarm interest in handling bilevel problems having multiple conflicting objectives in any or both levels. Here we provide a brief description of the main research outcomes so far in both single and multi-objective bilevel optimization areas.

3.1 Theoretical Developments

Several studies exist in determining the optimality conditions for an upper level solution. The difficulty arises due to the existence of another optimization problem as a hard constraint to the upper level problem. Usually the Karush-Kuhn-Tucker (KKT) conditions of the lower level optimization problems are first written and used as constraints in formulating the KKT conditions of the upper level problem, involving second derivatives of the lower level objectives and constraints as the necessary conditions of the upper level problem. However, 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 (2006) suggested a mapping concept in which a bilevel single-objective optimization problem (one objective each in upper and lower level problems)

can be converted to an equivalent four-objective optimization problem with a special cone dominance concept. Although the idea may apparently be extended for bilevel multi-objective optimization problems, no such suggestion with an exact mathematical formulation is made yet. Moreover, derivatives of original objectives are involved in the problem formulation, thereby making the approach limited to only differentiable problems.

3.2 Algorithmic Developments

One simple algorithm for solving bilevel optimization problems using a point-by-point approach would be to directly treat the lower level problem as a hard constraint. Every solution $[\mathbf{x} = (\mathbf{x}_u, \mathbf{x}_l)]$ must be sent to the lower level problem as an initial point and an optimization algorithm can then be employed to find the optimal solution $\mathbf{x}_l^*$ of the lower level optimization problem. Then, the original solution $\mathbf{x}$ of the upper level problem must be repaired as $(\mathbf{x}_u, \mathbf{x}_l^*)$. The employment of a lower level optimizer within the upper level optimizer for every upper level solution makes the overall search a *nested* optimization procedure, which may be computationally an expensive task. Moreover, if this idea is to be extended for multiple conflicting objectives in the lower level, for every upper level solution, multiple Pareto-optimal solutions for the lower level problem need to be found and stored by a suitable multi-objective optimizer.

Another idea (Herskovits et al., 2000; Bianco et al., 2009) of handling the lower level optimization problem having differentiable objectives and constraints is to include the explicit KKT conditions of the lower level optimization problem directly as constraints to the upper level problem. This will then involve Lagrange multipliers of the lower level optimization problem as additional variables to the upper level problem. As KKT points need not always be optimum points, further conditions will have to be included to ensure the optimality of lower level problem. For multi-objective bilevel problems, corresponding multi-objective KKT formulations need to be used, thereby involving further Lagrange multipliers and optimality conditions as constraints to the upper level problem.

Despite these apparent difficulties, there exist some useful studies, including reviews on bilevel programming (Colson et al., 2007; Vicente and Calamai, 2004), test problem generators (Calamai and Vicente, 1994), nested bilevel linear programming (Gaur and Arora, 2008), and applications (Fampa et al., 2008; Abass, 2005; Koh, 2007), mostly in the realm of single-objective bilevel optimization.

Recent studies by Eichfelder (2007, 2008) concentrated on handling multi-objective bilevel problems using classical methods. While the lower level problem uses a numerical optimization technique, the upper level problem is handled using an adaptive exhaustive search method, thereby making the overall procedure computationally expensive for large-scale problems. This method uses the nested optimization strategy to find and store multiple Pareto-optimal solutions for each of finitely-many upper level variable vectors.

Another study by Shi and Xia (2001) transformed a multi-objective bilevel programming problem into a bilevel ϵ-constraint approach in both levels by keeping one of the objective functions and converting remaining objectives to constraints. The ϵ values for constraints were supplied by the decision-maker as different levels of "satisfactoriness." Further, the lower-level single-objective constrained optimization problem was replaced by equivalent KKT conditions and a variable metric optimization method was used to solve the resulting problem.

Certainly, more efforts are needed to devise effective classical methods for multi-objective bilevel optimization, particularly to handle the upper level optimization task in a more coordinated way with the lower level optimization task.

3.3 Evolutionary Methods

Several researchers have proposed evolutionary algorithm based approaches in solving single-objective bilevel optimization problems. As early as in 1994, Mathieu et al. (1994) proposed a GA-based approach for solving bilevel linear programming problems having a single objective in each level. The lower level problem was solved using a standard linear programming method, whereas the upper level was solved using a GA. Thus, this early GA study used a nested optimization strategy, which may be computationally too expensive to extend to nonlinear and large-scale problems. Yin (2000) proposed another GA based nested approach in which the lower level problem was solved using the Frank-Wolfe gradient based linearized optimization method and claimed to solve nonconvex bilevel optimization problems better than an existing classical method. Oduguwa and Roy (2002) suggested a coevolutionary GA approach in which two different populations are used to handle variable vectors $\mathbf{x}_u$ and $\mathbf{x}_l$ independently. Thereafter, a linking procedure is used to set up cross-talk between the populations. For single-objective bilevel optimization problems, the final outcome is usually a single optimal solution in each level. The proposed coevolutionary approach is viable for finding corresponding single solution in $\mathbf{x}_u$ and $\mathbf{x}_l$ spaces. But in handling multi-objective bilevel programming problems, multiple solutions corresponding to each upper level solution must be found and maintained during the coevolutionary process. It is not clear how such a coevolutionary algorithm can be designed effectively for handling multi-objective bilevel optimization problems. We do not address this issue in this paper, but recognize that Oduguwa and Roy's study (2002) was the first to suggest a coevolutionary procedure for single-objective bilevel optimization problems. Since 2005, a surge in research in this area can be found in algorithm development mostly using the nested approach and the explicit KKT conditions of the lower level problem, and in various application areas (Hecheng and Wang, 2007; Li and Wang, 2007; Dimitriou et al., 2008; Yin, 2000; Mathieu et al., 1994; Sun et al., 2006; Wang et al., 2007; Koh, 2007; Wang et al., 2005; G. Wang et al., 2008).

Li et al. (2006) proposed particle swarm optimization (PSO) based procedures for both lower and upper levels, but instead of using a nested approach, they proposed a serial application of upper and lower levels iteratively. This idea is applicable in solving single-objective problems in each level due to the sole target of finding a single optimal solution. As discussed above, in the presence of multiple conflicting objectives in each level, multiple solutions need to be found and preserved for each upper level solution and then a serial application of upper and lower level optimization does not make sense for multi-objective bilevel optimization. Halter and Mostaghim (2006) also used PSO on both levels, but since the lower level problem in their application problem was linear, they used a specialized linear multi-objective PSO algorithm and used an overall nested optimization strategy at the upper level.

Recently, we have proposed a number of EMO algorithms through conference publications (Deb and Sinha, 2009a,b; Sinha and Deb, 2009) using NSGA-II to solve both level problems in a synchronous manner. First, our methodologies were generic so that they can be used to linear/nonlinear, convex/nonconvex, differentiable/nondifferentiable and single-objective/multi-objective problems at both levels. Second, our

methodologies did not use the nested approach, nor did they use a serial approach, but employed a structured intertwined evolution of upper and lower level populations. But they were computationally demanding. However, these initial studies made us understand the complex intricacies by which both level problems can influence each other. Based on this experience, in this paper, we suggest a less-structural, self-adaptive, computationally fast, hybrid evolutionary algorithm coupled with a local search procedure for handling multi-objective bilevel programming problems.

Bilevel programming problems, particularly with multiple conflicting objectives, should have been paid more attention than what has been made so far. As more and more studies are performed, the algorithms must be tested and compared against each other. This process needs an adequate number of test problems with tunable difficulties. In the next section, we suggest a generic procedure for developing test problems and suggest a test suite.

4 Test Problems for Multi-Objective Bilevel Programming

To date there has not been any systematic study in developing test problems for multi-objective bilevel programming. However, Eichfelder (2007) used a number of test problems in her study, in which some problems were not analyzed for their exact Pareto-optimal fronts. Here, we first describe some of these existing test problems (we refer to these existing test problems as TP here) and then discuss an extension of our recently proposed test problem development procedure (Deb and Sinha, 2009a) for any number of objectives and scenarios. The principle is used to generate five test problems (we refer to our new test problems as DS here).

4.1 Problem TP1

Problem TP1 has a total of three variables with $\mathbf{x}_l = (x_1, x_2)^T$ and $\mathbf{x}_u = (y)$ (Eichfelder, 2007):

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

The Pareto-optimal set for the lower-level optimization task for a fixed y is the bottom-left quarter of the circle: $\{(x_1, x_2) \in \mathbb{R}^2 \mid x_1^2 + x_2^2 = y^2, x_1 \leq 0, x_2 \leq 0\}$. 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 (2007) reported the following Pareto-optimal solutions for this problem:

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

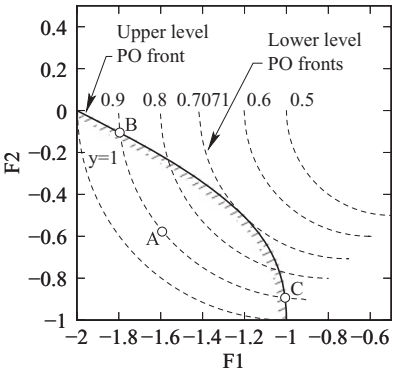

Figure 1: Pareto-optimal fronts of upper level (complete problem) and some representative lower level optimization tasks are shown for problem TP1.

Figure 1 shows the Pareto-optimal front for the problem TP1. Lower level Pareto-optimal fronts of some representative y values are also shown in the figure using dashed lines, 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 become the candidate Pareto-optimal solutions of the upper level problem.

4.2 Problem TP2

We created a simplistic bilevel two-objective optimization problem elsewhere (Deb and Sinha, 2009b), having $\mathbf{x}_l = (x_1, \ldots, x_K)$ and $\mathbf{x}_u = (y)$:

$$\begin{aligned}
&\text{Minimize} \quad \mathbf{F}(\mathbf{x}) = \begin{pmatrix} (x_1-1)^2 + \sum_{i=2}^{K} x_i^2 + y^2 \\ (x_1-1)^2 + \sum_{i=2}^{K} x_i^2 + (y-1)^2 \end{pmatrix}, \\
&\text{subject to} \\
&(x_1, x_2, \ldots, x_K) \in \text{argmin}_{(x_1,x_2,\ldots,x_K)} \left\{ \mathbf{f}(\mathbf{x}) = \begin{pmatrix} x_1^2 + \sum_{i=2}^{K} x_i^2 \\ (x_1-y)^2 + \sum_{i=2}^{K} x_i^2 \end{pmatrix} \right\}, \\
&-1 \le (x_1, x_2, \ldots, x_K, y) \le 2.
\end{aligned} \tag{4}$$

For a fixed value of y, the Pareto-optimal solutions of the lower level optimization problem are given as follows: $\{\mathbf{x}_l \in \mathbb{R}^K \mid x_1 \in [0, y], x_i = 0, \text{ for } i = 2, \ldots, K\}$. However, for the upper level problem, the Pareto-optimal solutions correspond to the following conditions: $\{\mathbf{x} \in \mathbb{R}^{K+1} \mid x_1 = y, x_i = 0, \text{ for } i = 2, \ldots, K, y \in [0.5, 1.0]\}$. If an algorithm fails to find the true Pareto-optimal solutions of the lower level problem and ends up finding a solution below the $x_1 = y$ curve in Figure 2 (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 a difficult task. We use $K = 14$ here, so that the total number of variables is 15 in this problem.

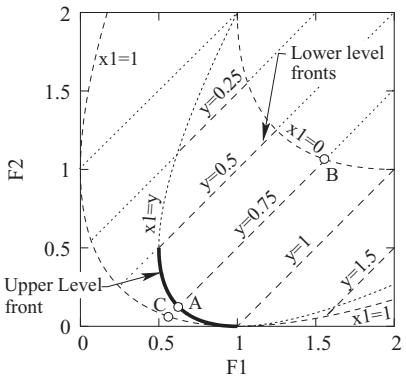

Figure 2: Pareto-optimal front of the upper level problem for problem TP2 with $x_i = 0$ for $i = 2, \ldots, K$.

4.3 Test Problem TP3

This problem is taken from Eichfelder (2007):

$$
\begin{aligned}
&\text{Minimize} \quad \mathbf{F(x)} = \begin{pmatrix} x_1 + x_2^2 + y + \sin^2(x_1 + y) \\ \cos(x_2)(0.1 + y)\left(\exp\left(- \dfrac{x_1}{0.1 + x_2} \right) \right) \end{pmatrix}, \\
&\text{subject to} \\
&(x_1, x_2) \in \left\{ \begin{array}{l} \text{argmin}_{(x_1, x_2)} \mathbf{f(x)} = \left(\begin{array}{c} \dfrac{(x_1 - 2)^2 + (x_2 - 1)^2}{4} + \dfrac{x_2 y + (5 - y_1)^2}{16} + \sin\left(\dfrac{x_2}{10} \right) \\ \dfrac{x_1^2 + (x_2 - 6)^4 - 2x_1 y_1 - (5 - y_1)^2}{80} \end{array} \right) \Bigg| \\ g_1(\mathbf{x}) = x_2 - x_1^2 \geq 0 \\ g_2(\mathbf{x}) = 10 - 5x_1^2 - x_2 \geq 0 \\ g_3(\mathbf{x}) = 5 - \dfrac{y}{6} - x_2 \geq 0 \\ g_4(\mathbf{x}) = x_1 \geq 0 \end{array} \right\}, \\
&G_1(\mathbf{x}) \equiv 16 - (x_1 - 0.5)^2 - (x_2 - 5)^2 - (y - 5)^2 \geq 0, \\
&0 \leq x_1, x_2, y \leq 10.
\end{aligned}
\tag{5}
$$

For this problem, the exact Pareto-optimal front of the lower or the upper level optimization problem are not derived mathematically. For this reason, we do not consider this problem any further here. Some results using our earlier BLEMO procedure can be found elsewhere (Deb and Sinha, 2009b).

4.4 Test Problem TP4

The next problem comes from a company scenario dealing with the management decisions between a CEO (leader) and heads of branches (follower; Zhang et al., 2007). Although fuzziness in the coefficients was used in the original study, here we present

the deterministic version of the problem:

$$\begin{aligned}
&\text{Maximize} \quad \mathbf{F}(\mathbf{x},\mathbf{y}) = \qquad \text{subject to } (\mathbf{x}) \in \operatorname{argmin}_{(\mathbf{x})} \\
&\begin{pmatrix} (1,9)(y_1,y_2)^T \\ \quad +(10,1,3)(x_1,x_2,x_3)^T \\ (9,2)(y_1,y_2)^T \\ \quad +(2,7,4)(x_1,x_2,x_3)^T \end{pmatrix}, \left\{ \begin{array}{l} \mathbf{f(x)} = \begin{pmatrix} (4,6)(y_1,y_2)^T + (7,4,8)(x_1,x_2,x_3)^T \\ (6,4)(y_1,y_2)^T + (8,7,4)(x_1,x_2,x_3)^T \end{pmatrix} \Big| \\ g1 = (3,-9)(y_1,y_2)^T + (-9,-4,0)(x_1,x_2,x_3)^T \le 61 \\ g2 = (5,9)(y_1,y_2)^T + (10,-1,-2)(x_1,x_2,x_3)^T \le 924 \\ g3 = (3,-3)(y_1,y_2)^T + (0,1,5)(x_1,x_2,x_3)^T \le 420 \end{array} \right\}, \\
&G1 = (3,9)(y_1,y_2)^T + (9,5,3)(x_1,x_2,x_3)^T \le 1039, \\
&G2 = (-4,-1)(y_1,y_2)^T + (3,-3,2)(x_1,x_2,x_3)^T \le 94, \\
&x_1,x_2,y_1,y_2,y_3 \ge 0.
\end{aligned} \tag{6}$$

4.5 Development of Tunable Test Problems

Bilevel multi-objective optimization problems are different from single-level multi-objective optimization problems in that the Pareto-optimality of a lower level multi-objective optimization problem is a requirement 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 in addition to their interactions. Further, the test problems should be such that we would have a precise knowledge about the exact location (and relationships) of Pareto-optimal points. Thinking along these lines, we outline a number of desired properties in a bilevel multi-objective test problem:

1. *Exact location of Pareto-optimal solutions in both lower and upper level problems are possible to be established.* This will enable a user to evaluate the performance of an algorithm easily by comparing the obtained solutions with the exact Pareto-optimal solutions.

2. *Problems are scalable with respect to number of variables.* This will allow a user to investigate whether the proposed algorithm 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 levels.* This will enable a user to evaluate whether the proposed algorithm scales well with the number of objectives in each level.

4. *Lower level problems are difficult to solve to Pareto-optimality.* If the lower level Pareto-optimal front is not found exactly, the corresponding upper level solution is not feasible. Therefore, these problems will test an algorithm's ability to converge to the correct Pareto-optimal front. Here, ideas can be borrowed from single-level EMO test problems (Deb et al., 2005) to construct difficult lower level optimization problems. The shape (convex, nonconvex, disjointedness, and multi-modality) of the Pareto-optimal front will also play an important role in this respect.

5. *There is a conflict between lower and upper level problem solving tasks.* For two solutions $\mathbf{x}$ and $\mathbf{y}$ of which $\mathbf{x}$ is Pareto-optimal and $\mathbf{y}$ is a dominated solution in the lower level, solution $\mathbf{y}$ can be better than solution $\mathbf{x}$ in the upper level. Due to these discrepancies, these problems will cause a conflict in converging to the appropriate Pareto-optimal front in both lower and upper level optimization tasks.

6. *Extension to higher level optimization problems is possible.* Although our focus here is for bilevel problems only, test problems scalable to three or more levels would be interesting, as there may exist some practical problems formulated in three or more levels. On the other hand, it will also be ideal to have bilevel test problems that will degenerate to challenging single level test problems, if a single objective function is chosen for each level.

7. *Different lower level problems may contribute differently to the upper level front in terms of their extent of representative solutions on the upper level Pareto-optimal front.* These test problems will test an algorithm's ability to emphasize different lower level problems differently in order to find a well-distributed set of Pareto-optimal solutions at the upper level.

8. *Test problems must include constraints at both levels.* This will allow algorithms to be tested for their ability to handle constraints in both lower and upper level optimization problems.

Different principles are possible to construct test problems following the above guidelines. Here, we present a generalized version of a recently proposed procedure (Deb and Sinha, 2009a).

4.5.1 A Multi-Objective Bilevel Test Problem Construction Procedure

We suggest a test problem construction procedure for a bilevel problem having M and m objectives in the upper and lower level, respectively. The procedure needs at most three functional forms and is described below:

Step 1: First, a parametric trade-off function $\Phi_U : \mathbb{R}^{M-1} \rightarrow \mathbb{R}^M$ which determines a trade-off frontier $(v_1(\mathbf{u}), \ldots, v_M(\mathbf{u}))$ on the **F**-space as a function of $(M-1)$ parameters $\mathbf{u}$ (can be considered as a subset of $\mathbf{x}_u$) is chosen. Figure 3 shows such a v_1-v_2 relationship on a two-objective bilevel problem.

Step 2: Next, for every point $\mathbf{v}$ on the Φ_U-frontier, an $(M-1)$-dimensional envelope $(U_1(\mathbf{t}), \ldots, U_M(\mathbf{t}))^{\mathbf{V}})$ on the **F**-space as a function of $\mathbf{t}$ (having $(M-1)$ parameters) is chosen. The nondominated part of the agglomerate envelope $\cup_{\mathbf{V}} \cup_{\mathbf{t}}$ $[(v_1(\mathbf{u}) + U_1(\mathbf{t})^{\mathbf{V}}), \ldots, (v_M(\mathbf{u}) + U_M(\mathbf{t})^{\mathbf{V}})]$ constitutes the overall upper level Pareto-optimal front. Figure 3 indicates this upper level Pareto-optimal front and some specific Pareto-optimal points (marked with bigger circles) derived from specific $\mathbf{v}$-vectors.

Step 3: Next, for every point $\mathbf{v}$ on the Φ_U-frontier, a mapping function $\Phi_L : \mathbb{R}^{M-1} \rightarrow \mathbb{R}^{m-1}$ which maps every $\mathbf{v}$-point from the **U**-frontier to the lower level Pareto-optimal front $(f_1^*(\mathbf{s}), \ldots, f_m^*(\mathbf{s}))^{\mathbf{V}}$ is chosen. Here, $\mathbf{s}$ is an $(m-1)$-dimensional vector and can be considered as a subset of $\mathbf{x}_l$. Figure 3 shows this mapping. The envelope A′C′B′ (a circle in the figure) is mapped to the lower level Pareto-optimal frontier ACB (inlet figure on top).

Step 4: After these three functions are defined, the lower level problem can be constructed by using a bottom-up procedure adopted in Deb et al. (2005) through additional terms arising from other lower level decision variables: $f_j(\mathbf{x}_l) = f_j^*(\mathbf{s}) + e_j(\mathbf{x}_l \backslash \mathbf{s})$ with $e_j \geq 0$. The task of the lower level optimization task would

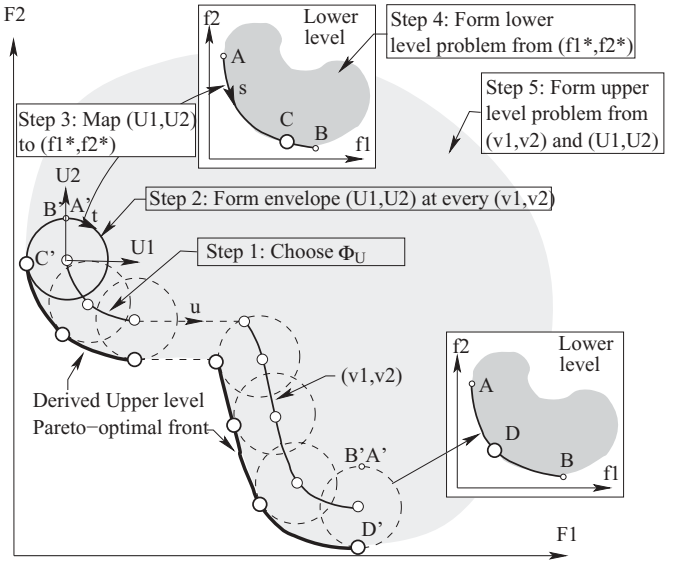

Figure 3: A multi-objective bilevel test problem construction procedure is illustrated through two objectives in both upper and lower levels.

be to make the e_j term zero for each objective. The term e_j can be made complex (multi-modal, nonlinear, or large-dimensional) to make the convergence to the lower level Pareto-optimal front difficult by an optimization algorithm.

Step 5: Finally, the upper level objectives can be formed from u_j functions by including additional terms from other upper level decision variables. An additive form is as follows: $F_j(\mathbf{x}) = u_j(\mathbf{u}) + E_j(\mathbf{x}_u \backslash \mathbf{u})$ with $E_j \geq 0$. Like the e_j term, the term E_j can also be made complex for an algorithm to properly converge to the upper level Pareto-optimal front.

Step 6: Additionally, a number of linked terms $l_j(\mathbf{x}_u \backslash \mathbf{u}, \mathbf{x}_l \backslash \mathbf{s})$ and $L_j(\mathbf{x}_u \backslash \mathbf{u}, \mathbf{x}_l \backslash \mathbf{s})$ (non-negative terms) involving remaining $\mathbf{x}_u$ (without $\mathbf{u}$) and $\mathbf{x}_l$ (without $\mathbf{s}$) variables can be added to both lower and upper level problems, respectively, to make sure a proper coordination between lower and upper level optimization tasks is needed to converge to the respective Pareto-optimal fronts.

Another interesting yet difficult scenario can be created with the linked terms. An identical link term can be added to the lower level problem, but subtracted from the upper level problem. Thus, an effort to reduce the value of the linked term will make an improvement in the lower level, whereas it will cause a deterioration in the upper level. This will create a conflict in the working of both levels of optimization. The following two-objective test problems are constructed using the above procedure.

4.5.2 Problem DS1

This problem has $2K$ variables with K real-valued variables each for lower and upper levels. Since in this study we consider bilevel optimization problems having $m = M = 2$,

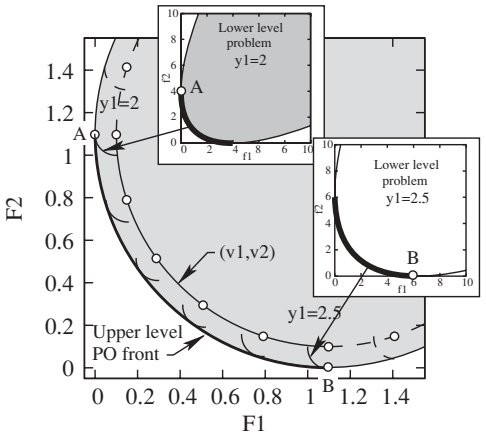

Figure 4: Pareto-optimal front for problem DS1.

the vectors **u**, **t**, and **s** all have a single element. Three functionals used for developing test problem DS1 are discussed through the following steps:

Step 1: Here, one of the upper level variables y_1 is chosen as **u**. The mapping Φ_U is chosen as follows: $v_1 = (1+r) - \cos(\pi y_1)$ and $v_2 = (1+r) - \sin(\pi y_1)$, where r is a user-supplied constant. Depending on the value of y_1, the v_1-v_2 point lies on the quarter of a circle of radius one and is centered at $(1+r, 1+r)$ in the **F**-space, as shown in Figure 4.

Step 2: For every (v_1, v_2) point on the **F**-space, the following one-dimensional (t) envelope is chosen: $v_1(y_1) = -r\cos(\gamma\frac{\pi}{2}\frac{t}{y_1})$ and $v_2(y_1) = -r\sin(\gamma\frac{\pi}{2}\frac{t}{y_1})$, where $t \in [0, y_1]$ and $\gamma(=1)$ is a constant. The envelope is a quarter of a circle of radius r and center located at (v_1, v_2) point. Although for each (v_1, v_2) point, the entire envelope (quarter circle) is a nondominated front when all (v_1, v_2) points are considered, only one point from each envelope is qualified to be on the upper level Pareto-optimal front. Thus, the Pareto-optimal front for the upper level problem lies on the quarter of a circle having radius $(1+r)$ and is centered at $(1+r, 1+r)$ on the **F**-space, as indicated in the figure.

Step 3: Each U_1-U_2 point is then mapped to a (f_1^*, f_2^*) point by the following mapping: $f_1^* = s^2$, $f_2^* = (s-y_1)^2$, where $s = t$ is assumed.

Step 4: We do not use any function l_j here.

Step 5: However, we use $E_j = (y_j - \frac{j-1}{2})^2$ for $j = 2, \ldots, K$ in the upper level problem. This will ensure that $y_j = (j-1)/2$ (for $j = 2, \ldots, K$) will correspond to the upper level Pareto-optimal front.

Step 6: Different l_j and L_j terms are used with $(K-1)$ remaining upper and lower level variables such that all lower level Pareto-optimal solutions must satisfy $x_i = y_i$ for $i = 2, \ldots, K$.

The complete DS1 problem is given below:

$$\begin{aligned}
&\text{Minimize} \quad \mathbf{F}(\mathbf{y},\mathbf{x}) = \\
&\left(\begin{array}{l}
(1+r-\cos(\alpha\pi y_1)) + \sum_{j=2}^{K}\left(y_j - \frac{j-1}{2}\right)^2 \\
\quad + \tau\sum_{i=2}^{K}(x_i-y_i)^2 - r\cos\left(\gamma\frac{\pi}{2}\frac{x_1}{y_1}\right) \\
(1+r-\sin(\alpha\pi y_1)) + \sum_{j=2}^{K}\left(y_j - \frac{j-1}{2}\right)^2 \\
\quad + \tau\sum_{i=2}^{K}(x_i-y_i)^2 - r\sin\left(\gamma\frac{\pi}{2}\frac{x_1}{y_1}\right)
\end{array}\right), \\
&\text{subject to} \quad (\mathbf{x}) \in \operatorname{argmin}_{(\mathbf{x})}\mathbf{f}(\mathbf{x}) = \\
&\left\{\begin{array}{l}
x_1^2 + \sum_{i=2}^{K}(x_i-y_i)^2 \\
\quad + \sum_{i=2}^{K} 10\left(1-\cos\left(\frac{\pi}{K}(x_i-y_i)\right)\right) \\
\sum_{i=1}^{K}(x_i-y_i)^2 \\
\quad + \sum_{i=2}^{K} 10\left|\sin\left(\frac{\pi}{K}(x_i-y_i)\right)\right|
\end{array}\right\}, \\
&-K \le x_i \le K, \quad \text{for } i = 1,\ldots,K, \\
&1 \le y_1 \le 4, \quad -K \le y_j \le K, \quad j = 2,\ldots,K.
\end{aligned} \tag{7}$$

For this test problem, we suggest $K = 10$ (overall 20 variables), $r = 0.1$, $\alpha = 1$, $\gamma = 1$, and $\tau = 1$. Since $\tau = 1$ is used, for every lower level Pareto-optimal point, $x_i = y_i$ for $i = 2,\ldots,K$ and both l_j and L_j terms are zero, thereby making an agreement for this relationship between optimal values of x_i and y_i variables in both levels.

An interesting scenario occurs when $\tau = -1$ is set. If any $\mathbf{x}$ is not Pareto-optimal to a lower level problem (meaning $x_i \neq y_i$ for $i = 2,\ldots,K$), a positive quantity is deducted from the upper level objectives from each L_j term, thereby indicating that this point will dominate some true Pareto-optimal points of the upper level problem. In fact, such points are infeasible to the upper level problem due to their nonoptimality property at the lower level problem and if allowed to exist in the upper level, they will dominate the true upper level Pareto-optimal front. Thus, this problem with $\tau = -1$ will be difficult to solve, in general, compared to problems with $\tau = 1$.

The problem DS1 is likely to provide following difficulties to a bilevel optimization algorithm:

- Lower level problem has multi-modalities, thereby making the lower level problem difficult to solve to Pareto-optimality.
- The problem is scalable to any even number of variables (by adjusting K).
- By choosing $\tau = -1$ in the linked term, a conflict in the working of lower and upper level problems can be introduced.
- By adjusting α, a small fraction of y_1 values can be made responsible for the upper level Pareto-optimal front, thereby making an algorithm difficult to locate the true Pareto-optimal points.
- By adjusting γ, only a part of the U_1-U_2 envelope (and thereby only a part of the f_1^*-f_2^* front) can be made responsible for the upper level Pareto-optimal front.

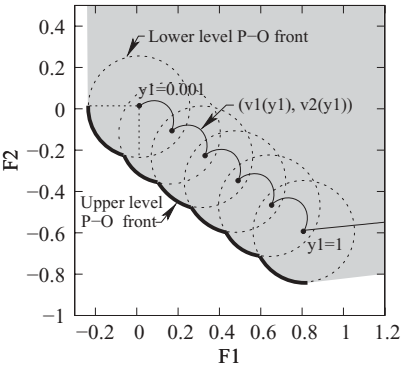

Figure 5: Pareto-optimal front for problem DS2.

4.6 Problem DS2

The next problem uses the Φ_U parametric function, which causes a few discrete values of y_1 to determine the upper level Pareto-optimal front. The Φ_U mapping function is chosen as follows and is shown in Figure 5.

$$v_1(y_1) = \begin{cases} \cos(0.2\pi)y_1 + \sin(0.2\pi)\sqrt{|0.02\sin(5\pi y_1)|}, & \text{for } 0 \le y_1 \le 1, \\ y_1 - (1 - \cos(0.2\pi)), & y_1 > 1 \end{cases}$$
$$v_2(y_1) = \begin{cases} -\sin(0.2\pi)y_1 + \cos(0.2\pi)\sqrt{|0.02\sin(5\pi y_1)|}, & \text{for } 0 \le y_1 \le 1, \\ 0.1(y_1 - 1) - \sin(0.2\pi), & \text{for } y_1 > 1. \end{cases} \tag{8}$$

The U_1-U_2 parametric function is identical to that used in DS1, but here we use $\gamma = 4$. This will cause the U_1-U_2 envelope to be a complete circle, as shown by the dashed lines in the figure. The f_1^*-f_2^* mapping is chosen identical to that in DS1. Again, the term e_j is not considered here and a multimodal E_j term is used. Different linked terms l_j and L_j compared to those used in DS1 are used here. The overall problem is given as follows:

$$\begin{aligned} &\text{Minimize} \quad \mathbf{F}(\mathbf{x}, \mathbf{y}) = \\ &\begin{pmatrix} v_1(y_1) + \sum_{j=2}^{K}\left[y_j^2 + 10\left(1 - \cos\left(\frac{\pi}{K}y_i\right)\right)\right] \\ \quad +\tau\sum_{i=2}^{K}(x_i - y_i)^2 - r\cos\left(\gamma\frac{\pi}{2}\frac{x_1}{y_1}\right) \\ v_2(y_1) + \sum_{j=2}^{K}\left[y_j^2 + 10\left(1 - \cos\left(\frac{\pi}{K}y_i\right)\right)\right] \\ \quad +\tau\sum_{i=2}^{K}(x_i - y_i)^2 - r\sin\left(\gamma\frac{\pi}{2}\frac{x_1}{y_1}\right) \end{pmatrix}, \quad \text{subject to} \quad (\mathbf{x}) \in \mathbf{f}(\mathbf{x}) = \\ &\text{argmin}_{(\mathbf{x})}\left\{\begin{pmatrix} x_1^2 + \sum_{i=2}^{K}(x_i - y_i)^2 \\ \sum_{i=1}^{K} i(x_i - y_i)^2 \end{pmatrix}\right\}, \\ &-K \le x_i \le K, \quad i = 1, \ldots, K, \\ &0.001 \le y_1 \le K, \quad -K \le y_j \le K, \ j = 2, \ldots, K, \end{aligned} \tag{9}$$

Due to the use of periodic terms in v_1 and v_2 functions, 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), despite y_1 taking any real value within $[0.001, K]$. We suggest using $r = 0.25$ here.

This problem has the following specific properties:

- The upper level problem has multi-modalities, thereby causing an algorithm difficulty in finding the upper level Pareto-optimal front.
- With $\tau = -1$, the conflict between upper and lower level problems can be introduced, as in DS1.
- The dimension of both upper and lower level problems can be increased by increasing K.
- The parameter γ can be adjusted to cause a small proportion of lower level Pareto-optimal points to be responsible for the upper level Pareto-optimal front.

4.7 Problem DS3

In this problem, the $\Phi_U 2$ and the f_1^*-f_2^*-frontiers lie on constraints and thus are not defined parametrically as in DS1 and DS2 here, but are defined directly as a function of problem variables. Since a constraint function determines the lower level Pareto-optimal front, Φ_U and Φ_L-frontiers are defined with M variables. The linked terms ($l_j = (x_i - y_i)^2$ for $i = 3, \ldots, K$) are included. We use $L_j = \tau l_j$. As before, we do not use any e_j term, but use an E_j term in the upper level problem. The variable y_1 is considered to be discrete, thereby causing only a few y_1 values to represent the upper level Pareto-optimal front. The overall problem is given below:

$$
\begin{aligned}
&\text{Minimize} \quad \mathbf{F}(\mathbf{x}, \mathbf{y}) = \\
&\begin{pmatrix}
y_1 + \sum_{j=3}^{K}(y_j - j/2)^2 + \tau \sum_{i=3}^{K}(x_i - y_i)^2 - R(y_1)\cos(4\tan^{-1}\left(\frac{y_2 - x_2}{y_1 - x_1}\right)\\
y_2 + \sum_{j=3}^{K}(y_j - j/2)^2 + \tau \sum_{i=3}^{K}(x_i - y_i)^2 - R(y_1)\sin(4\tan^{-1}\left(\frac{y_2 - x_2}{y_1 - x_1}\right)
\end{pmatrix},\\
&\text{subject to} \quad (\mathbf{x}) \in \text{argmin}_{(\mathbf{x})}\\
&\left\{ \mathbf{f}(\mathbf{x}) = \begin{pmatrix} x_1 + \sum_{i=3}^{K}(x_i - y_i)^2 \\ x_2 + \sum_{i=3}^{K}(x_i - y_i)^2 \end{pmatrix} \;\middle|\; g_1(\mathbf{x}) = (x_1 - y_1)^2 + (x_2 - y_2)^2 \le r^2 \right\},\\
&G(\mathbf{y}) = y_2 - (1 - y_1^2) \ge 0,\\
&-K \le x_i \le K, \quad \text{for } i = 1, \ldots, K, \quad 0 \le y_j \le K, \quad \text{for } j = 1, \ldots, K,\\
&\text{where } y_1 \text{ is a multiple of } 0.1.
\end{aligned}
\tag{10}
$$

Here we suggest a periodically changing radius: $R(y_1) = 0.1 + 0.15|\sin(2\pi(y_1 - 0.1)|$ and use $r = 0.2$. For the upper level Pareto-optimal points, $y_i = j/2$ for $j \le 3$. 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 lie on the third quadrant of a circle of radius r and center at (y_1, y_2) in the **F**-space. Note in Figure 6 how lower level Pareto-optimal solutions for $y_1 = 0.1$ and 0.2 (shown in dashed lines in the figure) mapped to corresponding circles in the

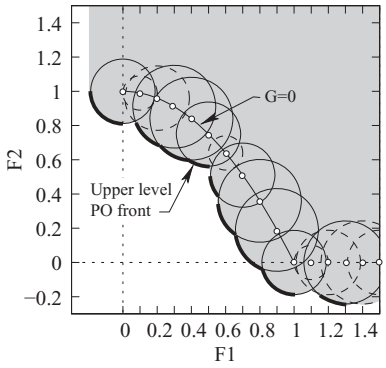

Figure 6: Pareto-optimal front for problem DS3.

upper level problem get dominated by that for $y_1 = 0$ and 0.3. The following properties are observed for this problem:

- The Pareto-optimal fronts for both the lower and upper level lie on constraint boundaries, thereby requiring good constraint handling strategies to solve both problems optimally.
- Not all lower level Pareto-optimal solutions qualify as upper level Pareto-optimal solutions.
- Every lower level front has an unequal contribution to the upper level Pareto-optimal front.
- By choosing $\tau = -1$, conflict between two levels of optimization can be introduced.

4.8 Problem DS4

In this problem, the v_1-v_2 relationship is linear ($v_1 = 2 - y_1$, $v_2 = 2(y_1 - 1)$), spanning the first quadrant of **F**-space as shown in Figure 7. The mapping U_1-U_2 is not considered here. For every (v_1, v_2) point, the following relationship is chosen for the lower level Pareto-optimal front: $f_1^* + f_2^* = y_1$. Additional terms having a minimum value of one are multiplied to form the lower and upper level search spaces. This problem has $K + L + 1$ variables, which are all real-valued:

$$\text{Minimize} \quad \mathbf{F}(\mathbf{x}, \mathbf{y}) = \begin{pmatrix} (1 - x_1)\left(1 + \sum_{j=2}^{K} x_j^2\right) y_1 \\ x_1 \left(1 + \sum_{j=2}^{K} x_j^2\right) y_1 \end{pmatrix}, \quad \text{subject to} \quad (\mathbf{x}) \in \text{argmin}_{(\mathbf{x})} \mathbf{f}(\mathbf{x}) = \left\{ \begin{pmatrix} (1 - x_1)\left(1 + \sum_{j=K+1}^{K+L} x_j^2\right) y_1 \\ x_1 \left(1 + \sum_{j=K+1}^{K+L} x_j^2\right) y_1 \end{pmatrix} \right\}, \tag{11}$$

$$G_1(\mathbf{x}) = (1 - x_1)y_1 + \tfrac{1}{2}x_1 y_1 - 1 \geq 0,$$
$$-1 \leq x_1 \leq 1, \quad 1 \leq y_1 \leq 2,$$
$$-(K + L) \leq x_i \leq (K + L), \; i = 2, \ldots, (K + L).$$

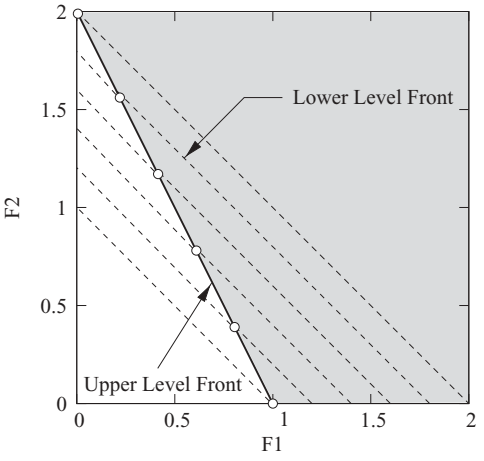

Figure 7: Pareto-optimal front for problem DS4.

The upper level Pareto-optimal front is formed with $x_i = 0$ for all $i = 2, \ldots, (K + L)$ and $x_1 = 2(1 - 1/y_1)$ and $y_1 \in [1, 2]$. This problem has the following properties:

- By increasing K and L, the problem complexity in converging to the appropriate lower and upper level fronts can be increased.
- Only one Pareto-optimal point from each participating lower level problem qualifies to be on the upper level front.

For our study here, we choose $K = 5$ and $L = 4$ (an overall 10-variable problem).

4.9 Problem DS5

This problem is similar to problem DS4 except that the upper level Pareto-optimal front is constructed from multiple points from a few lower level Pareto-optimal fronts. There are $K + L + 1$ real-valued variables in this problem as well:

$$\begin{aligned}
&\text{Minimize} \quad \mathbf{F}(\mathbf{x}, \mathbf{y}) && \text{subject to} \quad (\mathbf{x}) \in \operatorname{argmin}_{(\mathbf{x})} \mathbf{f}(\mathbf{x}) \\
&= \begin{pmatrix} (1 - x_1)\left(1 + \sum_{j=2}^{K} x_j^2\right) y_1 \\ x_1 \left(1 + \sum_{j=2}^{K} x_j^2\right) y_1 \end{pmatrix}, && = \left\{ \begin{pmatrix} (1 - x_1)\left(1 + \sum_{j=K+1}^{K+L} x_j^2\right) y_1 \\ x_1 \left(1 + \sum_{j=K+1}^{K+L} x_j^2\right) y_1 \end{pmatrix} \right\}, \\
&G_1(\mathbf{x}) = (1 - x_1)y_1 + \tfrac{1}{2}x_1 y_1 - 2 + \tfrac{1}{5}\left[5(1 - x_1)y_1 + 0.2\right] \geq 0, \\
&\quad [\cdot] \text{ denotes greatest int. function}, \\
&-1 \leq x_1 \leq 1, \quad 1 \leq y_1 \leq 2, \\
&-(K + L) \leq x_i \leq (K + L), \; i = 2, \ldots, (K + L).
\end{aligned} \tag{12}$$

For the upper level Pareto-optimal front, $x_i = 0$ for $i = 2, \ldots, (K + L)$, $x_1 \in [2(1 - 1/y_1), 2(1 - 0.9/y_1)]$, $y_1 \in \{1, 1.2, 1.4, 1.6, 1.8\}$, as shown in Figure 8. For this test problem we

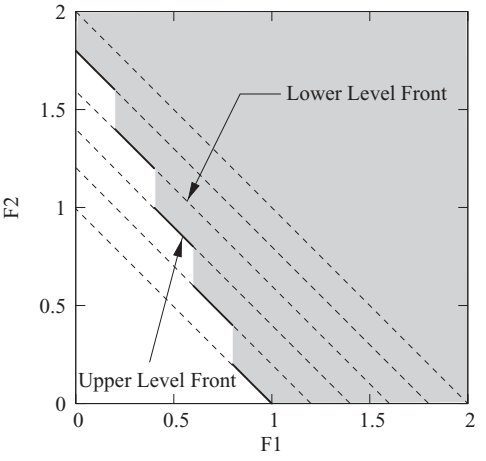

Figure 8: Pareto-optimal front for problem DS5.

have chosen $K = 5$ and $L = 4$ (an overall 10-variable problem). This problem has similar difficulties as in DS4, except that only a finite number of y_1 qualifies at the upper level Pareto-optimal front and that a consecutive set of lower level Pareto-optimal solutions now qualify to be on the upper level Pareto-optimal front.

5 Hybrid Bilevel Evolutionary Multi-Objective Optimization (H-BLEMO) Algorithm

The proposed hybrid BLEMO procedure is motivated from our previously suggested algorithms (Deb and Sinha, 2009a,b), but differs in many different fundamental ways. Before we describe the differences, we first outline the proposed hybrid procedure.

A sketch of the population structure is shown in Figure 9. The initial population (marked with upper level generation counter $T = 0$ of size N_u) has a subpopulation of lower level variable set $\mathbf{x}_l$ for each upper level variable set $\mathbf{x}_u$. Initially the subpopulation size ($N_l^{(0)}$) is kept identical for each $\mathbf{x}_u$ variable set, but it is allowed to change adaptively with generation T. Initially, an empty archive A_0 is created. For each $\mathbf{x}_u$, we perform a lower level NSGA-II operation on the corresponding subpopulation having variables $\mathbf{x}_l$ alone, not to the point where the true lower level Pareto-optimal front is found, but only until a small number of generations at which the specified lower level termination criterion (discussed in Section 5.2) is satisfied. Thereafter, a local search is performed on a few rank-one lower level solutions until the local search termination criterion is met (discussed in Step 3 in Section 5.3). The archive is maintained at the upper level containing solution vectors $(\mathbf{x}_{u_a}, \mathbf{x}_{l_a})$, which are optimal at the lower level and nondominated at the upper level. The solutions in the archive are updated after every lower level NSGA-II call. The members of the lower level population undergoing a local search are lower level optimal solutions and hence are assigned an optimality tag. These local searched solutions $(\mathbf{x}_l)$ are then combined with corresponding $\mathbf{x}_u$ variables and become eligible to enter the archive if it is nondominated when compared to the existing members of the archive. The dominated members in the archive are then eliminated.

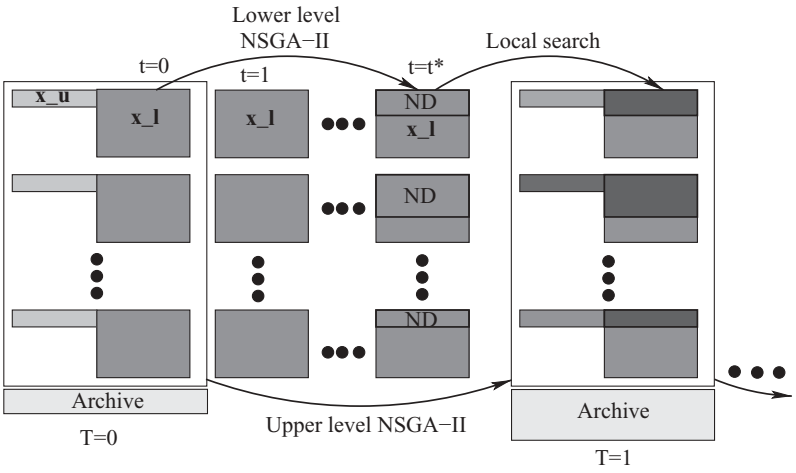

Figure 9: A sketch of the proposed bilevel optimization algorithm.

The solutions obtained from the lower level ($\mathbf{x}_l$) are combined with corresponding $\mathbf{x}_u$ variables and are processed by the upper level NSGA-II operators to create a new upper level population. This process is continued until an upper level termination criterion (described in Section 5.2) is satisfied.

In order to make the proposed algorithm computationally faster, we have used two different strategies: (i) for every upper level variable vector $\mathbf{x}_u$, we do not completely solve the lower level multi-objective optimization problem, thereby not making our approach a nested procedure, and (ii) the subpopulation size and number of generations for a lower level NSGA-II simulation are computed adaptively based on the relative location of $\mathbf{x}_u$ compared to archive solutions, thereby making the overall procedure less parametric and more computationally efficient in terms of overall function evaluations. However, before we present a detailed step-by-step procedure, we discuss the automatic update procedure of population size and termination criteria of the lower level NSGA-II.

5.1 Update of Population Sizes

The upper level population size N_u is kept fixed and is chosen to be proportional to the number of variables. However, the subpopulation size (N_l) for each lower level NSGA-II is sized in a self-adaptive manner. Here we describe the procedure.

In a lower level problem, $\mathbf{x}_l$ is updated by a modified NSGA-II procedure and the corresponding $\mathbf{x}_u$ is kept fixed throughout. Initially, the population size of each lower level NSGA-II ($N_l^{(0)}$) is set depending upon the dimension of lower and upper level variables ($|\mathbf{x}_l|$ and $|\mathbf{x}_u|$, respectively). The number of lower level subpopulations ($n_s^{(0)}$) signifies the number of independent population members for $\mathbf{x}_u$ in a population. Our intention is to set the population sizes ($n_s^{(0)}$ and $N_l^{(0)}$) for $\mathbf{x}_u$ and $\mathbf{x}_l$ to be proportional to their dimensions, yielding

$$\frac{n_s^{(0)}}{N_l^{(0)}} = \frac{|\mathbf{x}_u|}{|\mathbf{x}_l|}. \tag{13}$$

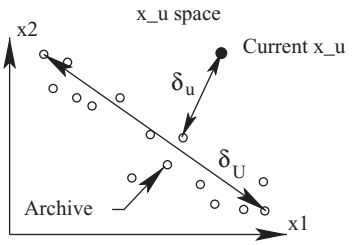

Figure 10: Computation of δ_u and δ_U.

Noting also that $n_s^{(0)} N_l^{(0)} = N_u$, we obtain the following sizing equations:

$$n_s^{(0)} = \sqrt{\frac{|\mathbf{x}_u|}{|\mathbf{x}_l|} N_u}, \tag{14}$$

$$N_l^{(0)} = \sqrt{\frac{|\mathbf{x}_l|}{|\mathbf{x}_u|} N_u}. \tag{15}$$

For an equal number of lower and upper level variables, $n_s^{(0)} = N_l^{(0)} = \sqrt{N_u}$. The above values are set for the initial population only, but are allowed to be modified thereafter in a self-adaptive manner by directly relating the location of the corresponding $\mathbf{x}_u$ variable vector from the points in the archive in the variable space. As shown in Figure 10, first the maximum Euclidean distance (δ_U) in the $\mathbf{x}_u$-space among the members of the archive is computed. Then, the Euclidean distance (δ_u) between the current $\mathbf{x}_u$ vector and the closest archive member is computed. The subpopulation size N_l is then set proportional to the ratio of δ_u and δ_U as follows:

$$N_l = (\text{round}) \frac{\delta_u}{\delta_U} N_l^{(0)}. \tag{16}$$

In order to eliminate the cases with too small or too large population sizes, N_l is restricted between four (due to the need of two binary tournament selection operations to choose two parent solutions for a single recombination event in the NSGA-II) and $N_l^{(0)}$. If the current $\mathbf{x}_u$ variable vector is far away from the archive members, a large number of generations will have to be spent in the corresponding lower level NSGA-II, as dictated by Equation (16).

5.2 Termination Criteria

In a bilevel optimization, it is clear that the lower level optimization must have to be run more often than the upper level optimization, as the former task acts as a constraint to the upper level task. Thus, any judicious and efficient efforts in terminating a lower level optimization can make for a substantial savings in the overall computational effort. For this purpose, we first gauge the difficulty of solving all lower level problems by observing the change in their hypervolume measures in the initial generation ($T = 0$) of the upper level optimization.

The maximum ($H^{\max}$) and minimum ($H^{\min}$) hypervolume is calculated from the lower level nondominated set (with a reference point constructed from the worst objective values of the set) in every τ generations of a lower level run. The H_l-metric is then computed as follows:

$$H_l = \frac{H_l^{\max} - H_l^{\min}}{H_l^{\max} + H_l^{\min}}. \tag{17}$$

If $H_l \leq \epsilon_l$ (a threshold parameter) is encountered, indicating that an adequate convergence in the hypervolume measure is obtained, the lower level NSGA-II simulation is terminated. The number of lower level generations needed to meet the above criterion is calculated for each subpopulation during the initial generation ($T = 0$) of the upper level NSGA-II and an average (denoted here as $t_l^{\max}$) is computed. Thereafter, no subsequent lower level NSGA-II simulations are allowed to proceed beyond t_l generations (derived from $t_l^{\max}$, as calculated below) or the above $H_l \leq \epsilon_l$ is satisfied. We bound the limiting generation (t_l) to be proportional to the distance of current $\mathbf{x}_u$ from the archive, as follows:

$$t_l = (\text{int})\ \frac{\delta_u}{\delta_U} t_l^{\max}. \tag{18}$$

For terminating the upper level NSGA-II, the normalized change in hypervolume measure H_u of the upper level population (as in Equation (17) except that the hypervolume measure is computed in the upper level objective space) is computed in every τ consecutive generations. When $H_u \leq \epsilon_u$ (a threshold parameter) is obtained, the overall algorithm is terminated. We have used $\tau = 10$, $\epsilon_l = 0.1$ (for a quick termination) and $\epsilon_u = 0.0001$ (for a reliable convergence of the upper level problem) for all problems in this study.

Now, we are ready to describe the overall algorithm for a typical generation in a step-by-step format.

5.3 Step-by-Step 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 nondominated rank ND_u corresponding to $\mathbf{F}$ and $\mathbf{G}$, (ii) a crowding distance value CD_u corresponding to $\mathbf{F}$, (iii) a nondominated 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 quantities, for the members stored in the archive A_T, we have also computed (v), a crowding distance value CD_a corresponding to $\mathbf{F}$, and (vi) a nondominated rank ND_a corresponding to $\mathbf{F}$ and $\mathbf{G}$.

Step 1a: **Creation of new $\mathbf{x}_u$:** We apply two binary tournament selection operations on members ($\mathbf{x} = (\mathbf{x}_u, \mathbf{x}_l)$) of P_T using ND_u and CD_u lexicographically. Also, we apply two binary tournament selections on the archive population A_T using ND_a and CD_a lexicographically. Of the four selected members, two participate in the recombination operator based on stochastic events. 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 recombination. The upper level

variable vectors $\mathbf{x}_u$ of the two selected parents are then recombined using the SBX operator (Deb and Agrawal, 1995) to obtain two new vectors of which one is chosen for further processing at random. The chosen vector is then mutated by the polynomial mutation operator (Deb, 2001) to obtain a child vector (say, $\mathbf{x}_u^{(1)}$).

Step 1b: **Creation of new $\mathbf{x}_l$:** First, the population size ($N_l(\mathbf{x}_u^{(1)})$) for the child solution $\mathbf{x}_u^{(1)}$ is determined by Equation (16). The creation of $\mathbf{x}_l$ depends on how close the new variable set $\mathbf{x}_u^{(1)}$ is compared to the current archive, A_T. If $N_l = N_l^{(0)}$ (indicating that the $\mathbf{x}_u$ is away from the archive members), new lower level variable vectors $\mathbf{x}_l^{(i)}$ (for $i = 1, \ldots, N_l(\mathbf{x}_u^{(1)})$) are created by applying selection-recombination-mutation operations on members of P_T and A_T. Here, a parent member is chosen from A_T with a probability $\frac{|A_T|}{|A_T|+|P_T|}$, otherwise a member from P_T is chosen at random. A total of $N_l(\mathbf{x}_u^{(1)})$ child solutions are created by concatenating upper and lower level variable vectors together, as follows: $c_i = (\mathbf{x}_u^{(1)}, \mathbf{x}_l^{(i)})$ for $i = 1, \ldots, N_l(\mathbf{x}_u^{(1)})$. Thus, for the new upper level variable vector $\mathbf{x}_u^{(1)}$, a subpopulation of $N_l(\mathbf{x}_u^{(1)})$ lower level variable vectors are created by genetic operations from P_T and A_T.

However, if the lower level population size ($N_l(\mathbf{x}_u^{(1)})$) is less than $N_l^{(0)}$ (indicating that the variable set $\mathbf{x}_u$ is close to the archive members), a different strategy is used. First, a specific archive member (say, $\mathbf{x}_u^{(a)}$) closest to $\mathbf{x}_u^{(1)}$ is identified. Instead of creating new lower level variable vectors, $N_l(\mathbf{x}_u^{(1)})$ vectors are chosen from the subpopulation to which $\mathbf{x}_u^{(a)}$ belongs. Complete child solutions are created by concatenating upper and lower level variable vectors together. If, however the previous subpopulation does not have $N_l(\mathbf{x}_u^{(1)})$ members, the remaining slots are filled by creating new child solutions by the procedure of the previous paragraph.

Step 2: **Lower level NSGA-II:** For each subpopulation, we now perform a NSGA-II procedure using lower level objectives ($\mathbf{f}$) and constraints ($\mathbf{g}$) for t_l generations (Equation (18)). It is important to reiterate that in each lower level NSGA-II run, the upper level variable vector $\mathbf{x}_u$ is not changed. The selection process is different from that in the usual NSGA-II procedure. If the subpopulation has no member in the current archive A_T, the parent solutions are chosen as usual by the binary tournament selection using ND_l and CD_l lexicographically. If, however, the subpopulation has a member or members that already exist in the archive, only these solutions are used in the binary tournament selection. This is done to emphasize already-found good solutions. The mutation operator is applied as usual. After the lower level NSGA-II simulation is performed on a subpopulation, the members are sorted according to the constrained nondomination level (Deb et al., 2002) and are assigned their nondominated rank (ND_l) and crowding distance value (CD_l) based on lower level objectives ($\mathbf{f}$) and lower level constraints ($\mathbf{g}$).

Step 3: **Local search:** The local search operator is employed next to provide us with a solution that is guaranteed to be on a locally Pareto-optimal front. Since the local search operator can be expensive, we use this operator sparingly. We apply the local search operator to good solutions having the following

properties: (i) it is a nondominated solution in the lower level having $ND_l = 1$, (ii) it is a nondominated solution in the upper level having $ND_u = 1$, and (iii) it does not get dominated by any current archive member, or it is located at a distance less than $\delta_U N_l / N_l^{(0)}$ from any of the current archive members. In the local search procedure, the achievement scalarizing function problem (Wierzbicki, 1980) formulated at the current NSGA-II solution ($\mathbf{x}_l$) with $z_j = f_j(\mathbf{x}_l)$ is solved:

$$\begin{aligned} &\text{Minimize}_{\mathbf{p}} \max_{j=1}^{m} \frac{f_j(\mathbf{p}) - z_j}{f_j^{\max} - f_j^{\min}} + \rho \sum_{j=1}^{m} \frac{f_j(\mathbf{p}) - z_j}{f_j^{\max} - f_j^{\min}}, \\ &\text{subject to } \mathbf{p} \in \mathcal{S}_l, \end{aligned} \tag{19}$$

where $\mathcal{S}_l$ is the feasible search space for the lower level problem. The minimum and maximum function values are taken from the NSGA-II minimum and maximum function values of the current generation. The optimal solution $\mathbf{p}^*$ to the above problem is guaranteed to be a Pareto-optimal solution to the lower level problem (Miettinen, 1999). Here, we use $\rho = 10^{-6}$, which prohibits the local search from converging to a weak Pareto-optimal solution. We use a popular software KNITRO (Byrd et al., 2006; which employs a sequential quadratic programming (SQP) algorithm) to solve the above single objective optimization problem. The KNITRO software terminates when a solution satisfies the Karush-Kuhn-Tucker (KKT) conditions (Reklaitis et al., 1983) with a prespecified error limit. We fix this error limit to 10^{-2} in all problems of this study. The solutions that meet this KKT satisfaction criterion are assigned an optimal tag for further processing. For handling nondifferentiable problems, a nongradient, adaptive step-size based hill-climbing procedure (Nolle, 2006) can be used.

Step 4: **Updating the archive:** The optimally tagged members, if feasible with respect to the upper level constraints (**G**), are then compared with the current archive members. If these members are nondominated when compared to the members of the archive, they become eligible to be added into the archive. The dominated members in the archive are also eliminated, thus the archive always keeps nondominated solutions. We limit the size of archive to $10N_u$. If and when more members are to be entered in the archive, the archive size is maintained to the above limit by eliminating extra members using the crowding distance (CD_a) measure.

Step 5: **Formation of the combined population:** Steps 1 to 4 are repeated until the population Q_T is filled with newly created solutions. Each member of Q_T is now evaluated with **F** and **G**. Populations P_T and Q_T are combined together to form R_T. The combined population R_T is then ranked according to constrained nondomination (Deb et al., 2002) based on upper level objectives (**F**) and upper level constraints (**G**). Solutions are thus assigned a nondominated rank (ND_u) and members within an identical nondominated rank are assigned a crowding distance (CD_u) computed in the **F**-space.

Step 6: **Choosing half the population:** From the combined population R_T of size $2N_u$, half of its members are retained 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 S_T has N_u population members.

Step 7: Upgrading old lower level subpopulations: Each subpopulation of S_T that is not created in the current generation is modified using the lower level NSGA-II procedure (Step 2) applied with **f** and **g**. This step helps progress each lower level population toward its individual Pareto-optimal frontier.

The final population is renamed as P_{T+1}. This marks the end of one generation of the overall H-BLEMO algorithm.

5.4 Algorithmic Complexity

With self-adaptive operations to update population sizes and number of generations, it becomes difficult to compute an exact number of function evaluation (FE) needed in the proposed H-BLEMO algorithm. However, using the maximum allowable values of these parameters, we estimate that the worst case function evaluation is $N_u(2T_u + 1)(t_l^{\max} + 1) + FE_{\text{LS}}$. Here T_u is the number of upper level generations and FE_{LS} is the total function evaluations used by the local search (LS) algorithm. Lower level NSGA-II is able to bring the members close to the Pareto-optimal front which requires a relatively small number of function evaluations for the local search operator. Moreover, toward the end of a simulation, most upper level solutions are close to the archive, thereby requiring a much smaller number of function evaluations than that used in the above expression. As evidence to this fact, Figure 22, shown later, can be referred to for a variation in N_l and t_l for two test problems of this study.

However, it is important to note that the computations needed in the local search may be substantial and any effort to reduce the computational effort will be useful. In this regard, the choice of the local search algorithm and the chosen KKT error limit for terminating the local search will play an important role. Also, the termination parameter for a lower level NSGA-II run (parameter ϵ_l) may also make an important contribution. For both these parameters, we have used reasonable values (KKT error threshold of 0.01 and $\epsilon_l = 0.1$) for a good balance between accuracy and computational efficiency. In Section 6.9, we shall discuss these issues using empirical evidence of independent computations needed in the local search, in the lower level NSGA-II, and in the upper level NSGA-II.

5.5 Review of the Drawbacks in Earlier Approaches

The proposed self-adaptive and hybrid bilevel procedure is quite different from our earlier rather rigid and computationally expensive BLEMO procedures (Deb and Sinha, 2009a,b). We carry out a review of the drawbacks in the earlier approaches and elucidate how these drawbacks have been tackled in H-BLEMO.

1. The previous approaches did not perform a local search at the lower level NSGA-II and hence did not guarantee a lower level solution to be optimal. The methods

relied on the solutions provided by the EMO at the lower level. The H-BLEMO procedure incorporates a local search to the best nondominated lower level NSGA-II solutions so that an indication of the distance and direction of the lower Pareto-optimal front from the current solution can be obtained. Moreover, in later generations, the use of local search in the lower level optimization guarantees convergence to the locally Pareto-optimal front, thereby satisfying the principle of bilevel programming which requires that the final archive solutions are true lower level Pareto-optimal solutions.

2. The earlier approaches were rigid in the sense that every time it made a call to the lower level NSGA-II, a fixed population size and number of generations were used for the lower level run. This was done irrespective of the solutions being close to the front. The approaches did not have a measure to estimate the proximity of the solutions from the lower level Pareto-optimal front, which made it necessary to run the lower level NSGA-II with a sufficient number of generations and with an adequate population size to obtain near Pareto-optimal solutions. Moreover, in the absence of local search, it is necessary to have sufficient population size and number of generations, otherwise the run would provide solutions that are not close to the true Pareto-optimal front at the lower level and hence infeasible at the upper level. This led to unnecessary lower level evaluations which have been avoided in the H-BLEMO. The H-BLEMO procedure is self-adaptive in nature. This allows each lower level NSGA-II to use a different population size and run for a different number of generations adaptively depending on the proximity of the upper level variable vector to current archive solutions. This should allow the H-BLEMO procedure to allocate function evaluations as and where needed.

3. The previous approaches had a fixed population size for the lower level. This made the numbers of upper level variable vectors fixed from one generation to another for these approaches, whereas in H-BLEMO, the use of a variable population size for different lower level NSGA-IIs establishes that the upper level population can hold varying numbers of upper level variable vectors. This allows the H-BLEMO procedure to provide a varying importance between the extent of upper and lower level optimizations, as may be demanded by a problem.

4. The earlier procedures did not have algorithmically sensible termination criteria for the upper or the lower level NSGA-II and had to run for a fixed number of generations. In H-BLEMO, a hypervolume-based termination criterion was used where both lower and upper levels terminate, based on the dynamic performance of the algorithms. The criterion ensures the termination of each NSGA-II whenever the corresponding nondominated front has stabilized, thereby avoiding unnecessary function evaluations.

6 Results of Test Problems

We use the following standard NSGA-II parameter values in both lower and upper levels for all problems of this study: Crossover probability of 0.9, distribution index for SBX of 15, mutation probability of 0.1, distribution index for the polynomial mutation of 20. The upper level population size is set proportional to the total number of variables (n): $N_u = 20n$. As described in the algorithm, all other parameters, including the lower level population size termination criteria, are set in a self-adaptive manner during the

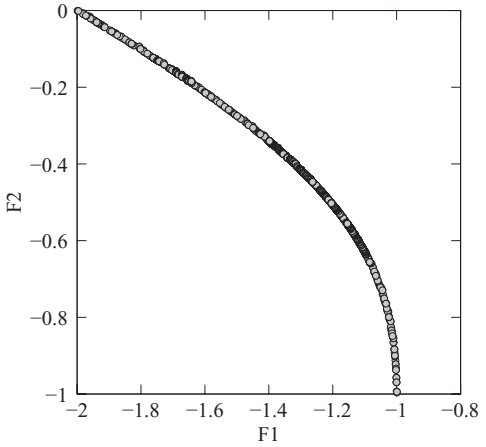

Figure 11: Final archive solutions for problem TP1.

optimization run. In all cases, we have used 21 different simulations starting from different initial populations and show the 0, 50, and 100% attainment surfaces (Fonseca and Fleming, 1996) to describe the robustness of the proposed procedure.

6.1 Problem TP1

This problem has three variables (one for upper level and two for lower level). Thus, we use $N_u = 60$ population members. Figure 11 shows the final archive members of a single run on the upper level objective space. It can be observed that our proposed procedure is able to find solutions on the true Pareto-optimal front. Conditions for the exact Pareto-optimal solutions are given in Equation (3). For each of the obtained solutions, we use the upper level variable (y) value to compute lower level optimal variable (x_1^* and x_2^*) values using the exact conditions for Pareto-optimality and compare the values with H-BLEMO solutions. The average error $\sum_{i=1}^{2}(x_i - x_i^*)^2/2$ for each obtained solution is computed and then averaged over all archive solutions. This error value for our H-BLEMO is found to be 7.0318×10^{-5}, whereas the same error value computed for the solutions reported with our earlier algorithm (Deb and Sinha, 2009a) is found to be 2.2180×10^{-3}. The closer adherence to optimal variable values with H-BLEMO indicates a better performance of our hybrid algorithm. The minimum, median, and worst number of function evaluations needed over 21 different runs of H-BLEMO are 567,848, 643,753, and 716,780, respectively. Although for a three-variable problem, these numbers may seem too large, one needs to realize that the bilevel problems have one optimization algorithm nested into another and Pareto-optimality for the lower level problem is essential for an upper level solution. In comparison, our previous algorithm (Sinha and Deb, 2009) required 1,030,316, 1,047,769, and 1,065,935 function evaluations for best, median, and worst performing runs. Despite using about 40% less function evaluations, our H-BLEMO also finds more accurate solutions.

The attainment surfaces obtained for the archive solutions over 21 runs are shown in Figure 12. All three surfaces are so close to each other that they are difficult to distinguish from one another. This indicates the robustness of the procedure. The hypervolume

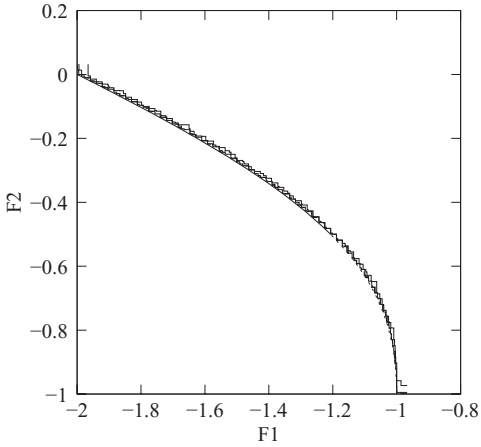

Figure 12: Attainment surfaces (0%, 50%, and 100%) for problem TP1 after 21 runs.

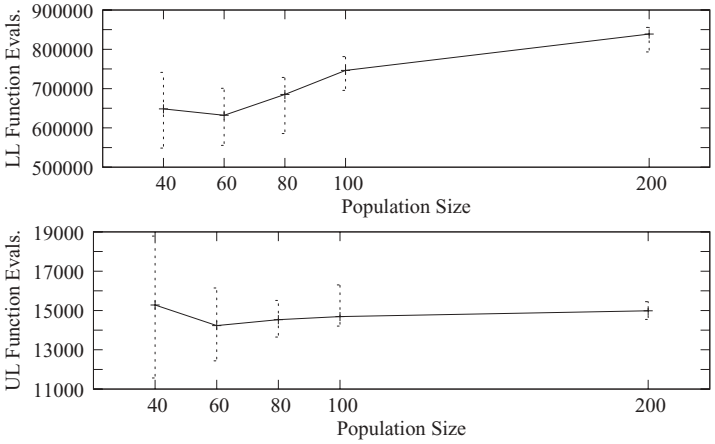

Figure 13: Average lower and upper level function evaluations with different population sizes (N_u) for problem TP1. The average has been taken for 21 runs. On average, $N_u = 60$ is found to perform the best.

values are computed after normalizing the upper level objective values by their minimum and maximum values. The hypervolumes for the 0%, 50%, and 100% attainment surfaces are 0.3583, 0.3678, and 0.3700, respectively. The difference in the hypervolume value over 21 runs is only about 3%.

In order to investigate the effect of N_u on the performance of the algorithm, next, we use different N_u values but maintain an identical termination criterion. Figure 13 shows the function evaluations needed for lower (including the local search) and upper level optimization tasks for different N_u values ranging from 40 to 200. It is clear from the figure that a population size of $N_u = 60$ (which we used in Figures 11 and 12) performs the best, on average, in both lower and upper level optimization tasks.

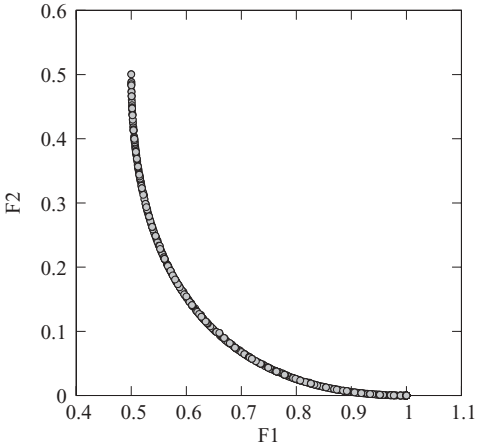

Figure 14: Final archive solutions for problem TP2.

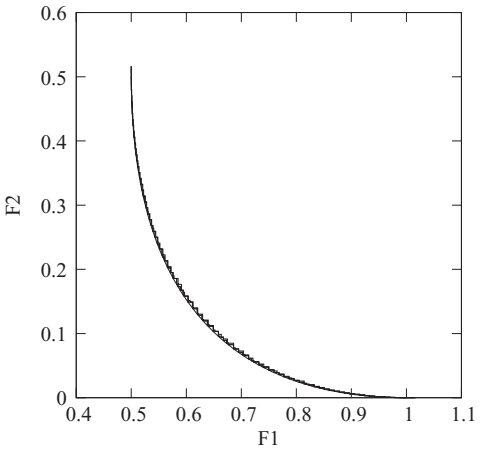

Figure 15: Attainment surfaces (0%, 50%, and 100%) for problem 2 after 21 runs for problem TP2.

6.2 Problem TP2

The second test problem has $n = 15$ variables. Thus, we use $N_u = 300$. Figure 14 shows the final archive population of a typical run. The attainment surface plot in Figure 15 shows that the proposed algorithm is fairly robust in all 21 different simulations. The algorithm finds an almost identical front close to the true Pareto-optimal front in multiple runs. The hypervolumes for the obtained attainment surfaces are 0.8561, 0.8582, and 0.8589, making a maximum difference of only 0.3%. A comparison of our current local search based algorithm with our previously proposed BLEMO procedure (Sinha and Deb, 2009) in terms of an error measure (as discussed for problem TP1) from the exact Pareto-optimal solutions indicates a smaller error for our current approach. Despite the

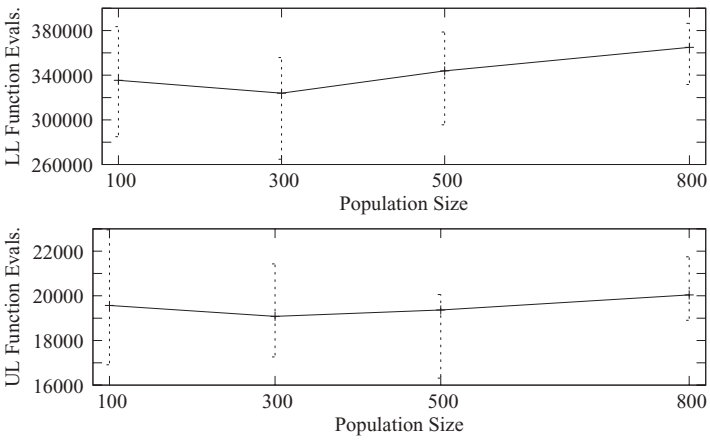

Figure 16: Average lower and upper level function evaluations with different population sizes for problem TP2 indicates $N_u = 300$ is the best choice. Twenty-one runs are performed in each case.

use of 15 variables here, as opposed to seven variables used in the previous study, the error in the current approach is 7.920×10^{-6}, compared to 11.158×10^{-6} in the previous study. In terms of the median performance, H-BLEMO requires 338,232 function evaluations for the 15-variable problem, as opposed to 771,404 function evaluations needed by our previous approach (Sinha and Deb, 2009) on a seven-variable version of the problem.

Interestingly, the function evaluation plots (Figure 16) for the lower and upper level tasks indicate that the proposed $N_u = 20n$ (300 in this case) works the best in terms of achieving a similar performance with the smallest number of function evaluations.

For brevity and space limitations, we do not show results on TP3, but similar results are found for this problem as well. We do show results on TP4 in order to highlight a comparison of H-BLEMO with a previous study.

6.3 Problem TP4

This problem is linear and has five variables in total. Thus, we use $N_u = 100$. Figure 17 shows the final archive which follows a linear relationship among upper level objectives. This multi-objective bilevel problem was solved in the original study (Zhang et al., 2007) by converting two objectives into a single objective by the weighted-sum approach. The reported solution is marked on the figure with an asterisk. It is interesting to note that this solution is one of the extreme solutions of our obtained front. For a problem having a linear Pareto-optimal front, the solution of a weighted-sum optimization approach is usually one of the extreme points. To this account, this result signifies the accuracy and efficiency of the H-BLEMO procedure.

Figure 18 shows three attainment surface plots which are close to each other, as they are in the previous two problems. The hypervolumes for the obtained attainment surfaces are 0.5239, 0.5255, and 0.5260, with a maximum difference of only 0.4%, indicating the robustness of our procedure.

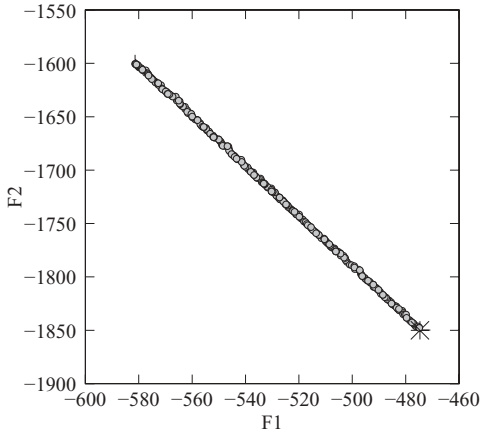

Figure 17: Final archive solutions for problem TP4. The point marked with an asterisk is taken from Zhang et al. (2007).

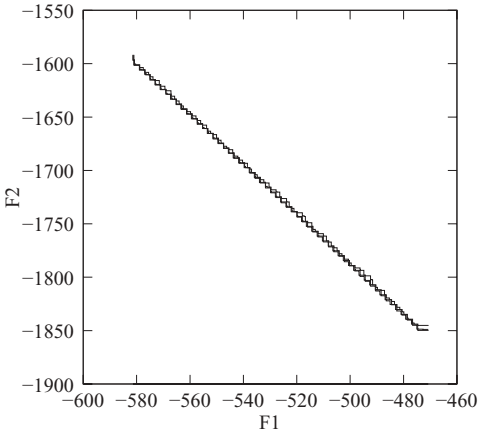

Figure 18: Attainment surfaces (0%, 50%, and 100%) for problem TP4 after 21 runs.

6.4 Problem DS1

This problem has 10 upper and 10 lower level variables. Thus, an overall population of size $N_u = 400$ is used here. For this problem, we first consider $\tau = 1$. Figure 19 shows the obtained archive solutions for a typical run. It is worth mentioning here that this problem was possible to be solved up to only six variables (in total) by our earlier fixed BLEMO approach (Deb and Sinha, 2009a). But here with our hybrid and self-adaptive approach, we are able to solve the 20-variable version of the problem. Later, we present results with 40 variables as well for this problem.

The attainment surface plots in Figure 20 further show the robustness of the proposed algorithm. The hypervolumes for the obtained attainment surfaces are 0.7812, 0.7984, and 0.7992, with a maximum difference of about 2%. The parametric study with

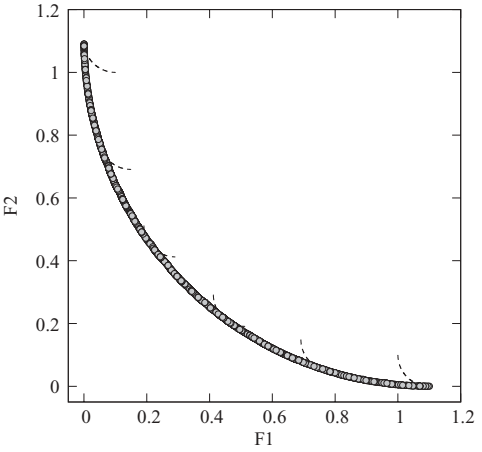

Figure 19: Final archive solutions for problem DS1.

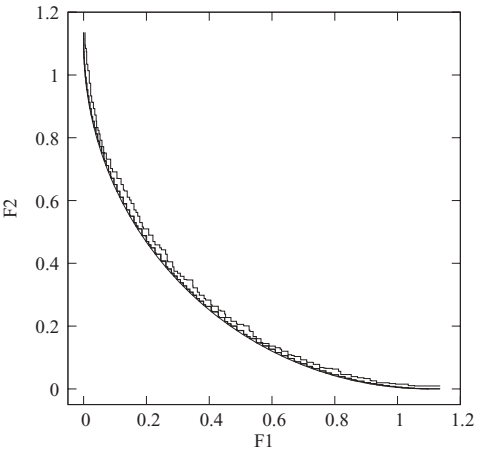

Figure 20: Attainment surfaces (0%, 50%, and 100%) for problem DS1 after 21 runs.

N_u in Figure 21 shows that $N_u = 400$ is the best choice in terms of achieving a similar performance on the hypervolume measure using the smallest number of function evaluations, thereby supporting our setting $N_u = 20n$.

Since this problem is more difficult compared to the previous problems (TP1, TP2, and TP4), we investigate the effect of self-adaptive changes in lower level NSGA-II parameters (N_l and t_l) with the generation counter. In the left side plot of Figure 22, we show the average value of these two parameters for every upper level generation (T) from a typical simulation run. It is interesting to note that, starting with an average of 20 members in each lower level subpopulation, the number reduces with each generation counter, meaning that smaller population sizes are needed for later lower level simulations. The average subpopulation size reduces to its lower permissible value of

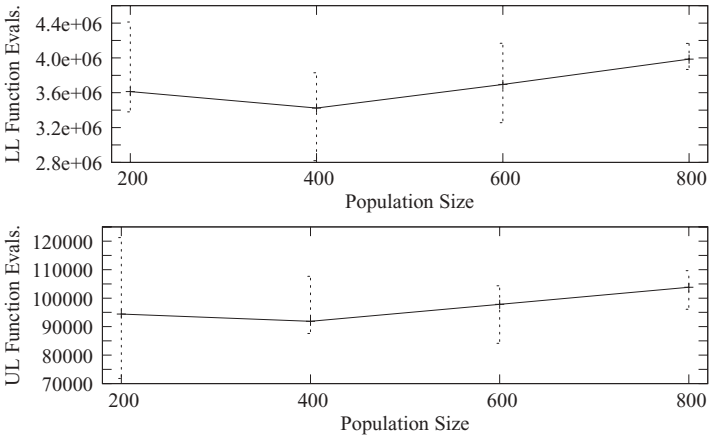

Figure 21: Average lower and upper level function evaluations with different population sizes for problem DS1 indicates an optimal population size of $N_u = 400$. The average has been taken for 21 runs.

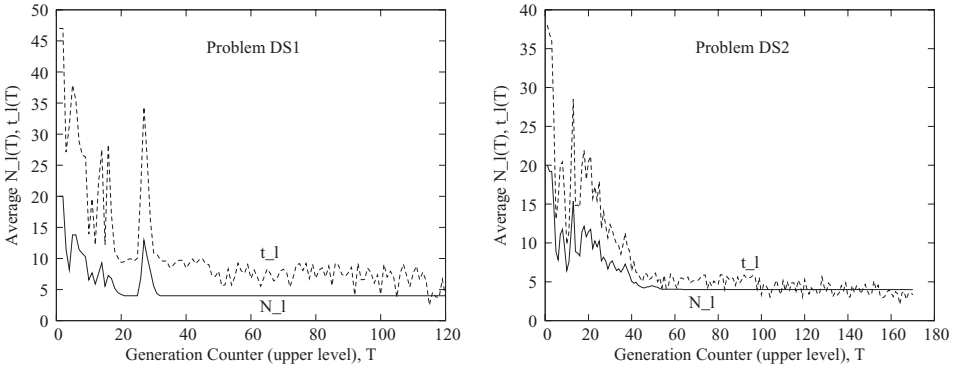

Figure 22: Variation of N_l and t_l with upper level generation for problems DS1 (left figure) and DS2 (right figure). The algorithm adapts these two important parameters automatically.

four in 32 generations. An occasional increase in average N_l indicates that an upper level variable vector may have been found near a previously undiscovered region of the Pareto-optimal front. The hybrid algorithm increases its lower level population to explore the region better in such occasions.

The variation in the number of lower level generations (t_l) before termination also follows a similar trend, except that at the end only 3–9 generations are required to fulfill the lower level termination condition, although initially as many as 47 generations were needed. Interestingly, whenever there is a surge in N_l, a similar surge is observed for t_l as well. This supports our argument about possible discovery of new and isolated $\mathbf{x}_u$ variable vectors. Our previous implementation (Deb and Sinha, 2009a) used predefined fixed values for N_l and t_l throughout, thereby requiring an unnecessarily

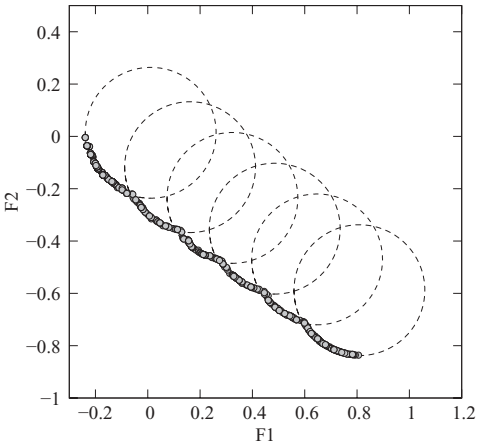

Figure 23: Final archive solutions for problem DS2.

large computational effort. In our current hybrid algorithm, we reduce the computational burden by using the self-adaptive updates of the two most crucial parameters involving computations in the lower level optimization task.

6.5 Problem DS2

This problem also has 20 variables in total, thereby motivating us to use $N_u = 400$. Here too, we use $\tau = 1$ at first and postpone a discussion on the difficult version of the problem with $\tau = -1$ later. Figure 23 shows the final archive solutions from a typical run, indicating the efficiency of our procedure. Our earlier BLEMO algorithm (Deb and Sinha, 2009a) could only solve at most a four-variable version of this problem. The hypervolumes for the obtained attainment surfaces are 0.5776, 0.6542, and 0.6629, respectively. The 0%, 50%, and 75% attainment surfaces shown in Figure 24 are very close to each other, indicating that at least 75% of the runs are close to each other. The gap between 75% and 100% attainment surfaces indicates that a few solutions are found to be not so close to the true Pareto-optimal frontier in this problem.

Figure 25 confirms that $N_u = 400$ is the best choice of N_u to achieve a similar hypervolume measure with the smallest number of overall function evaluations. The right side plot of Figure 22 shows that N_l and t_l start with large values but drop to small values adaptively to make the optimization process efficient and computationally fast.

6.6 Problem DS3

This problem has 20 variables. Thus, we have used $N_u = 400$. Figure 26 shows the final archive population and Figure 27 shows corresponding attainment surface plot. Our earlier algorithm was able to solve at maximum an eight-variable version of this problem. The hypervolumes for the attainment surfaces are 0.5528, 0.5705, and 0.5759, respectively. All 21 runs find the entire Pareto-optimal front with a maximum difference in hypervolume value of about 4%.

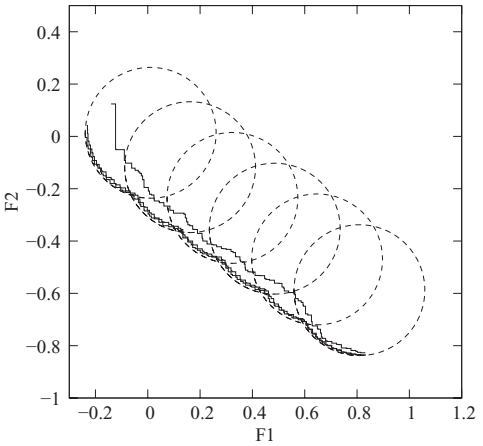

Figure 24: Attainment surfaces (0%, 50%, 75%, and 100%) for problem DS2 after 21 runs.

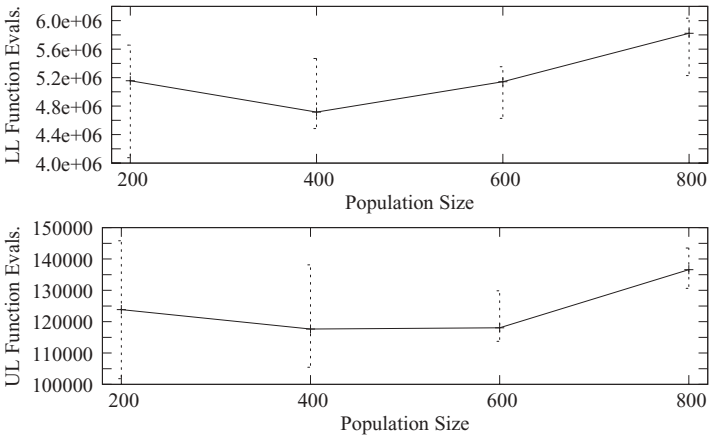

Figure 25: Average lower and upper level function evaluations with different population sizes for problem DS2 indicate $N_u = 400$ is the best choice. Twenty-one runs are performed in each case.

6.7 Problem DS4

This problem is considered for 10 variables; thus we use $N_u = 200$. Figures 28 and 29 show the archive population and the attainment surface for this problem. The hypervolumes for the obtained attainment surfaces are 0.5077, 0.5241, and 0.5264, respectively. The maximum difference in hypervolume measures in 21 runs is only about 3.6%, indicating the robustness of the proposed procedure.

6.8 Problem DS5

This problem is also considered with 10 variables, for which we used $N_u = 200$. Figure 30 shows the final archive population for a typical run. Figure 31 shows the corresponding

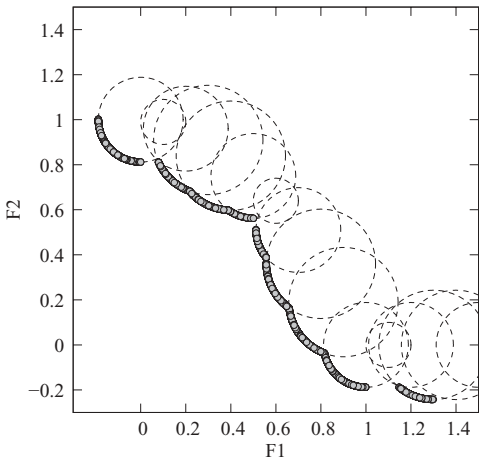

Figure 26: Final archive solutions for problem DS3.

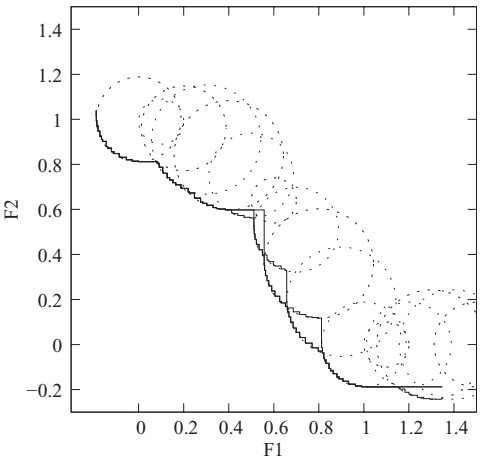

Figure 27: Attainment surfaces (0%, 50%, and 100%) for problem DS3 after 21 runs.

attainment surface plots, which are very close to each other, indicating the efficacy of the procedure. The hypervolumes for the obtained attainment surfaces are 0.5216, 0.5281, and 0.5308, respectively. The difference in hypervolume is only 1.7% for this problem.

6.9 Computational Effort

Next, we investigate two aspects related to the computational issues. First, we record the total function evaluations needed by the overall algorithm to achieve the specified termination criterion ($\epsilon_u = 0.0001$) and the same needed exclusively for the lower level optimization task, which includes the local search. Table 1 shows these values for the best, median, and worst of 21 simulation runs for all eight problems. It is clear

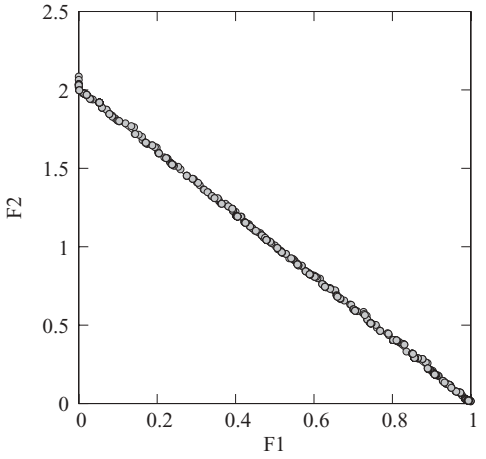

Figure 28: Final archive solutions for problem DS4.

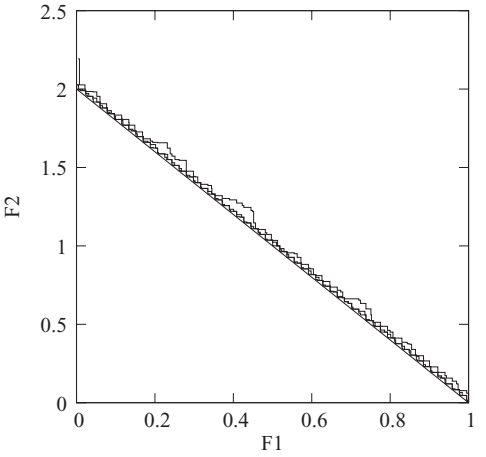

Figure 29: Attainment surfaces (0%, 50%, and 100%) for problem DS4 after 21 runs.

from the table that most of the computational effort is spent in the lower level solution evaluations. Despite this effort being different from a nested algorithm in not solving a lower level problem all the way for every upper level solution, the nature of bilevel programming problems demands that the lower level optimization task must be emphasized. The use of the archive method in sizing lower level subpopulations in a self-adaptive manner and the use of a coarse terminating condition for the lower level optimization task enabled our algorithm to use a comparatively smaller number of function evaluations than what would be needed in a nested algorithm. We compare the function evaluations of H-BLEMO with a nested bilevel optimization algorithm later in Section 9 to illustrate this fact.

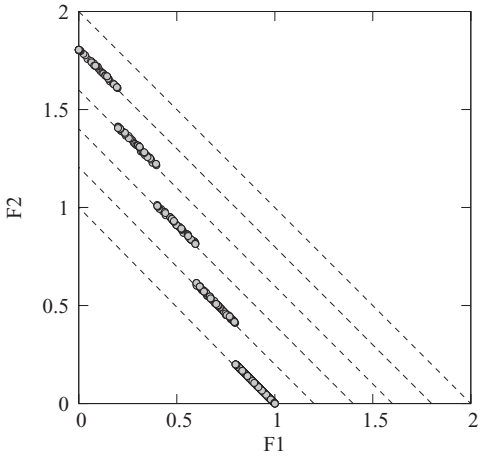

Figure 30: Final archive solutions for problem DS5.

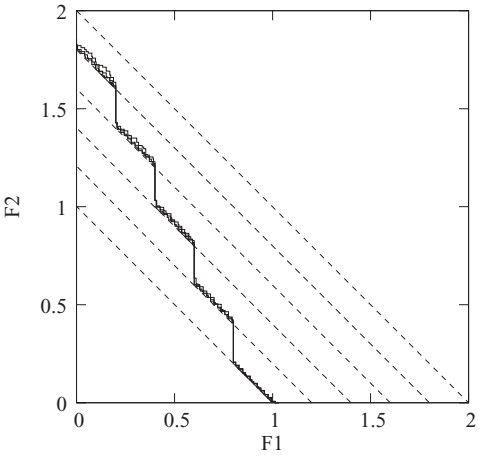

Figure 31: Attainment surfaces (0%, 50%, and 100%) for problem DS5 after 21 runs.

In order to investigate the computational effort needed in the local search operator, we record the average function evaluations in the local search procedure and in the overall lower level optimization task. Table 2 presents the results. The local search effort varies from problem to problem. However, it contributes to more than half the computations in the lower level optimization task in most problems. Thus, putting both tables together, we conclude that the effort of the local search is about 50% of the overall computational effort of the proposed H-BLEMO algorithm. Of course, this quantity depends on the chosen termination criterion for the lower level, upper level, and the local search optimization tasks. Nevertheless, the termination conditions chosen for this study seemed to be adequate for the overall algorithm to work on all the problems of this paper and an effort of about 50% for achieving guaranteed convergence (with respect

Table 1: Total function evaluations for the upper and lower level (21 runs). The lower level function evaluations include the evaluations of local search as well.

		Best		Median		Worst	
Pr. No.	# var.	Total LL FE	Total UL FE	Total LL FE	Total UL FE	Total LL FE	Total UL FE
TP1	3	555,410	12,438	629,590	14,163	700,634	16,146
TP2	15	264,675	17,262	319,499	18,733	355,904	21,429
TP4	5	1,624,658	37,218	1,927,022	39,960	2,036,208	41,570
DS1	20	2,819,770	87,582	3,423,544	91,852	3,829,812	107,659
DS2	20	4,484,580	105,439	4,695,352	116,605	5,467,633	138,107
DS3	20	3,970,411	112,560	4,725,596	118,848	5,265,074	125,438
DS4	10	1,356,598	38,127	1,435,344	53,548	1,675,422	59,047
DS5	10	1,666,953	47,127	1,791,511	56,725	2,197,470	71,246

Table 2: Function evaluations needed by the local search.

Pr. No.	Avg. Local search FE	FE per call of local search	Avg. Lower level FE	% Local search FE
TP1	91,382.2	19.89	640,696.1	0.14
TP2	232,426.9	108.59	324,093.8	0.72
TP4	504,084.2	43.93	1,940,248.9	0.26
DS1	1,996,769.6	77.90	3,744,071.8	0.53
DS2	2,620,456.7	97.79	5,039,594.9	0.52
DS3	2,867,432.3	75.43	4,841,572.7	0.59
DS4	1,020,262.2	60.24	1,472,883.7	0.69
DS5	1,427,828.6	68.56	1,893,837.5	0.75

to satisfying KKT conditions) by the local search may seem a reasonable proposition. However, future efforts may focus on devising quicker local search and lower level optimization tasks for reducing the overall computational effort. As we discuss next, the local search operator may have a larger role to play than simply guaranteeing convergence to a locally Pareto-optimal frontier.

7 Introducing Further Difficulties in DS1 and DS2

In this section, we use $\tau = -1$ in 20-variable DS1 and DS2 problems (refer to Equations (7) and (9), respectively). As discussed earlier, this choice makes a conflicting scenario in the working principles between upper and lower level optimization tasks. In order to demonstrate the difficulties caused by this setting, we consider three different algorithms: (i) A1: H-BLEMO procedure with local search, (ii) A2: H-BLEMO procedure without local search, and (iii) A3: BLEMO procedure (Deb and Sinha, 2009a). All parameter values are the same as before and are maintained for all three algorithms. To compare the performances, we first identify the nadir point in each problem and then compute the true hypervolume measure H^* by computing 10,000 well-distributed Pareto-optimal points. Thereafter, we record a normalized hypervolume measure $DH(T) = (H(T) - H^*)/H^*$ at each upper level generation T from the hypervolume ($H(T)$) computed using the true nadir point as the reference point. Note that if the $DH(T)$ value reaches zero, the corresponding algorithm can be said to reach

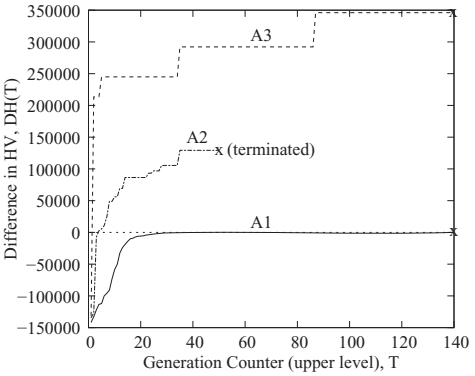

Figure 32: Difference in hypervolume from ideal $DH(T)$ with upper level generation counter T for problem DS1 using three algorithms. Only algorithm A1 (H-BLEMO) reaches the Pareto-optimal front by making $DH(T) = 0$.

the true Pareto-optimal front. A negative value of $DH(T)$ indicates that the obtained set of solutions lies above the true Pareto-optimal front so that the hypervolume $H(T)$ is smaller than H^*. This is a usual scenario of a multi-objective optimization run, where solutions are usually worse than the true Pareto-optimal front in the beginning of a run (having negative $DH(T)$ values) and then the solutions come closer to the Pareto-optimal front with each generation.

However, in the case of $\tau = -1$ for both DS1 and DS2 problems, if the lower level problem is unable to find the true Pareto-optimal points, the corresponding solutions lie much below the true Pareto-optimal front in the upper level objective space. Thus, the hypervolume measure $H(T)$ in the upper level objective space will be very large, but this may be construed as a case in which the obtained solutions are infeasible. Since $H(T)$ will be larger than H^* in this case, the $DH(T)$ value will be positive.

Figures 32 shows the $DH(T)$ measure for all three algorithms with generation counter T for problem DS1. It is clear that only our proposed hybrid algorithm (A1) is able to find the true Pareto-optimal front, by approaching a $DH(T)$ value of zero from an initial negative value. Two other algorithms get stuck to a large positive $DH(T)$ value, indicating that the obtained set of solutions lie in the infeasible region beneath the true Pareto-optimal front in the upper level objective space. Our hybrid algorithm without the local search (A2) is somewhat better than our previous algorithm (A3). A similar performance is also observed for problem DS2, as shown in Figure 33. These scenarios clearly indicate the importance of using the local search approach in the lower level optimization task.

Having shown the importance of the local search and self-adaptive update of NSGA-II parameters for the lower level optimization task, we now investigate the proposed H-BLEMO algorithm's extent of sensitivity to the parameter τ. We compare the function evaluations for both DS1 and DS2 problems with $\tau = +1$ and $\tau = -1$ in Table 3. It is interesting to note that both problems are harder with $\tau = -1$, but the H-BLEMO algorithm is not overly sensitive to the difficulty caused by the discrepancy in search directions in lower and upper level problems achieved with $\tau = -1$. For the median performance, DS1 and DS2 require an increase in overall function evaluations of 5.1%

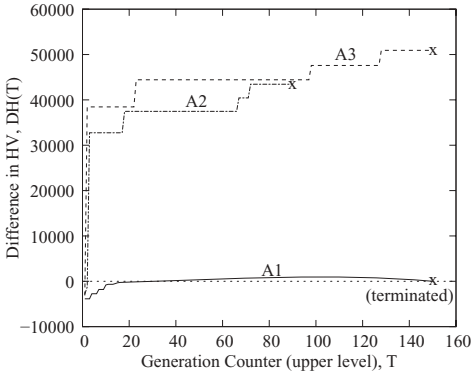

Figure 33: Difference in hypervolume from ideal $DH(T)$ with upper level generation counter T for problem DS2 using three algorithms. Only algorithm A1 (H-BLEMO) reaches the Pareto-optimal front by making $DH(T) = 0$.

Table 3: Comparison of function evaluations for $\tau = -1$ and $\tau = +1$ cases with the H-BLEMO algorithm.

Problem Number	Best		Median		Worst	
	Total LL FE	Total UL FE	Total LL FE	Total UL FE	Total LL FE	Total UL FE
DS1 ($\tau = +1$)	2,819,770	87,582	3,423,544	91,852	3,829,812	107,659
DS1 ($\tau = -1$)	3,139,381	92,624	3,597,090	98,934	4,087,557	113,430
DS2 ($\tau = +1$)	4,484,580	105,439	4,695,352	116,605	5,467,633	138,107
DS2 ($\tau = -1$)	4,796,131	112,563	4,958,593	122,413	5,731,016	144,428

and 5.6%, respectively. However, an absence of local search or our previous BLEMO algorithm is unable to solve the $\tau = -1$ version of both problems (Figures 32 and 33).

8 Scalability Study

In this section, we consider DS1 and DS2 (with $\tau = 1$) and show the scalability of our proposed procedure up to 40 variables. For this purpose, we consider four different variable sizes: $n = 10$, 20, 30, and 40. Based on parametric studies performed on these problems in Section 6, we set $N_u = 20n$. All other parameters are automatically set in a self-adaptive manner during the course of a simulation, as before.

Figure 34 shows the variation of function evaluations for obtaining a fixed termination criterion on normalized hypervolume measure ($H_u < 0.0001$) calculated using the upper level objective values for problem DS1. Since the vertical axis is plotted in a logarithmic scale and the relationship is found to be sublinear, the hybrid methodology performs better than an exponential algorithm. The break-up of computations needed in the local search, lower level NSGA-II, and upper level NSGA-II indicate that the majority of the computations are spent in the lower level optimization task. This is an important insight to the working of the proposed H-BLEMO algorithm and suggests that further work should be focused on making the lower level optimization more computationally efficient. Figure 35 shows a similar outcome for problem DS2,

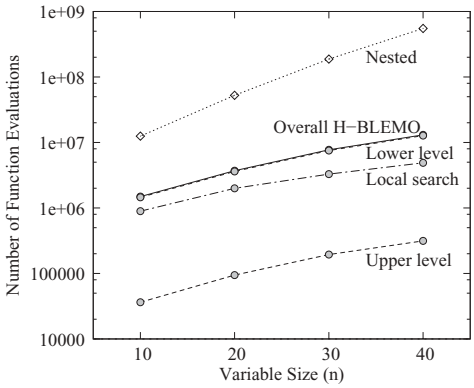

Figure 34: Variation of function evaluations with problem size n for DS1.

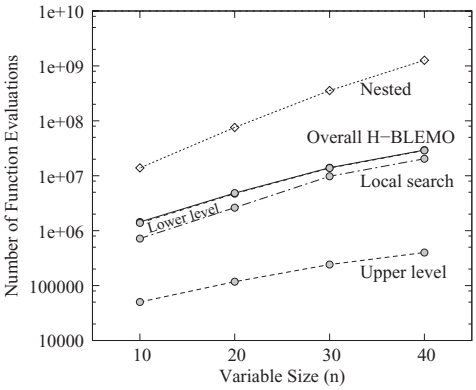

Figure 35: Variation of function evaluations with problem size n for DS2.

but a comparison with problem DS1 indicates that DS2 is more difficult to solve with an increase in problem size than is DS1.

9 Comparison with a Nested Algorithm

We have argued before that by allowing lower level and upper level NSGA-IIs to proceed partially in tandem, we have created a computationally efficient and accurate algorithm that progresses toward the true Pareto-optimal front on a number of difficult problems (Section 6). The algorithm is even found to converge in problems in which there is a conflict between upper and lower level problems (Section 7). The proposed algorithm is also found to solve scaled-up problems up to 40 real-parameter variables (Section 8). In this section, we compare the proposed H-BLEMO algorithm with an efficient yet nested bilevel optimization algorithm using the NSGA-II-cum-local-search procedure. This algorithm uses a fixed population structure, but for every $\mathbf{x}_u$, the lower level optimization is terminated by performing a local search to all nondominated

Table 4: Comparison of function evaluations needed by a nested algorithm and by H-BLEMO on problems DS1 and DS2. The results from 21 runs are summarized.

Problem DS1						
		Median			Min. overall	Max. overall
n	Algorithm	Lower FE	Upper FE	Overall FE	FE	FE
10	Nested	12,124,083	354,114	12,478,197	11,733,871	14,547,725
10	Hybrid	1,454,194	36,315	1,490,509	1,437,038	1,535,329
20	Nested	51,142,994	1,349,335	52,492,329	42,291,810	62,525,401
20	Hybrid	3,612,711	94,409	3,707,120	2,907,352	3,937,471
30	Nested	182,881,535	4,727,534	187,609,069	184,128,609	218,164,646
30	Hybrid	7,527,677	194,324	7,722,001	6,458,856	8,726,543
40	Nested	538,064,283	13,397,967	551,462,250	445,897,063	587,385,335
40	Hybrid	12,744,092	313,861	13,057,953	10,666,017	15,146,652

Problem DS2						
		Median			Min. overall	Max. overall
n	Algorithm	Lower FE	Upper FE	Overall FE	FE	FE
10	Nested	13,408,837	473,208	13,882,045	11,952,650	15,550,144
10	Hybrid	1,386,258	50,122	1,436,380	1,152,015	1,655,821
20	Nested	74,016,721	1,780,882	75,797,603	71,988,726	90,575,216
20	Hybrid	4,716,205	117,632	4,833,837	4,590,019	5,605,740
30	Nested	349,242,956	5,973,849	355,216,805	316,279,784	391,648,693
30	Hybrid	13,770,098	241,474	14,011,572	14,000,057	15,385,316
40	Nested	1,248,848,767	17,046,212	1,265,894,979	1,102,945,724	1,366,734,137
40	Hybrid	28,870,856	399,316	29,270,172	24,725,683	30,135,983

solutions of the final lower level NSGA-II population. The termination criterion for lower and upper level NSGA-IIs and for the local search procedure are identical to the results found in H-BLEMO algorithm. Since for every $\mathbf{x}_u$, we find a set of well-converged and well-distributed lower level Pareto-optimal solutions, this approach is truly a nested bilevel optimization procedure.

For the simulation with this nested algorithm on DS1 and DS2 problems ($\tau = 1$), we use $N_u = 400$. In order to make a fair comparison, we use the same subpopulation size N_l that was used in the very first iteration of our H-BLEMO algorithm using Equation (15). The number of generations for the lower level NSGA-II is kept fixed for all upper level generations to that computed by Equation (14) in the initial generation. Similar archiving strategy and other NSGA-II parameter values are used as before. Table 4 shows the comparison of overall function evaluations needed by the nested algorithm and by the hybrid BLEMO algorithm. The table shows that for both problems, the nested algorithm takes at least one order of magnitude more function evaluations to find a set of solutions having an identical hypervolume measure. The difference between our proposed algorithm and the nested procedure widens with an increase in number of decision variables. The median number of function evaluations is also plotted in Figures 34 and 35. The computational efficacy of our proposed hybrid approach and difference of our approach from a nested approach are clearly evident from these plots.

10 Conclusions

Bilevel programming problems appear commonly in practice; however, due to complications associated in solving them, often they are treated as single-level optimization

problems by adopting approximate solution principles for the lower level problem. Although single-objective bilevel programming problems are studied extensively, there does not seem to be enough emphasis on multi-objective bilevel optimization studies. This paper has made a significant step in presenting past key research efforts, identifying insights for solving such problems, suggesting scalable test problems, and implementing a viable hybrid evolutionary-cum-local-search algorithm. The proposed algorithm is also self-adaptive, allowing an automatic update of the key parameters from generation to generation. Simulation results on eight different multi-objective bilevel programming problems and their variants, and a systematic overall analysis, amply demonstrate the usefulness of the proposed approach. Importantly, due to the multi-solution nature of the problem and intertwined interactions between both levels of optimization, this study helps to showcase the importance of evolutionary algorithms in solving such complex problems.

The study of multi-objective bilevel problem solving methodologies elevates every aspect of an optimization effort to a higher level, thereby making them interesting and challenging to pursue. Although formulation of theoretical optimality conditions is possible and has been suggested, viable methodologies to implement them in practice are challenging. Although a nested implementation of lower level optimization from the upper level is an easy fix-up and has been attempted by many researchers, suitable practical algorithms coupling the two levels of optimization in a computationally efficient manner is not easy and definitely challenging. Developing hybrid bilevel optimization algorithms involving various optimization techniques, such as evolutionary, classical, mathematical, simulated annealing methods, and so on, are possible in both levels independently or synergistically, allowing a plethora of implementational opportunities. This paper has demonstrated one such implementation involving evolutionary algorithms, a classical local search method, and a mathematical optimality condition for termination, but certainly many other ideas are possible and must be pursued urgently.

Every upper level Pareto-optimal solution comes from a lower level Pareto-optimal solution set. Thus, a decision-making technique for choosing a single preferred solution in such scenarios must involve both upper and lower level objective space considerations, which may require and give birth to new and interactive multiple criterion decision making (MCDM) methodologies. Finally, a successful implementation and understanding of multi-objective bilevel programming tasks should motivate us to understand and develop higher level (say, three- or four-level) optimization algorithms, which should also be of great interest to computational science due to the hierarchical nature of the systems approach often followed in complex computational problem solving tasks today.

Acknowledgments

The authors wish to thank the Academy of Finland and the Foundation of Helsinki School of Economics (under grant 118319) for their support of this study.

References

Abass, S. A. (2005.) Bilevel programming approach applied to the flow shop scheduling problem under fuzziness. *Computational Management Science*, 4(4), 279–293.

Alexandrov, N., and Dennis, J. E. (1994). Algorithms for bilevel optimization. In *AIAA/NASA/USAF/ISSMO Symposium on Multidisciplinary Analyis and Optimization*, pp. 810–816.

Bard, J. F. (1998). *Practical bilevel optimization: Algorithms and applications*. Dordrecht, The Netherlands: Kluwer.

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

Byrd, R. H., Nocedal, J., and Waltz, R. A. (2006). *KNITRO: An integrated package for nonlinear optimization* (pp. 35–59). Berlin: Springer-Verlag.

Calamai, P. H., and Vicente, L. N. (1994). Generating quadratic bilevel programming test problems. *ACM Transactions on Mathematical 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*. New York: Wiley.

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

Deb, K., and Sinha, A. (2009a). Constructing test problems for bilevel evolutionary multi-objective optimization. In *Proceedings of the Congress on Evolutionary Computation (CEC-2009)*. Piscataway, NJ: IEEE Press. (Also KanGAL Report No. 2008010.)

Deb, K., and Sinha, A. (2009b). Solving bilevel multi-objective optimization problems using evolutionary algorithms. In *Proceedings of Evolutionary Multi-Criterion Optimization (EMO-2009)*, pp. 110–124.

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 A. Abraham, L. Jain, and R. Goldberg (Eds.), *Evolutionary Multiobjective Optimization* (pp. 105–145). London: Springer-Verlag.

Dempe, S. (2002). *Foundations of bilevel programming*. Dordrecht, The Netherlands: Kluwer.

Dempe, S., Dutta, J., and Lohse, S. (2006). Optimality conditions for bilevel programming problems. *Optimization*, 55(5–6), 505–524.

Dimitriou, L., Tsekeris, T., and Stathopoulos, A. (2008). Genetic computation of road network design and pricing Stackelberg games with multi-class users. In *Proceedings of EvoWorkshops*, pp. 669–678. (Also LNCS 4974.)

Eichfelder, G. (2007). Solving nonlinear multiobjective bilevel optimization problems with coupled upper level constraints. Tech. Rep. Preprint No. 320, Preprint-Series of the Institute of Applied Mathematics, University of Erlangen-Nürnberg, Germany.

Eichfelder, G. (2008). Multiobjective bilevel optimization. *Mathematical Programming*. DOI 10.1007/s10107-008-0259-0.

Fampa, M., Barroso, L. A., Candal, D., and Simonetti, L. (2008). Bilevel optimization applied to strategic pricing in competitive electricity markets. *Computational Optimization and Applications*, 39, 121–142.

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

Fonseca, C. M., and Fleming, P. J. (1996). On the performance assessment and comparison of stochastic multiobjective optimizers. In H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel (Eds.), *Parallel Problem Solving from Nature (PPSN IV)* (pp. 584–593). Berlin: Springer. (Also available as Lecture Notes in Computer Science 1141.)

Fudenberg, D., and Tirole, J. (1993). *Game theory*. Cambridge, MA: MIT Press.

Gaur, A., and Arora, S. R. (2008). Multi-level multi-attribute multi-objective integer linear programming problem. *AMO-Advanced Modeling and Optimization*, 10(2), 297–322.

Halter, W., and Mostaghim, S. (2006). Bilevel optimization of multi-component chemical systems using particle swarm optimization. In *Proceedings of World Congress on Computational Intelligence (WCCI-2006)*, pp. 1240–1247.

Hecheng, L., and Wang, Y. (2007). A genetic algorithm for solving a special class of nonlinear bilevel programming problems. In *Proceedings of the 7th International Conference on Computational Science, Part IV: ICCS 2007*, pp. 1159–1162. (Also LNCS 4490).

Herskovits, J., Leontiev, A., Dias, G., and Santos, G. (2000). Contact shape optimization: A bilevel programming approach. *Structural and Multidisciplinary Optimization*, 20, 214–221.

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

Li, H., and Wang, Y. (2007). A hybrid genetic algorithm for solving nonlinear bilevel programming problems based on the simplex method. In *Proceedings of the Third International Conference on Natural Computation (ICNC 2007)*, pp. 91–95.

Li, X., Tian, P., and Min, X. (2006). A hierarchical particle swarm optimization for solving bilevel programming problems. In *Proceedings of Artificial Intelligence and Soft Computing (ICAISC 2006)*, pp. 1169–1178. (Also LNAI 4029.)

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

Miettinen, K. (1999). *Nonlinear multiobjective optimization*. Dordrecht, The Netherlands: Kluwer.

Nolle, L. (2006). On a hill-climbing algorithm with adaptive step size: Towards a control parameter-less black-box optimization algorithm. In B. Reusch (Ed.), *Computational Intelligence, Theory and Applications* (pp. 587–596). Berlin: Springer-Verlag.

Oduguwa, V., and Roy, R. (2002). Bi-level optimisation using genetic algorithm. In *Proceedings of the 2002 IEEE International Conference on Artificial Intelligence Systems (ICAIS-02)*, pp. 322–327.

Pakala, R. R. (1993). *A study on applications of Stackelberg game strategies in concurrent design models*. Master's thesis, Department of Mechanical Engineering, University of Houston.

Paruchuri, P., Pearce, J. P., Marecki, J., Tambe, M., Ordonez, F., and Kraus, S. (2008). Efficient algorithms to solve Bayesian Stackelberg games for security applications. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, pp. 1559–1562.

Rao, S. S. (1984). *Optimization: Theory and applications*. New York: Wiley.

Reklaitis, G. V., Ravindran, A., and Ragsdell, K. M. (1983). *Engineering optimization methods and applications*. New York: Wiley.

Shi, X., and Xia, H. S. (2001). 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.

Sinha, A., and Deb, K. (2009). Towards understanding evolutionary bilevel multi-objective optimization algorithm. Tech. Rep. on Proceedings of the IFAC Workshop on Control Applications of Optimization, Kanpur, Indian Institute of Technology, India. (Also KanGAL Report No. 2008006.)

Sun, D., Benekohal, R. F., and Waller, S. T. (2006). Bi-level programming formulation and heuristic solution approach for dynamic traffic signal optimization. *Computer-Aided Civil and Infrastructure Engineering*, 21(5), 321–333.

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

Wang, G., Wan, Z., Wang, X., and Lv, Y. (2008). Genetic algorithm based on simplex method for solving linear-quadratic bilevel programming problem. *Computers and Mathematics with Applications*, 56(10), 2550–2555.

Wang, G.-M., Wang, X.-J., Wan, Z.-P., and Jia, S.-H. (2007). An adaptive genetic algorithm for solving bilevel linear programming problem. *Applied Mathematics and Mechanics*, 28(12), 1605–1612.

Wang, J. F., and Periaux, J. (2001). 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)*, pp. 552–559.

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.

Wierzbicki, A. P. (1980). The use of reference objectives in multiobjective optimization. In G. Fandel and T. Gal (Eds.), *Multiple criteria decision making theory and applications* (pp. 468–486). Berlin: Springer-Verlag.

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

Zhang, G., Liu, J., and Dillon, T. (2007). Decentralized multi-objective bilevel decision making with fuzzy demands. *Knowledge-Based Systems*, 20, 495–507.