Towards Understanding Evolutionary Bilevel Multi-Objective Optimization Algorithm

Ankur Sinha and Kalyanmoy Deb

Helsinki School of Economics, PO Box 1210, FIN-00101, Helsinki, Finland (e-mail: ankur.sinha@hse.fi, kalyanmoy.deb@hse.fi).

Abstract: A number of studies can be found in the context of bilevel single objective optimization problems, but not many exist, which tackle the bilevel multi-objective problems. Deb and Sinha (October, 2008) proposed a bilevel multi-objective optimization algorithm based on evolutionary multi-objective optimization (EMO) principles and discussed the issues involved in solving such a problem. In this paper we suggest an improved version of the previous algorithm which leads to high number of savings in function evaluations. Simulation results have been presented for two test problems and a comparison with the previous version has been done. The paper also discusses the complexity of bilevel problems and challenges involved in handling such problems. The existence of these problems in many practical problem solving tasks like optimal control, process optimization, game-playing strategy development, transportation problems, and others make it an important area which still needs to be considered by researchers. A two level optimization task involved in solving such problems makes the problem difficult and poses a number of challenges in getting close to the pareto front which has been addressed in the paper.

Keywords: Multi-objective Optimization and Control, Optimal Control, Optimization Methods, Applications in engineering, economics and management.

1. INTRODUCTION

Bilevel problems are a class of problems which require to solve an optimization problem at the lower level to get a feasible solution for another optimization problem at the upper level. In such problems, the lower level optimization problem acts as constraint to the upper level problem. The problems at each level can be a single or a multi-objective problem. Problems of this kind can be abundantly found in practice, particularly in optimal control, process optimization, transportation problems, game playing strategies, reliability based design optimization, and others. Deb and Sinha (October, 2008) suggested an algorithm which uses evolutionary techniques to handle such problems, the algorithm is scalable to any number of objectives and has been shown to solve bilevel single objective as well as bilevel bi-objective problems successfully.

Inspecting a practical bilevel optimization problem we find that the lower level optimization task ensures a certain quality or certain physical properties which make a solution acceptable. Often, such requirements come up as equilibrium conditions, stability conditions, mass/energy balance conditions, which are mandatory for any solution to be feasible. For example. in reliability based design optimization, a feasible design must correspond to a certain specified reliability against failures. Solutions satisfying such conditions or requirements are not intuitive to obtain, rather they often demand an optimization problem to be solved. These essential tasks are posed as lower level optimization tasks in a bilevel optimization framework. The upper level optimization then must search among such reliable, equilibrium or stable solutions to find an optimal solution corresponding to one or more different (higher level) objectives.

Solutions to problems of such type, which are quite important in practice, pose difficulty in searching and also defining the optimal solution (Dempe et al. (2006)). Despite the lack of theoretical results, there exists a plethora of studies related to bilevel single-objective optimization problems (Calamai and Vicente (1994); Colson et al. (2007); Oduguwa and Roy (2002); Yin (2000)) in which both upper and the lower level optimization tasks involve exactly one objective each. Despite having a single objective in the lower level task, usually in such problems the lower level optimization problem has more than one optimum. The goal of a bilevel optimization technique is then to first find the lower level optimal solutions and then search for the optimal solution for the upper level optimization task. In the context of bilevel multi-objective optimization studies, however, there does not exist too many studies using classical methods (Eichfelder (2007)) and none to our knowledge using evolutionary methods, probably because of the added complexities associated with solving each level. In such problems, every lower level optimization problem has a number of trade-off optimal solutions and the task of the upper level optimization algorithm is to focus its search on multiple trade-off solutions which are members of optimal trade-off solutions of lower level optimization problems.

In this paper, we suggest two changes in the bilevel evolutionary multi-objective optimization algorithm by

Deb and Sinha (October, 2008). Firstly, the offsprings being produced in the upper level generations were being evaluated twice in the previous algorithm which led to a significant increase in the function evaluations without much benefit. In the modification the double evaluations have been avoided making the algorithm more economical. Secondly, in the previous version, the parents were chosen for crossover only from the parent population. In the new version the archive members are also allowed to participate in crossover which leads to better results.

2. DESCRIPTION OF BILEVEL MULTI-OBJECTIVE OPTIMIZATION PROBLEM

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

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

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

3. PROPOSED PROCEDURE (BLEMO)

The proposed method uses the elitist non-dominated sorting GA or NSGA-II (Deb et al. (2002)), however any other EMO procedures can also be used instead. The upper level population (of size N_u) uses NSGA-II operations for T_u generations with upper level objectives ($\mathbf{F}$) and constraints ($\mathbf{G}$) in determining non-dominated rank and crowding distance values of each population member. However, the evaluation of a population member calls a lower level NSGA-II simulation with a population size of N_l for T_l generations. The upper level population has a special feature. The population has $n_s = N_u/N_l$ subpopulations of size N_l each. Each subpopulation has the same $\mathbf{x}_u$ variable vector. To start the proposed BLEMO, we create all solutions at random, but maintain the above structure. From thereon, the proposed operations ensure that the above-mentioned structure is maintained from one generation to another. In the following, we describe one iteration of the proposed BLEMO procedure. At the start of the upper level NSGA-II generation t, we have a population P_t of size N_u. Every population member has the following quantities computed from the previous iteration: (i) a non-dominated rank ND_u corresponding to $\mathbf{F}$ and $\mathbf{G}$, (ii) a crowding distance value CD_u corresponding to $\mathbf{F}$, (iii) a non-dominated rank ND_l corresponding to $\mathbf{f}$ and $\mathbf{g}$, and (iv) a crowding distance value CD_l using $\mathbf{f}$. In addition to these, for the members stored in the archive, we have (v) a crowding distance value CD_a corresponding to $\mathbf{F}$and (vi) a non-dominated rank ND_a corresponding to $\mathbf{F}$ and $\mathbf{G}$. For every subpopulation in the upper level population, members having the best non-domination rank (ND_u) are saved as an 'elite set' which will be used in the recombination operator in the lower level optimization task of the same subpopulation.

Step 1: Apply a pair of binary tournament selections on members ($\mathbf{x} = (\mathbf{x}_u, \mathbf{x}_l)$) of P_t using ND_u and CD_u lexicographically. Also, apply a pair of binary tournament selections, using ND_a and CD_a lexicographically, on members randomly chosen from the archive A_t. Randomly choose one of the two winners from A_t and one of the two winners from P_t. The member from A_t participates as one of the parents with a probability of $\frac{|A_t|}{|A_t|+|P_t|}$ otherwise the member from P_t becomes the first parent for crossover. Perform a similar operation on rest of the two parents to decide the second parent for crossover. The upper level variable vectors $\mathbf{x}_u$ of two selected parents are then recombined using the SBX operator (Deb and Agrawal (1995)) to obtain two new vectors of which one is chosen at random. The chosen solution is mutated by the polynomial mutation operator (Deb (2001)) to obtain a child vector (say, $\mathbf{x}_u^{(1)}$). We then create N_l new lower level variable vectors $\mathbf{x}_l^{(i)}$ by applying selection-recombination-mutation operations on entire P_t and A_t. Thereafter, N_l 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 $i = 1, \ldots, N_l$. Thus, for every new upper level variable vector, a subpopulation of N_l lower level variable vectors are created by genetic operations from P_t and A_t. The above procedure is repeated for a total of n_s new upper level variable vectors.

Step 2: For each subpopulation of size N_l, we now perform a NSGA-II procedure using lower level objectives ($\mathbf{f}$) and constraints ($\mathbf{g}$) for T_l generations. It is interesting to note that in each lower level NSGA-II, the upper level variable vector $\mathbf{x}_u$ is not changed. For every mating, one solution is chosen as usual using the binary tournament selection using a lexicographic use of ND_l and CD_l, but the second solution is always chosen randomly from the 'elite set'. The mutation is performed as usual. After the lower level NSGA-II simulation is performed for a subpopulation, the resulting solutions are marked with their non-dominated rank (ND_l) and crowding distance value (CD_l). All N_l members from each subpopulation are then combined together in one population (the child population, Q_t). It is interesting to note that in Q_t, there are at least n_s members having $ND_l = 1$ (at least one coming from each subpopulation). Also, in Q_t, there are exactly n_s different $\mathbf{x}_u$ variable vectors.

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

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

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

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

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

The good solutions of every generation is saved in an archive (A_t). Initially, the archive A_0 is an empty set. Thereafter, at the end of every upper level generation, solutions which have undergone and survived r number of lower level generations and have both $ND_u = 1$ and $ND_l = 1$ from P_t is saved in the archive A_t. The non-dominated solutions (with $\mathbf{F}$ and $\mathbf{G}$) of the archive are kept in A_t and rest members are deleted from the archive. The number r is called as reliability of the solution and is an integer greater than 0. It should be noted that if lower level generations and population size are high then a low value $(1-3)$ of r would suffice but for low values of generations and population size at the lower level, a high value $(4-7)$ is given to r.

In the above BLEMO, we have used a simple termination rule based on specified number of generations for both lower and upper level tasks. The number of function evaluations, until the termination criteria is met depends on N_u, T_u and T_l. The previous version of the algorithm, requires exactly $N_u(2T_u + 1)(T_l + 1)$ number of function evaluations. The modification avoids the dual lower level run for the immediate offspring subpopulation and hence leads to reduced number of evaluations. It should be noted that in the worst case scenario the new version would also need the same number of evaluations but that is unlikely, as it would happen only if throughout the entire upper level generations none of the offsprings enter the parent population. The results shown in the later part of the paper show a two fold benefit of the two modifications, that is, saving huge number of evaluations and attaining a better hypervolume.

4. TEST PROBLEMS AND PARETO-OPTIMAL SOLUTIONS

In the context of bilevel single-objective optimization, there exists some studies (Colson et al. (2007); Calamai and Vicente (1994)) which suggest linear, quadratic and transport related problems. However, to our knowledge, there does not exist any systematic study suggesting test problems for bilevel multi-objective optimization. In this study, we use two test problem, the first one is taken from Eichfelder (2007) and the other test problem has been taken from Deb and Sinha (October, 2008)

4.1 Problem 1

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

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

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

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

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

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

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

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

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

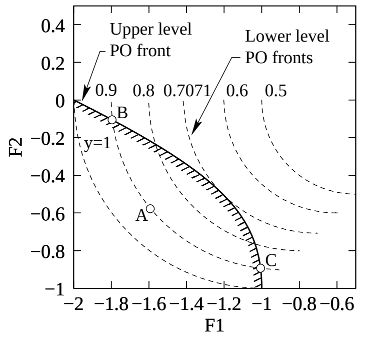

Fig. 1. Pareto-optimal fronts of upper level (complete problem) and some representative lower level optimization tasks are shown for problem 1.

4.2 Problem 2

Next, we consider a bilevel two-objective optimization problem, taken from Deb and Sinha (October, 2008):

$$\text{minimize} \quad \mathbf{F}(\mathbf{x}) = \left\{ \begin{array}{c} (x_1 - 1)^2 + \sum_{i=1}^{K} x_{i+1}^2 + y^2 \\ (x_1 - 1)^2 + \sum_{i=1}^{K} x_{i+1}^2 + (y-1)^2 \end{array} \right\},$$
$$\text{st } (x_1, x_2, \ldots, x_{K+1}) \in \text{argmin}_{(x_1, x_2, \ldots, x_{K+1})}$$
$$\left\{ \mathbf{f}(\mathbf{x}) = \left(\begin{array}{c} x_1^2 + \sum_{i=1}^{K} x_{i+1}^2 \\ (x_1 - y)^2 + \sum_{i=1}^{K} x_{i+1}^2 \end{array} \right) \right\},$$
$$-1 \le x_1, x_2, \ldots, x_{K+1}, y \le 2. \quad (5)$$

For this problem the pareto front corresponds to $x_i = 0$ for $i = 2, \ldots, (K+1)$, $x_1 = y$ and $y \in [0.5, 1]$. In our simulation here, we use $K = 5$, so that total number of variables is 7.

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

When we plot all solutions for which $x_1 = y$ and $x_i = 0$ for $i = 2, \ldots, (K+1)$, we obtain the dotted line marked with '$x_1 = y$' in the figure. Different lower level Pareto-optimal fronts (for different y values) are shown in the figure with dashed straight lines. It is interesting to note that all solutions on this '$x_1 = y$' curve are not Pareto-optimal to the overall problem.

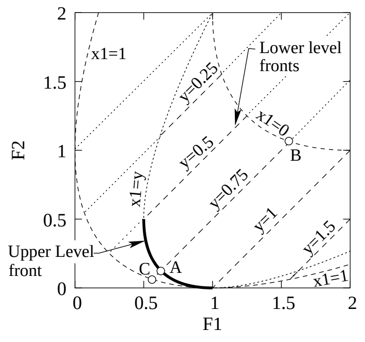

Fig. 2. Pareto-optimal fronts of upper level (complete problem) and some representative lower level optimization tasks are shown for problem 2.

This problem does not have any constraint in its lower or upper level. If an algorithm fails to find true Pareto-optimal solutions of a lower level problem and ends up finding a solution below the '$x_1 = y$' curve, such as solution C, it can potentially dominate a true Pareto-optimal point (such as point A) thereby making the task of finding true Pareto-optimal solutions difficult.

5. PROOF-OF-PRINCIPLE RESULTS

We use the following parameter settings: $N_u = 300$, $T_u = 100$, $N_l = 20$, and $T_l = 30$ for problem 1 and $N_u = 200$, $T_u = 100$, $N_l = 20$, and $T_l = 30$ for problem 2. Since lower level search is made interacting with the upper level search, we have run lower level optimization algorithm for a fewer generations and run the upper level simulations longer. The other NSGA-II parameters are set as follows: for SBX crossover, $p_c = 0.9$, $\eta_c = 15$ Deb and Agrawal (1995) and for polynomial mutation operator, $p_m = 0.1$, and $\eta_m = 20$ Deb (2001).

5.1 Problem 1

Figure 3 shows the obtained solutions using proposed BLEMO. It is clear that the obtained solutions are very close to the theoretical Pareto-optimal solutions of this problem. The lower boundary of the objective space is also shown to indicate that although solutions could have been found lying between the theoretical front and the boundary and dominate the Pareto-optimal points, BLEMO is able to avoid such solutions and find solutions very close to the Pareto-optimal solutions. Also, BLEMO is able to find a good spread of solutions on the entire range of true Pareto-optimal front. Figure 4 shows the variation of $\mathbf{x}$ for these solutions. It is clear that all solutions are close to being on the upper level constraint $G(\mathbf{x})$ boundary $(x_1 + x_2 = -1)$ and they follow the relationship depicted in equation 3.

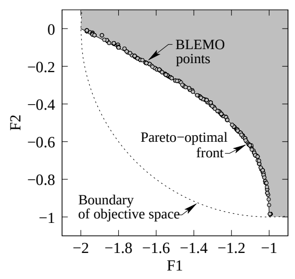

Fig. 3. BLEMO Results for problem 1.

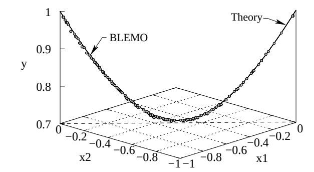

Fig. 4. Variable values of obtained solutions for problem 1. BLEMO solutions are close to theoretical results.

Figure 5 shows all the members in a particular run of the algorithm. The figure contains members which satisfy only the lower level constraints and have undergone at least a single lower level run. Since many of the members in the figure have undergone very few number of lower level runs so they might not be an optimal solution to the lower level problem. All such members become infeasible for the upper level optimization problem. Only the members which have undergone suffcient number of lower level runs (greater than reliability parameter, r) can be considered lower level optimal and upper level feasible solutions for practical purposes.

Figure 6 shows the members which satisfy the lower as well as the upper level constraints and have undergone atleast a single lower level run. It represents all such points produced in a single run of the algorithm for test problem 1. Here we find that a large number of solutions, inspite of satisfying all the constraints, lie below the pareto optimal front. This can again be attributed to the fact that these solutions are not lower level optimum and hence not feasible for the upper level. Such points which satisfy all the constraints but yet lie ahead of the true front make the job of finding the true solutions very diffcult. These members pose a challenge for the algorithm as they dominate the actual solutions, eliminating them and themselves entering the archive. The suggested BLEMO is able to take care of such points and is not misguided by their dominatiion of the good solutions.

Figure 7 and figure 8 show the entire population members of the second and the third generation. All the subpopulation members can be observed to be aligning to its lower level pareto fronts. It can be observed that the subpopulations are spread in the objective search space in the second generaion and have moved towards the global pareto front in the third generaion.

5.2 Problem 2

Figure 9 shows the obtained BLEMO points on problem 3. Although solutions in between this front and the feasible

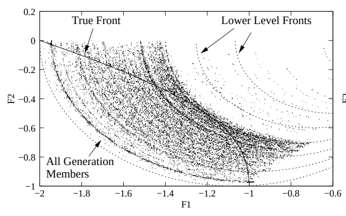

Fig. 5. All population members throughout an entire algorithm run for problem 1.

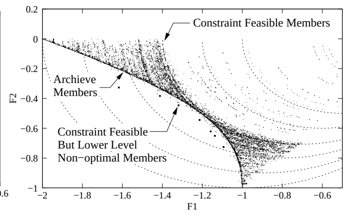

Fig. 6. All constraint feasible population members throughout an entire algorithm run.

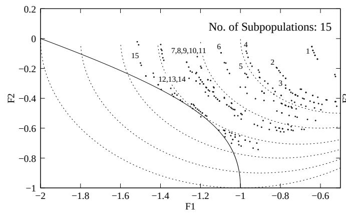

Fig. 7. Generation 2 members for problem 1.

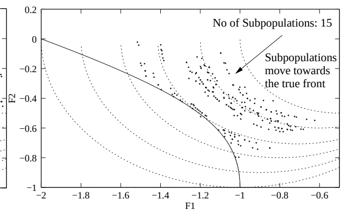

Fig. 8. Generation 3 members for problem 1.

Table 1. Function evaluations for problem 1 by the two versions (20 runs).

Estim.	Old	New
Mean	1869300	1050267
Median	1869300	1047769
Min	1869300	1030316
Max	1869300	1065935
Saving	-	43.82%

Table 2. Function evaluations for problem 2 by the two versions (20 runs).

Estim.	Old	New
Mean	1246200	778503
Median	1246200	771404
Min	1246,200	759221
Max	1246200	805566
Saving	-	37.53%

boundary of objective space could have been found for an apparently better non-dominated front, these solutions would be non-Pareto-optimal with respect to the lower level problems and our algorithm has succeeded in eliminating them to appear on the final front. The figure shows that BLEMO is able to find a good distribution of solutions on the entire range of the true Pareto-optimal front. Figure 10 shows that for obtained optimal solutions, the relationship $y = x_1$ in the range $x_1 \in [0.5, 1]$ holds. Additionally, we observed that $x_i = 0$ for $i = 2, \ldots, (K+1)$ for all obtained solutions. These observations match with the theoretical Pareto-optimal solutions on this problem discussed in the previous section.

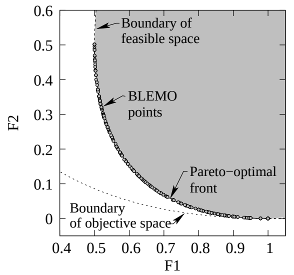

Fig. 9. Results obtained using BLEMO for problem 2.

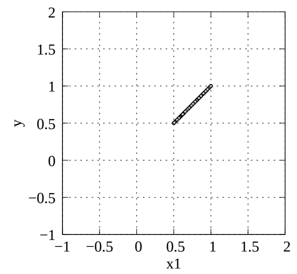

Fig. 10. Variable values of obtained solutions for problem 2.

Figure 11 and figure 12 show the population members in generation two and generation three respectively. It can be observed that the distribution of the population members on the pareto front has improved as we moved from generaion two to three. This suggests that along with lower level optimization the algorithm is able to make right judgements for the choice of variable $\mathbf{y}$ and maintain a good distribution on the front. The problem is difficult, as it requires the end point of the pareto front from the lower level problem and only those points for $y \in [0.5, 1]$ can be the pareto front members of the bilevel problem. Here again we observe that some members from the initial populations have moved ahead of the front and the algorithm has to keep these members away from entering the archive. The algorithm is able to show its efficacy in handling such high variable bilevel problems too.

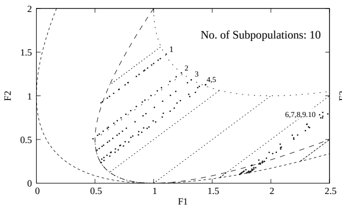

Fig. 11. Generation 2 members for problem 2.

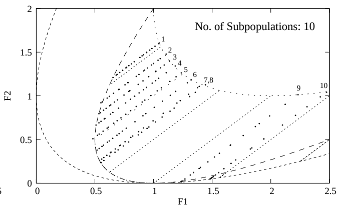

Fig. 12. Generation 3 members for problem 2.

Table 3. Hypervolume values obtained for problem 1 by the two versions (20 runs).

Estim.	Old	New
Mean	0.3007	0.3021
Median	0.3002	0.3023
Min	0.2983	0.2971
Max	0.3018	0.3034

Table 4. Hypervolume values obtained for problem 2 by the two versions (20 runs).

Estim.	Old	New
Mean	0.8274	0.8286
Median	0.8273	0.8289
Min	0.8263	0.8271
Max	0.8292	0.8303

5.3 Discussion of Results

As already mentioned, the previous algorithm, in order to solve the above test problems would take exactly $N_u(2T_u + 1)(T_l + 1)$ number of runs. Table 1 and table 2 give a comparison of function evaluations for the two versions. It can be observed that the modified version leads to a function evaluations saving of 43.82% for the first test problem and 37.53% for the second test problem. Inspite of reduced number of function evaluations in the modified version, it performs better than the previous version which is obvious from the attained hypervolumes in table 3 and table 4. For both the test problems, the mean hypervolume attained by the new version is slightly greater than the hypervolume attained by the previously suggested methodology.

Doing the lower level run for the offspring sub-population, which enters the parent population, once in a generation, gives us an opportunity to eliminate a poor offspring subpopulation soon after one lower level run, thus saving the extra evaluations which would go waste for such a sub-population. So, the high number of evaluation saving can be attributed to this modification. Secondly, the members stored in the archive are some of the best members achieved from previous generations. Making such members participate in the crossover tend to produce offsprings close to the pareto-front, hence enhancing the overall optimization process. This modification is, therefore responsible for better performance and also faster convergence of the algorithm.

6. CONCLUSIONS

In this paper, we have improved upon the existing BLEMO algorithm in terms of function evaluations as well as performance. The utilization of the good members in the archive for crossover steers the algorithm towards the front faster. Ensuring a single lower-level run of the offspring sub-populations in each upper-level generation allows the saving of large number of function evaluations. We have also shown in this study, with the help of two test problems, the complexities involved in solving bilevel-multiobjective problems and suggested techniques to handle them.

ACKNOWLEDGEMENTS

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

REFERENCES

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

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

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

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

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

Deb, K. and Sinha, A. (October, 2008). Solving bilevel multi-objective optimization problems using evolutionary algorithms. Technical Report Kangal Report No. 2008005, Kanpur, India: Department of Mechanical Engineering, Indian Institute of Technology Kanpur. Http://www.iitk.ac.in/kangal/pub.htm.

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

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

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

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