Evolutionary and Deterministic Methods for Design,
Optimization and Control with Applications to Industrial and Societal Problems
EUROGEN 2005
R. Schilling, W.Haase, J. Periaux, H. Baier, G. Bugeda (Eds)
© FLM, Munich, 2005

A MULTI-OBJECTIVE EVOLUTIONARY HYBRID OPTIMIZER

George S. Dulikravich [*], Ramon J. Moral and Debasis Sahoo

[*] Department of Mechanical and Materials Engineering, MAIDROC Lab., 10555 W. Flagler St.
Florida International University, Miami, Florida 33174, U.S.A.
e-mail: dulikrav@fiu.edu   Web page: http://maidroc.fiu.edu

Keywords: Multi-Objective, Hybrid Optimizer, SPEA, MOPSO, NSDE, Automatic Switching.

Abstract. *A new hybrid multi-objective, multivariable optimizer utilizing Strength Pareto Evolutionary Algorithm (SPEA), Non-dominated Sorting Differential Evolution (NSDE), and Multi-Objective Particle Swarm (MOPSO) has been created and tested. The optimizer features automatic switching among these algorithms to expedite the convergence of the optimal Pareto front in the objective function(s) space. The ultimate goal of using such a hybrid optimizer is to lower the total number of objective function evaluations in multi-objective multi-extrema optimisation problems.*

The SPEA is an elitist evolutionary algorithm proposed by Zitzler and Thiele. The algorithm introduces elitism by maintaining an external population for the non-dominated set at each of the iterations. This algorithm uses the elite members in genetic operation to steer the population towards optimal region in multi-dimensional search space. It has the in-built clustering technique, which helps in creating a better spread of the non-dominated solutions. It also supports multi-variables and multi-objective functions. The algorithm has been tested for standard multi-objective test functions and given acceptable results.

The MOPSO algorithm is a modified version on the algorithm proposed by Eberhart and Kennedy. Although this implementation of the original algorithm is primitive, MOPSO supports multi-variables and multi-objective functions. The algorithm has shown the ability to find acceptable optimal Pareto fronts for numerous standard multi-objective test functions.

1 INTRODUCTION

The traditional method for dealing with multi-objective optimization problems is either using a linearly weighted sum of single objectives or a constraint method where the Multi-Objective Optimization Problem (MOOP) is converted into a single objective optimization problem. These methods can unwittingly bias the results to favor a given objective function because of the weights or constraint constants used in the formulation of the MOOP. This issue is described by Deb [2]. If a problem can be formulated in the classical fashion and the formulation is convex and continuous, then the MOOP problem can be optimized using a single objective optimization method. For problems with many extrema, hybrid single objective optimizers can be used to find the global minima as demonstrated by Dulikravich et al. [4]. These optimizers switch between individual optimization routines by tracking changes in the optimum estimate. The optimizer then chooses its next routine by using rules based upon the known strengths and weaknesses of the individual optimization methods that comprise the hybrid optimizer. So, if routine A is known to fail for a given condition, the optimizer then chooses a method that is insensitive to the conditions that caused routine A to fail. The idea being that with enough optimization algorithms, any single algorithm's failure to find the optimum can be circumvented.

We developed a new hybrid optimization algorithm that is capable of dealing with several objective functions simultaneously in a Pareto optimal sense. The hybrid optimizer includes multi-objective versions of three evolutionary optimization algorithms: Non-dominated Sorting Differential Evolution (NSDE) [6], Strength Pareto Evolutionary Algorithm (SPEA) [17], and a Multi-Objective Particle Swarm (MOPSO) algorithm based on Particle Swarm [5]. An automatic switching algorithm was created to switch among these multi-objective optimization algorithms in order to avoid any stalling or meandering of successive Pareto approximations.

2 HYBRID MULTIOBJECTIVE OPTIMIZATION

Like all single-objective hybrid optimizers, multi-objective optimizers are created by combining the strengths of several multi-objective optimization algorithms. The following is the description of the multi-objective versions of the individual optimization routines to be used in this hybrid optimizer. The explanation of the switching methodology will then be discussed. Finally, the results for some standard test problems will be presented showing the Pareto estimate history and comparing the hybrid optimizer to the individual routines that comprise it.

2.1 NSDE – Non-dominated Sorting Differential Evolution (NSDE) Optimization Algorithm

Differential evolution [14] is one of the hybrid evolutionary algorithms, taking the concept of larger population from GA and self adapting mutation from evolutionary strategies. It was initially developed for single objective optimization. Many algorithms were developed for multi-objective optimization by amalgamating DE approach with other evolutionary algorithms having the concept of Pareto [2, 10, 12]. DE had the advantage of incorporating a simple and efficient form of self-adapting mutation.

The concept of DE that is used is the mutation operator. For each new member to be created, three other population members are chosen at random, label as a, b and c. The mutation is done with a probability equal to the crossover rate (CR). The new individual X_c' is created from the member X_c by the difference between members X_a and X_b.

$$X_c' = X_c + F\left(X_a - X_b\right) \tag{1}$$

The reason DE works well is that the mutation is driven by the difference between the parameter values of contemporary population members. This allows each parameter to self tune and give an appropriate reduction in magnitude as the optimization proceeds and convergence is approached.

To extend the idea of DE to multi-objective problems it has been amalgamated with some other multi-objective evolutionary algorithms. This gives us a hybrid algorithm which has the ability to handle the concept of Pareto while it has the advantage of DE's adaptive mutation and simplicity. One of the first examples of this was Pareto differential algorithm which uses non-dominated solution for reproduction and returns offsprings into the population if they dominate their parents [2]. The NSDE algorithm presented here is the combination of NSGA-II with mutation operator inspired by DE [3].

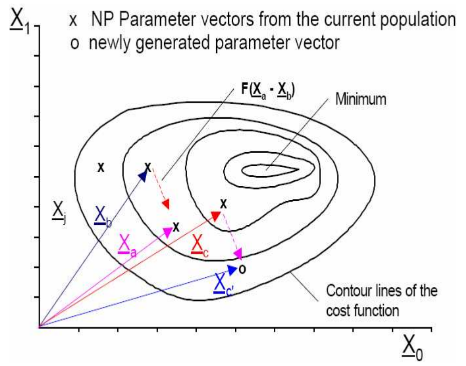
x NP Parameter vectors from the current population
o newly generated parameter vector
$\underline{X}_1$
F($\underline{X}_a$ - $\underline{X}_b$)
Minimum
$\underline{X}_j$
$\underline{X}_b$
$\underline{X}_a$
$\underline{X}_c$
$\underline{X}_{c'}$
Contour lines of the cost function
$\underline{X}_0$

Figure 1. Multi-objective differential evolution optimization algorithm dynamics.

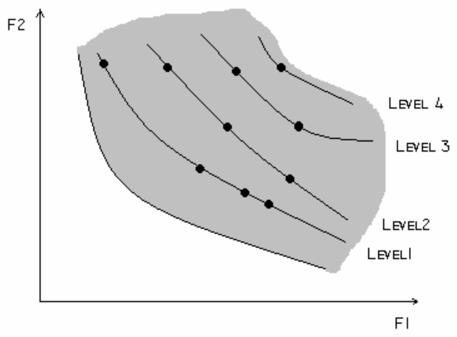

Figure 2. A sketch of sorting of a population according to non-domination levels.

NSGA was developed by Deb et al. [2]. It uses a real-coded crossover and mutation operator, but in the multi-objective implementation of DE, these mutation and recombination operators were replaced with DE. The NSGA-II algorithm uses elitism and a diversity preserving mechanism. In multi-objective DE a new candidate $u_{i,G+1}$ is first added to the new candidate offspring population until the offspring population is filled up to size N. The combined (parents plus offspring) population of size 2N is sorted into separate non-domination levels. Individuals are selected from this combined population to be inserted into the new population, based on their non-domination level. If there are more individuals in the last front than there are slots remaining in the new population of size N, a diversity preserving mechanism is used. The special feature of NSGA is its sorting according to non-domination rank and crowding operator for selecting the new population [3].

2.2 SPEA – Strength Pareto Evolutionary Algorithm

Strength Pareto EA was developed by Zitzler and Thiele [17]. It combines several features of other EAs in a unique manner characterized by:

1. Strongly non-dominated solutions stored externally and continuously updated;
2. Evaluating an individual's fitness depending on the number of external non-dominated points that dominate it;
3. Preserving population diversity using the Pareto dominance relationship;
4. All the solution in the external non-dominated set participate in selection;
5. Incorporating a clustering technique in order to reduce the non-dominated set without destroying its characteristics.

SPEA can be very effective in sampling from the entire Pareto front and distributing the generated solutions over the tradeoff multi-dimensional surface. SPEA is based on the principle of coevolving and the niching technique founded on the concept of Pareto dominance. It is quite capable in guiding the search towards the Pareto-optimal front.

The algorithm of SPEA is as follows:

1. Initial population P and an empty non-dominated population P' are generated
2. The non-dominated members of P are copied into the P' set.
3. Members of P' which are dominated by other members of P' are removed
4. If the number of members of P' exceeds the maximum number N' (specified) then prune it by the method of clustering.
5. Then fitness is calculated for each individual in P as well as in P'.
6. Binary tournament selection is used to select the new set of offspring from the mating pool of P+P'.
7. Two-point crossover and bit mutation are performed, but it can be different depending on the problem at hand.
8. Termination criteria can be the maximum number of generations. Else, go to step 2.

The special feature of this algorithm is its fitness assignment and the clustering procedure.

2.2.1 Fitness Assignment

The fitness assignment process is a two-stage process, separately for P' and P. Each solution $i \in P'$ is assigned a real value $s_i \in [0,1)$, called strength; s_i is proportional to the number of population members $j \in P$ for which $i \succ j$. If n denotes the number of individuals in P that is covered by i and assume N is the size of P then s_i is defined as

$$s_i = \frac{n}{N+1} \tag{2}$$

The fitness of i is equal to s_i. The fitness of an individual $j \in P$ is calculated by summing the strength of all external non-dominated solution $i \in P'$ that covers j. It is added with 1 so as to guarantee that a member of P' has always better fitness than that of P. (Here we will try to minimize the fitness).

$$\text{So, } f_j = 1 + \sum_{i,i \succ j} s_i \text{ where } f_i \in [1, N) \tag{3}$$

This mechanism intuitively reflects the idea of preferring individuals near the Pareto optimal front and distributing them at the same time along the trade-off surface.

2.2.2 Reducing Pareto Size by Clustering

In certain problems the Pareto set can be extremely large. So presenting all the non-dominating points is not quite reasonable and even useless when their number exceeds some bound. Moreover the size of the external non-dominated set N′ influences the behavior of SPEA. On the other hand as the external population participates in selection, too many non-dominated points might reduce selection pressure and slow down the search. And also the niching mechanism relies on a uniform granularity of the grid defined by the non-dominated set. If the points in P′ are not distributed uniformly, the fitness assignment method is possibly biased towards certain regions of search space, leading to unbalanced distribution in the population. So pruning the external population, while maintaining its characteristics, is necessary.

2.3 MOPSO – Multi-Objective Particle Swarm Optimization Algorithm

Particle swarm optimization (PS) is a stochastic optimization method based on the behavior of flocking birds or schooling fish. This method was created by an electrical engineer (Russel Eberhart) and a social psychologist (James Kennedy) [5; 9] as an alternative to GA. Based on the social behavior of various species, PSO tries to equilibrate the individuality and sociability of the individuals in order to locate the optimum of interest. Each particle has information concerning the “best” point in objective space it has come across and the “best” point in objective space that any member of the swarm has come across. The first type of information is individual; the second is global. The individual and global information is transferred to each particle when their velocity is calculated, at each iteration. The velocity equation used is:

$$v_i^{t+1} = \underbrace{w}_{\substack{\text{Inertia}\\\text{Factor}}} v_i^t + \underbrace{c_1}_{\substack{\text{Personal}\\\text{Trust}}} \underbrace{r_1}_{\substack{\text{Random}\\\text{Number}}} \underbrace{\left(P_i - x_i^t\right)}_{\text{Personal Best}} + \underbrace{c_2}_{\substack{\text{Global}\\\text{Trust}}} \underbrace{r_2}_{\substack{\text{Random}\\\text{Number}}} \underbrace{\left(G_i - x_i^t\right)}_{\text{Global Best}} \tag{4}$$

The coefficient w provides inertia from the last velocity calculation. The coefficients c_1 and c_2 are the learning factors and they bias the velocity to favor the particles’ individual knowledge or the swarm’s knowledge. Typically, c_1 and c_2 are given the same value; this is not a requirement [15]. The coefficients r_1 and r_2 are two random numbers varying between 0 and 1. They provide randomness to the particles’ motion. P_i and G_i are the decision vectors for the personal best and global best objective values, respectively. These values are determined for a single objective optimization paradigm. So, P_i and G_i would be a particle’s personal minima and the swarm’s global minima, respectively. This equation provides the data from past particle generations that is passed to future particle generations. Once each velocity has been calculated, the position of each particle is calculated using

$$x_i^{k+1} = x_i^k + t \cdot v_i^{k+1} \tag{5}$$

where, x_i^t is the current position of the particle and t is the time step. The variable t can be taken at any value. The value 1 is most common.

When applying PSO to multi-objective problems, several changes have to be made so that the PSO algorithm handles the Pareto approximation and non-dominated points. In order to do this we must look at the objectives' space, a Pareto approximation and how a particle's value can be ascertained in a multi-objective paradigm.

In Figure 3, the point px refers to the position of particle "x" in objective space. The point Px refers to the personal best for particle "x". The difference between the single objective personal best point and multi-objective personal best point is that the multi-objective personal best point is the non-dominated point in a particle's iteration history.

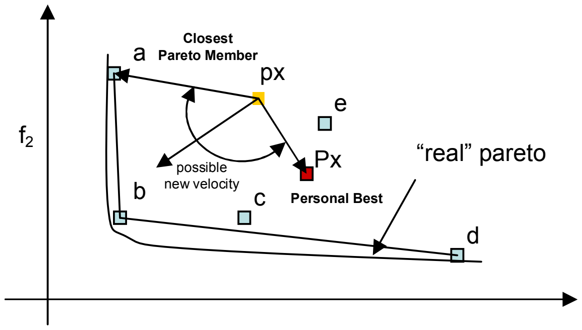

Figure 3. Two-objectives Pareto version of a particle swarm algorithm

Since for a Pareto optimal multi-objective problem there is no single optimum point, the global best point for a particle, is the point on the current Pareto approximation that is closest to the particle. In the figure above it turns out that point "a" is the closest point to point "x" on the Pareto approximation. Now that the P_i and G_i for this iteration have been identified the same velocity and position update equations for the single objective case can be applied.

So, the only difference between the single objective PSO and the Multi-Objective PSO (MOPSO) developed here is the way that the global and personal best points are chosen. Other researchers have used different methods to choose these points [1]. Some researchers have chosen to make weighted sums to adapt PSO to multi-objective problems [11].

2.5 Automatic Switching

As alluded to earlier, for single objective hybrid optimization software the performance of the individual routine can be characterized by its ability to improve the approximation of the global optimum. In multi-objective optimization it is difficult to look at a single value and compare the quality of one Pareto approximation to another [16]. Instead, it is more appropriate to look at multiple attributes of two successive Pareto approximations to determine if a routine is progressing in finding the non-dominated set. The switching criteria developed here looks at 5 different aspects of successive pareto approximations and population generations to see if a routine is expeditiously working to find an accurate approximation of the non-dominates set of an objective space. The five criteria (aspects) are:

1. The new population changes the number of points in the Pareto approximation. When this happens either points are being added to the approximation, or, more importantly, a new point is found that causes points to be dropped from the Pareto approximation.
2. The new population has at least one point that dominates a point, or points, in the current Pareto approximation. This means that the Pareto approximation is being improved.
3. The hyper volume of the dominated space changes. When the optimization software starts it picks a worst case objective vector from its initial population guess. At each optimization routine iteration hyper cubes are formed, with one vertex of the diagonal being a point on Pareto approximation and the other vertex of the diagonal being the worst case objective vector. Then the volume of the union of all the hyper cubes is calculated. The union is defined as the Boolean union of the cubes; in the same sense as this operation is performed in Constructive Solid Geometry for CAD applications [8]. When this occurs, the Pareto approximation is changing geometry.
4. The new population generation causes the average distance of Pareto approximation from the objective space origin to change. This also denotes a change on the Pareto approximation geometry. This is a backup to criteria 3) where two approximations may have the same volume but different average distances.
5. The new population causes the maximum spread of the Pareto approximation to increase. This is calculated using the expression developed by Zitzler as shown by Deb [2]. The formula is:

$$D = \sqrt{\sum_{m=1}^{M} \left(\max_{i=1}^{|Q|} f_m^i - \min_{i=1}^{|Q|} f_m^i \right)^2} \tag{6}$$

At the end of each iteration, the population of design vectors assigns itself a grade point for each of the above criterion that its new generation meets. If the new population earns a grade of 2 or more the current optimization routine is allowed to continue running. When the grade falls below 2, the software switches to the next routine in its repository. If the grad is zero the reason to switch is because the method is not contributing to improving the Pareto approximation. When the score is 1 then the current method may not be contributing to improving the Pareto approximation. As an example the population gives itself a grade of 1 because it meets criterion 4. This change could be caused just by clustering of the previous Pareto approximation and the new population. While this type of change in the Pareto approximation certainly has its uses, it has been found that the multi-objective routines used here can cause these kind of changes to the Pareto approximation ad infinitum, when the Pareto approximation is very near the actual non-dominated set of the objective space. It has been found that by enforcing at least 2 of the criteria, these situations are avoided.

The other limiting factor on how many consecutive iterations a given optimization routine can run is the sub-iteration limit. Although a routine may be able to score a grade of at least two, indefinitely, for each new generation, there may be an optimization method available that can do a better job. For this reason each routine is limited to a user defined maximum sub-iteration limit. This limit gives all the methods a chance to run.

2.6 Hybrid Optimization Software

A multi-objective hybrid optimization software was created using the optimization methods and switching criteria mentioned earlier. The code is object oriented and written in C++. The software has been compiled and runs on MS Windows XP workstation (using Borland C++ Builder X) and Linux Workstation (GNU g++). The software begins execution by creating the population that will be used for the optimization run. The population contains the decision vector and the objective vector for all population points. The population also stores the Pareto approximation and clustering routine. Clustering is performed by the population on the object vectors of the Pareto approximation using the algorithm described by Deb [2]. Once the population has been created, the decision vectors of all population points are evenly distributed over the decision space using Sobol's pseudo random sequence [13]. The software then passes the population from optimization routine to optimization routine as the switching criteria dictates. Each routine gets a chance to "work" on the population until it can no longer score a grade of two or better, or the maximum limit of routine sub-iterations is reached. The termination criterion is the user specified maximum number of objective function evaluations. Figure 4 is a flow chart of the main control loop of the hybrid optimization software illustrating the switching algorithm.

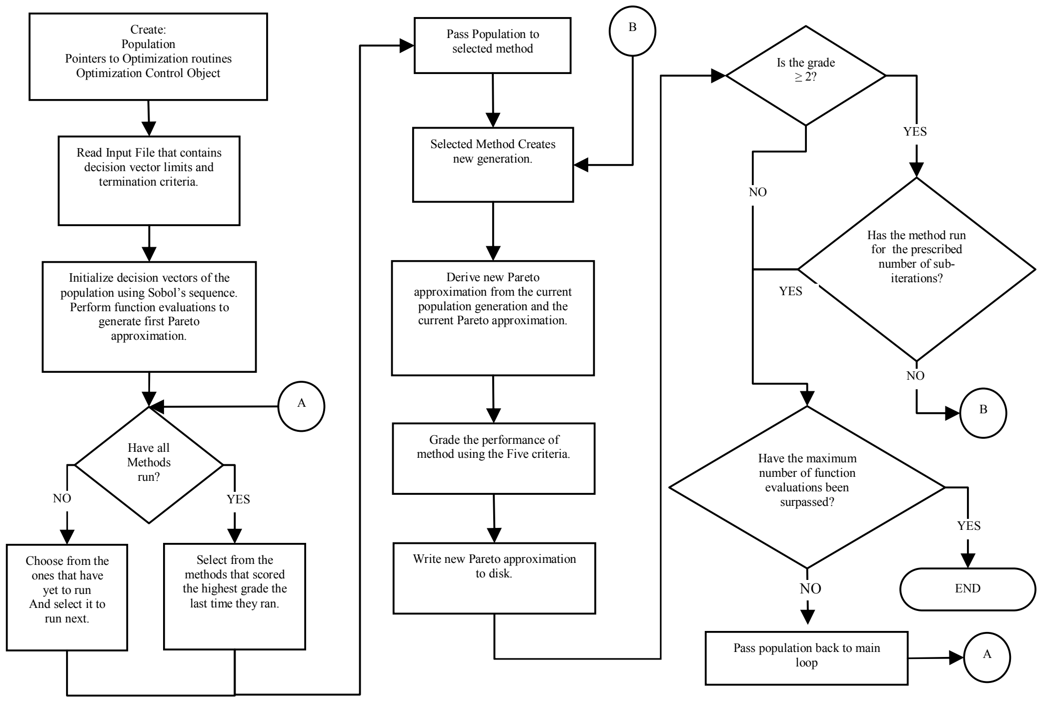
Create:
Population
Pointers to Optimization routines
Optimization Control Object
Read Input File that contains decision vector limits and termination criteria.
Initialize decision vectors of the population using Sobol's sequence. Perform function evaluations to generate first Pareto approximation.
A
Have all Methods run?
NO
YES
Choose from the ones that have yet to run And select it to run next.
Select from the methods that scored the highest grade the last time they ran.
Pass Population to selected method
B
Selected Method Creates new generation.
Derive new Pareto approximation from the current population generation and the current Pareto approximation.
Grade the performance of method using the Five criteria.
Write new Pareto approximation to disk.
Is the grade ≥ 2?
YES
NO
Has the method run for the prescribed number of sub-iterations?
YES
NO
B
Have the maximum number of function evaluations been surpassed?
YES
END
NO
Pass population back to main loop
A

Figure 4-Multi-Objectoive Hybrid Optimizer Flow chart

3 Results

The optimizer was tested on the following standard test functions:

1. Binh Test Function [7]

$$\text{Binh}\begin{cases}\text{Minimize} \quad f_1(\vec{x})=x_1^2+x_2^2 \\ \text{Minimize} \quad f_2(\vec{x})=\left(x_1-5\right)^2+\left(x_2-5\right)^2 \\ -5 \le x_i \le 10,\ i=1,2\end{cases}$$

2. Coello Test Function[7]

$$\text{Coello}\begin{cases}\text{Minimize} \quad f_1(\vec{x})=x_1 \\ \text{Minimize} \quad f_2(\vec{x})=\left(1+10\cdot x_2\right)\cdot\left(1-\left(\dfrac{x_1}{1+10\cdot x_2}\right)^a-\left(\dfrac{x_1}{1+10\cdot x_2}\right)\cdot\mathbf{sin}\left(2\cdot\pi\cdot q\cdot x_1\right)\right) \\ \text{where } q=4,\ a=2 \\ 0 \le x_i \le 1,\ i=1,2\end{cases}$$

3. Poloni Test Function [7]

$$\text{Poloni}\begin{cases}\text{Maximize} \quad f_1(\vec{x})=-\left(1+(A_1-B_1)^2+(A_2-B_2)^2\right) \\ \text{Maximize} \quad f_2(\vec{x})=-\left((x_1-3)^2+(x_2-1)^2\right) \\ A_1=0.5\cdot\mathbf{sin}\,1-2\cdot\mathbf{cos}\,1+\mathbf{sin}\,2-1.5\cdot\mathbf{cos}\,2 \\ A_2=1.5\cdot\mathbf{sin}\,1-\mathbf{cos}\,1+2\cdot\mathbf{sin}\,2-0.5\cdot\mathbf{cos}\,2 \\ B_1=0.5\cdot\mathbf{sin}\,x_1-2\cdot\mathbf{cos}\,x_1+\mathbf{sin}\,x_2-1.5\cdot\mathbf{cos}\,x_2 \\ B_2=1.5\cdot\mathbf{sin}\,x_1-\mathbf{cos}\,x_1+2\cdot\mathbf{sin}\,x_2-0.5\cdot\mathbf{cos}\,x_2 \\ -3.1416 \le x_i \le 3.1416,\ i=1,2\end{cases}$$

4. Kursawe Test Function [7]

$$\text{Kursawe}\begin{cases}\text{Minimize} \quad f_1(\vec{x})=\sum_{i=1}^{m-1}\left(-10\cdot e^{(-0.2)\cdot\sqrt{x_i^2+x_{i+1}^2}}\right) \\ \text{Minimize} \quad f_2(\vec{x})=\sum_{i=1}^{m-1}\left(\left|x_i\right|^a+5\cdot\mathbf{sin}^b(x_i)\right) \\ \text{where } m=3,\ a=0.8, b=3 \\ -5 \le x_i \le 5,\ i=1,m\end{cases}$$

5. Miele-Cantrell test problem combined with Bohachevsky problem #2 [7]

$$\text{Miele}\begin{cases}\text{Minimize} \quad f_1(\vec{x})=\left(e^{x_i}-x_2\right)^4+100\cdot\left(x_2-x_3\right)^6+\text{atan}^4\left(x_3-x_4\right)+x_1^2 \\ \text{Minimize} \quad f_2(\vec{x})=w_1^2+2\cdot w_2^2-0.3\cdot\mathbf{cos}\left(3\cdot\pi\cdot w_1\right)\cdot\mathbf{cos}\left(4\cdot\pi\cdot w_2\right)+0.3 \\ \text{where } w_j=x_j-2,\ i=1,2 \\ -5 \le x_i \le 5,\ i=1,4\end{cases}$$

Figure 5 shows the hybrid optimizer quickly finding a good approximation for the non-dominated set of the Binh problem. The issues with this problem stem in finding the extents of the non-dominated set. The quantification of the maximum spread of Pareto approximation, as implemented in switching criterion number five, contributes to the optimizer favoring an optimization method that can find these extents. Once the extents are established, it is up to the clustering algorithm not to remove the point from the Pareto approximation.

The hybrid optimizer efficiently handled the Coello test problem (Figure 6). This test problem has a discontinuous non-dominated set divided into four distinct groups. If the optimization routine misses a group during its initial guess or never ventures outside the limits of its Pareto approximation, it can misinterpret the topology of the non-dominated set and return a Pareto approximation that contains fewer than four groups. A search algorithm that concentrates on clustering and improving the point distribution over the known Pareto approximation too early in the run could fail to return a good approximation of the non-dominated set of this function if it has not found all four groups of the non-dominated set.

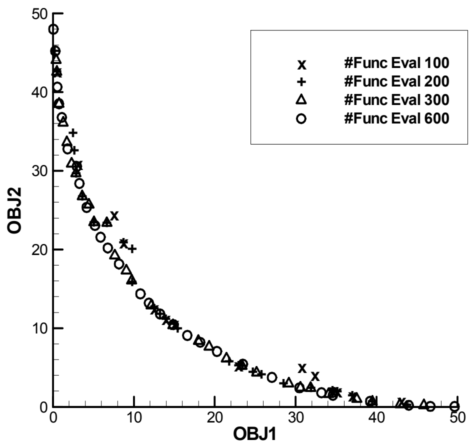

Figure 5. Multi-objective hybrid optimizer convergence for the Binh test problem.

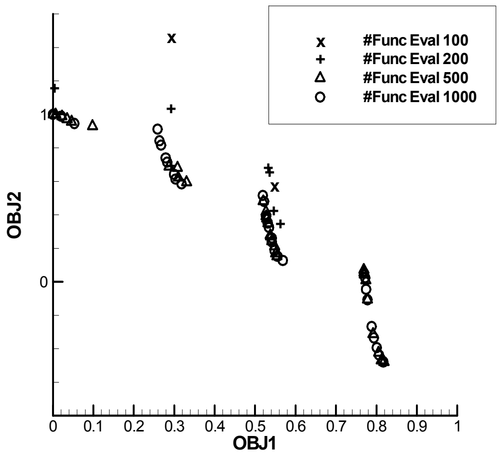

Figure 6. Multi-objective hybrid optimizer convergence for the Coello test problem.

The hybrid optimizer tested equally well on the Poloni problem (Figure 7). In this problem there is a large discontinuity in the non-dominated set. The graph shows that by 80 function evaluations the hybrid optimizer is able to estimate the gap in the non-dominated set with satisfactory accuracy. By 500 function evaluations the spread of the Pareto approximation is agreeing with the extents of the accepted non-dominated set for this problem.

The next problem used to test the hybrid optimizer is the Kursawe problem (Figure 8). The hybrid optimizer required at least 1500 function evaluations before an acceptable Pareto approximation could be found. This problem is more difficult than the previous problems because the population points are very close to each other in the decision space. For the results shown the magnitude of the vector (in decision space) that connects the population members with largest objective two value and the smallest objective two value is ~2.5. This means that an optimization routine that performs clustering in the decision space would have great difficulty finding a good approximation to the non-dominated set. Since the hybrid optimization software clusters in objective space this issue is avoided entirely. But the proximity of the non-dominated points in decision space still taxes the hybrid optimizer because the optimization routines in the optimizer are evolutionary. This means they can generate points outside of the region where the non-dominated set exists. Figure 9 shows the run history of the Miele test problem. In this problem the distance from one extreme of the Pareto approximation to the other, in decision space, only has a magnitude of about 2.25. Also, the magnitude of the first objective varies 3 orders of magnitude with interesting features forming where the first objective value is in 10^{-1} range.

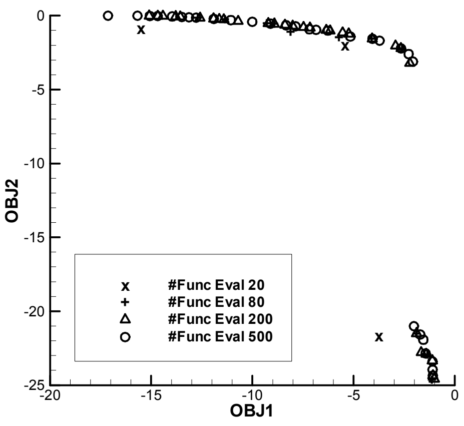

Figure 7. Multi-objective hybrid optimizer convergence for the Poloni test problem.

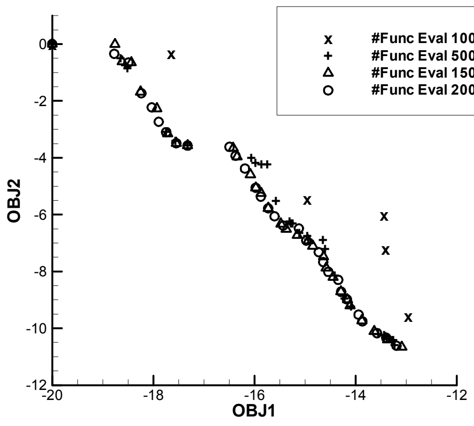

Figure 8. Multi-objective hybrid optimizer convergence for the Kursawe test problem.

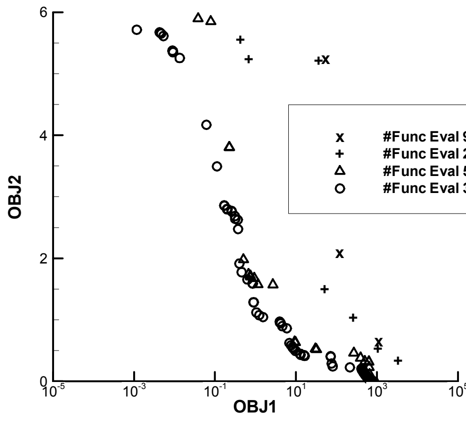

Figure 9. Multi-objective hybrid optimizer convergence for the Miele test problem.

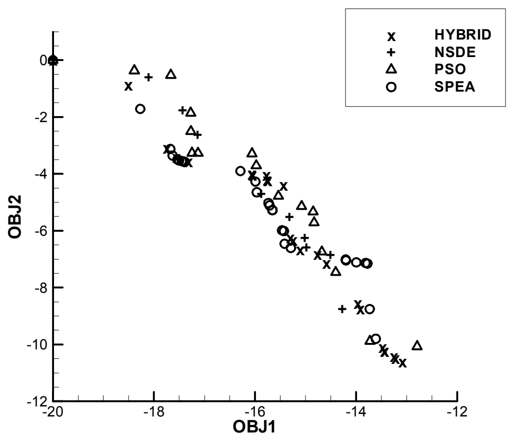

Figure 10. Comparison of different routines with the hybrid optimizer for Kursawe test case.

The hybrid optimizer was modified and three separate optimizers were created. SPEA-only, NSDE-only, and MOPSO-only optimizers were made by disabling the switching algorithm in the hybrid optimizer. The "one-algorithm" versions were then used to solve the test problems. Figure 10 shows a comparison between the hybrid optimizer and the "one-algorithm” optimizers for the Kursawe problem at 400 function evaluations for all optimization routines.

The methods performing the best at this point in the run are the SPEA-only routine and the hybrid optimizer. In the comparison for the Poloni problem after 100 function evaluations (Figure 11), it is found that PSO-only routine and the hybrid optimizer are performing best. Similar results were observed for other test problems. In other words, the fastest algorithms were always the hybrid optimizer and some other method. This shows that the hybrid optimizer reliably switches to the routine that performs most efficiently.

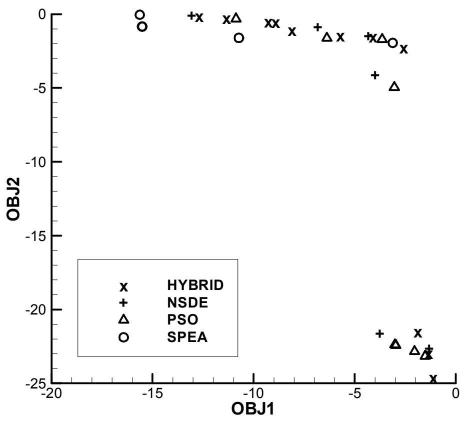

Figure 11. Comparison of different routines with the hybrid optimizer for Poloni test case.

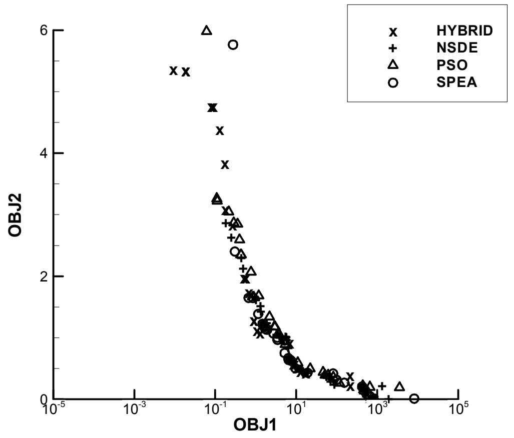

Figure 12. Comparison of different routines with the hybrid optimizer for Miele test case.

Figure 12 shows a comparison of the hybrid optimizer versus the single method optimizer for the Miele test. The results are plotted for all methods at 600 function evaluations. Here the NSDE-only and the hybrid are closest to the non-dominated set but, the hybrid optimizer’s Pareto approximation has larger spread. The MOPSO-only and SPEA-only optimizers have a spread of the Pareto approximation comparable to the hybrid optimizer but they deviate from the non-dominated set at its extremes.

12 CONCLUSIONS

By using multiple performance criterions, it is possible to gage the performance of a multi-objective optimization routine. These criteria can then form the bases for an automatic switching system in a multi-objective optimizer. The switching criteria were successfully used to create a hybrid optimizer. The hybrid optimizer was shown to correctly approximate the non-dominated sets of four different test problems. The hybrid optimizer was shown to perform no worse than the most efficient algorithm considered, on a given test problem. This supports the notion that the hybrid optimizer tends to switch to the most efficient routine available to it given information about the population and latest Pareto approximation

REFERENCES

[1] Alvarez-Benitez, J.E., Everson, R.M. and Fieldsend, J.E., "A MOPSO Algorithm Based Exclusively on Pareto Dominance Concepts", Proceedings of the 3rd International Conference on Evolutionary Multicriterion Optimization, Chihuahua, Mexico, 2005.

[2] Deb, K., *Multi-Objective Optimization Using Evolutionary Algorithms*, John Wiley & Sons. 2002.

[3] Deb, K., Pratap, A., Agarwal. S. and Meyrivan. T., "A Fast and Elitist Multi-objective Genetic Algorithm: NSGA-II", IEEE Trans. on Evolutionary Computation, Vol. 6, No. 2, pp. 182-197, 2002.

[4] Dulikravich, G.S., Martin, T.J., Dennis, B.H. and Foster, N.F., "Multidisciplinary Hybrid Constrained GA Optimization", in EUROGEN'99 - Evolutionary Algorithms in Engineering and Computer Science: Recent Advances and Industrial Applications (eds. K. Miettinen, M. M. Makela, P. Neittaanmaki and J. Periaux), John Wiley & Sons, Jyvaskyla, Finland, pp. 233-259, May 30 - June 3, 1999.

[5] Eberhart, R., Shi, Y. and Kennedy, J., *Swarm Intelligence*, Morgan Kaufmann, 2001.

[6] Iorio, A. and Li, X., "Solving Rotated Multi-objective Optimization Problems Using Differential Evolution", Proceeding of the 17th Joint Australian Conference on Artificial Intelligence, 2004.

[7] IOSO NM Version 1.0, *User Guide*, IOSO Technology Center, Moscow, Russia, 2003.

[8] Mortensen, M., *Geometric Modeling*, John Wiley & Sons, 1985.

[9] Naka, S., Yura, T.G. and Fukuyama, T., "Practical Distribution State Estimation using Hybrid Particle Swarm Optimisation", Proceedings IEEE Power Engineering Society Winter Meeting, Columbus, OH, January 28-February 1, 2001.

[10] Parsopoulos K.E, Tasoulis D.K, Pavlidis N.G., Plagianakos V.P, and Vrahatis M.N, "Vector Evaluated Differential Evolution for Multiobjective Optimization", Congress on Evolutionary Computation, 2004.

[11] Parsopoulos, K. and Vrahatis, M., "Particle Swarm Optimization Method in Multi-objective Problems", Proceedings of the 2002 ACM Symposium on Applied Computing (SAC), pp. 603-607, 2002.

[12] Sarker, R., Abbass, H. and Karim S., "An Evolutionary Algorithm for Constrained Multiobjective Optimization Problems", Proceedings of the 5th Australia-Japan Joint Workshop on Intelligent & Evolutionary Systems, 2001.

[13] Sobol, I.M., "Uniformly Distributed Sequences with an Additional Uniform Property", USSR Computational Mathematics and Mathematical Physics, Vol. 16, pp. 236-242, 1976.

[14] Storn, R. and Price, K.V., "Minimizing the Real Function of the ICEC'96 Contest by Differential Evolution", IEEE Conf. on Evolutionary Computation, pp. 842-844, 1996.

[15] Venter, G., Haftka, R. and Sobieszczanski-Sobieski, J., "Robust Design Using Particle Swarm and Genetic Algorithm Optimization", Proceedings of the 5th World Congress of Structural and Multidisciplinary Optimization, Venice, Italy, May 19-23, 2003.

[16] Zitzler, E.., Thiele, L., Fonseca, C.M. and Grunert da Fonseca, V., "Performance Assessment of Multiobjective Optimizers: An analysis and Review", IEEE Transactions on Evolutionary Computations, Vol. 7, no. 2, IEEE, 2003.

[17] Zitzler, E. and Thiele, L., "Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach", IEEE Transactions on Evolutionary Computation, Vol. 3, No. 4, pp. 257-271, 1999.