AIAA Paper AIAA-2006-6976, **11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference**, Portsmouth, VA, September 6-8, 2006.

Multi-Objective Hybrid Evolutionary Optimization With Automatic Switching

Ramon J. Moral[1], Debasis Sahoo[1] and George S. Dulikravich[2]
Department of Mechanical and Materials Engineering
Multidisciplinary Analysis, Inverse Design, and Robust Optimization Center (MAIDROC)
Florida International University, 10555 W. Flagler St., Miami, Florida 33174, U.S.A.

This work offers details of our recent research dealing with the development of a hybrid multi-objective optimization. Wolpert and Macready determined in single objective optimization that no single algorithm outperforms another over all possible classes of objective functions. This is the well known "No Free Lunch Theorem" (NFL) as it applies to searches and optimization problems. Koppen has shown how NFL can be extended to multi-objective optimization problems. NFL does not reject the possibility that there are classes of problems where one algorithm outperforms another. The problem faced by design engineers is that they may not know the class of problem, from an NFL point of view, they need to optimize beforehand. The objective function(s) form(s) as the design develops, and a robust optimization tool must be able to handle whatever class of objective functions any design renders. If an optimization tool provides multiple search algorithms which are automatically switched to, as the search demands, it may be possible to robustly optimize a large variety of objective function combinations.

In single objective optimization the measure of a search algorithm's progress can be easily calculated from its ability to improve the value of the objective function. The objective function value is the single quality factor that can be used to accurately compare the final results of different optimization routines. A common method to handle Multi-Objective Optimization Problems (MOOP) is to optimize using the concepts of Pareto fronts and non-dominated sets. Zitzler *et al.,* discussed how for MOOPs it is difficult to impossible to use one quality factor to determine which Pareto approximation set is superior among multiple approximations generated by multiple algorithms (when the true non-dominated set is unknown).

In this work the question of robustness in light of NFL, as applied to MOOP, is addressed with the development of a multi-objective hybrid evolutionary computer program. An approach utilizing multiple switching criteria is developed and implemented. The switching criteria are a set of quality factors that the switching algorithm uses to switch between search algorithms automatically. The software developed for this work was tested on multiple standard test function suites. The software was also tested against some of its single algorithm components to determine what performance differences a multi-objective hybrid search software may possess with respect to single search algorithm software.

I. Introduction

Single Objective optimization problems can be divided into two types of searches, single point searches and population based searches. In single point searches the algorithm uses a point in design space and mathematically determined information about the objective function, at that design point, to determine which new search point will yield the best objective function value. The advantage of these types of algorithms is that they tend to quickly, and accurately, converge on the first minima they encounter. The disadvantage of classical single point search algorithms is that the minima they converge to may not, necessarily, be the global minimum of the objective function.

In population based search algorithms, many design points are simultaneously considered by the search software. Evolutionary algorithms, currently very popular in the optimization community, fall into this category.

[1]Graduate student. Student member AIAA.

[2]Professor and Chairman of the Department. Director of MAIDROC. Associate Fellow AIAA.

These kinds of optimization algorithms are very effective at searching for the global minimum over large expanses of design space. They differ from the single point search algorithms in that they do not converge to the first minima they encounter. This versatility is paid for by not being able to quickly converge (when compared to single point searches) to the "absolute" minimum of the global minimum, when it is found. The dichotomy between the performance of single point algorithms and population based algorithms sets the stage for Hybrid Optimizers.

The goal of every Single Objective optimization routine should be to find the global minimum of the objective function using the least number of objective function evaluations possible. As the complexity of an objective function increases so do the computational resources needed to evaluate it. Engineering based objective functions can be based on complex Finite Element models, CFD models, or other physics based modeling techniques. Large scale models can consume hours, or days, of computer time. So, efficient use of the information obtained through each evaluation of the objective function determines if the optimization of a given problem is feasible.

Single Objective Hybrid Optimization algorithms try to combine the advantages of single point search algorithms and population based algorithms. What distinguishes one hybrid algorithm from another are the switching criteria employed. The simplest hybrid optimization algorithm would have two search algorithms, a global and a local. An appropriate set of switching criteria would treat the population based algorithm as the global search method and the single point algorithm as the local search method. The "global" search is conducted until the software is almost certain it has found the region where the global minimum resides. Then the software would switch to the local search to "home in" on the exact location of the global minimum. The criteria for switching between the two searches could be either a) switch if the best member of population cannot be improved after n-iterations or b) switch if the population member stays in the same region of the design space for n-iterations. The example above assumes both algorithms work successfully at their assigned tasks. Colaco et al.[1] uses a philosophy similar to the above utilizing 3 robust search algorithms: Particle Swarm, Differential Evolution, and BFGS. Another highly effective hybrid algorithm, by Dulikravich et al.[2], checks for the failure of its constituent search algorithms, diagnoses the algorithm failure mode, and switches to the correct algorithm to counteract the failures of the first. Regardless of the criteria used, Hybrid Optimization algorithms are designed to keep the search for the global minimum moving robustly so as to minimize the number of evaluations of the objective function. The number and depth of switching criteria vary in complexity according to the hybridization strategy being implemented. Applying the hybrid optimization concept to Multi-Objective Optimization software is complicated by the fact that there is no single quality factor that can be used to properly compare two Pareto approximations[3]. In the Single Objective framework the value of the objective function is the primary quality factor (the only factor if the search is unconstrained) that distinguishes the quality of one global minimum approximation from another.

In this work multiple quality factors are used to allow progress of the optimization software, developed here, to be monitored. These quality factors will then form the basis of Multi-Objective switching criteria. The criteria will be combined with a judging system to determine if the optimization algorithm being used is efficiently finding a good approximation to the non-dominated set. When an algorithm is determined to not be progressing towards the Pareto front in an expeditious manner the software switches out the algorithm for another one. The solution presented here utilizes three evolutionary algorithms: Multi-Objective Particle Swarm (MOPSO), Strength Pareto Evolutionary Algorithm (SPEA), and Non-sorting Differential Evolution (NSDE). Each of these algorithms was chosen because they have different theories of operation and each is effective at handling a good variety of Multi-Objective problems. Thus, they should have some classes of problems, not in common, that each should be able to handle on their own.

The effectiveness of the Multi-Objective Hybrid optimizer will be explored by testing the algorithm on standard test problems. If the hybrid algorithm works, it should be able to handle a wide array of test problems. The second comparison will be the hybrid algorithm versus the individual search algorithms that it is composed of. The idea here is to see if the hybrid algorithm performs as well as, or better, than an individual algorithm. If the switching criteria chooses the best algorithm for each problem the Hybrid algorithm should perform no worse than the best performing single algorithm, for a given set of objective functions. If this property of the hybrid algorithm can be established, then it makes sense to strictly use Multi-Objective Hybrid algorithms for real world designs. This is because, many times, a designer has no idea what class of objective function their design will generate until the design is established. It is then most useful to the industrial design engineer to have the most generalized optimization algorithm possible.

II. Multi-Objective Evolutionary Algorithms

Descriptions of the individual Multi-Objective Evolutionary algorithms used in the creation of the Multi-Objective Hybrid Optimizer follow. Each algorithm's operating operation theory will be reviewed.

II.1 Strength Pareto Evolutionary Algorithm

Strength Pareto EA was developed by Zitzler and Thiele[4]. 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.

II.1.1 Fitness Assignment

The fitness assignment process is a two-stage process, separately for P′ and P. Each solution is assigned a real value, called strength; s_i is proportional to the number of population members. 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

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

The fitness of i is equal to s_i. The fitness of an individual is calculated by summing the strength of all external non-dominated solution 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). So,

$$f_j = 1 + \sum_{i,i>j} s_i \quad \text{where} \quad f_i \in [1, N) \tag{2}$$

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.

II.1.2 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.

II.2 Non-Sorting Differential Evolution

Differential evolution[5] 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[6,7,8]. 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.

$$X_c' = X_c + F\left(X_a - X_b\right) \tag{3}$$

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[6]. The NSDE algorithm presented here is the combination of NSGA-II with mutation operator inspired by DE[5].

NSGA was developed by Deb et al.[6]. 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[9].

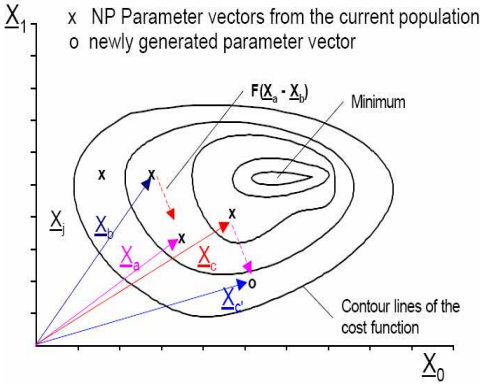

Figure 1. Multi-objective differential evolution optimization algorithm dynamics.

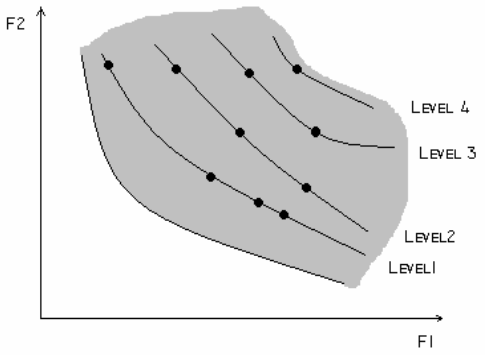

Figure 2. A sketch of sorting of a population according to non-domination levels.

II.3 MOPSO – Multi-Objective Particle Swarm Optimization Algorithm

Particle swarm optimization (PSO) 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)[10,11] 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

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

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[12]. 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.

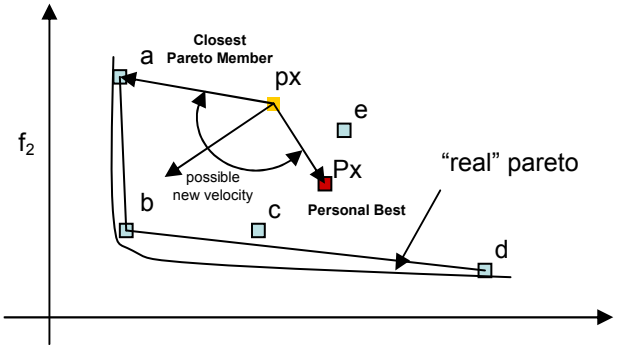

Figure 3. Two-objectives Pareto version of a particle swarm algorithm.

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[13]. Some researchers have chosen to make weighted sums to adapt PSO to multi-objective problems[14].

III. Multi-Objective Hybrid Optimization

Wolpert and Macready's "No Free Lunch Theorem" (NFL)[15,16] tells us that over all possible classes of objective functions, the performance of all search algorithms, on average, will be the same. Koppen[17] shows how NFL can be extended to Multi-Objective Optimization Problems. The purpose of a Multi-Objective Hybrid Optimizer is not to provide a single search algorithm that will have superior performance over all possible optimization problems. That is not possible under NFL. Instead the hybrid optimizer will automate the switching to the search algorithm that is most effective in solving the problem at hand.

In this section the theory of operation of the multi-objective hybrid optimizer[18] is described. First, the development of the switching criteria is described. Next the switching algorithm is defined. Lastly, the block diagram of the Multi-Objective Hybrid Optimizer is presented. For the purposes of this work the definition of hybrid optimization software will be limited to software that switches between multiple optimization algorithms.

The algorithm description assumes that the initial population has been generated by the software, the initial non-dominated population is empty, and switching metrics are at initial value (zero, most of the time). The software presented here uses the Sobol's Pseudo-Random sequence generator as available in the TOMS/ACM[19,20] archive, on www.netlib.org, to evenly distribute the points in the initial population. The algorithm is used to evenly distribute initial populations of design vectors with a maximum dimension of 1111 variables. For design vectors that are larger the 1111 dimension is just repeated over the remaining variables. Clustering is used to keep the population of the non-dominated set limited to user defined size.

IV. Switching Criteria

With a Single Objective Hybrid Optimization routine the switching criteria can be derived by either monitoring algorithm performance with respective to objective function value or algorithm performance with respect to the internal workings of the algorithm itself. The former condition is easy to monitor and easy to react to; the latter requires knowledge of the inner workings of each constituent algorithm. Both methods make effective software. When the objective function value is used for switching, search algorithms are switched out. This occurs when the objective function value is not being minimized at a satisfactory rate or converges to a value. The software then allows another algorithm an opportunity to improve the objective function value.

As mentioned earlier, Zitzler[3] discussed how it may not be possible compare two Pareto approximations from two different optimization algorithms using a single quality indicator. Single quality factors can be combined to form compound quality factors. As compound indicators are developed it then becomes a question of the compatibility and completeness of the derived quality factors. Since it is difficult to impossible to quantify which of two Pareto approximations is superior, for this work, the authors look to sensing only incremental improvements on an algorithm iteration to algorithm iteration basis (or a generation to generation basis). This is done by defining five desired improvements that a new generation makes to the Pareto approximation. The five desired improvements follow.

IV.1 1st Desired Improvement

The first desired improvement is that the size of the population of non-dominated changes at the end of the algorithm iteration. In this hybrid algorithm the non-dominated set of points is maintained as a separate population (P_i, using a notation similar to Zitzler's). To perform this test the new non-dominated set of solutions must be formed. The first step is to merge the new generation with the non-dominated set to form a temporary population.

$$P'_{i+1} = P_i \cup G_{i+1} \tag{6}$$

Next, the new non-dominated set is formed. This can be expressed in pseudo-code as:

$$P_{i+1} = \{P'_{i+1,j} \succeq P'_{i+1,k} \,?\, P'_{i+1,j} : P'_{i+1,k}\} \text{ where } j \neq k \tag{7}$$

Once the new non-dominated set of solutions has been formed the desired improvement can be determined by (again in pseudo-code):

$$\text{Improvement \#1} = \text{Sizeof}(P_{i+1}) \neq \text{Sizeof}(P_i) \,?\, \text{True: False}$$

When this occurs, the Pareto approximation (the non-dominated set) is changing configuration in a positive manner. Either a point is being added to it, which means the resolution of the Pareto approximation is increasing, or, points are being removed from the approximation, which means that shape of the approximation is changing to a more accurate approximation of the actual shape of the true Pareto set for that problem.

IV.2 2nd Desired Improvement

The second desired improvement is that one of the new population points dominates at least one of the points in the last non-dominated set. This can be expressed as:

$$\begin{aligned} &\text{i)} \quad \text{Set A = False} \\ &\text{ii)} \quad \text{Let m = Sizeof}(G_{i+1}) \text{ and n = Sizeof}(P_i) \\ &\text{iii)} \quad (G_{i+1,j} \succeq P_{i,k} \text{ ? True:False}) \vee A;\ j=1,\ldots,m;k=1,\ldots,n \\ &\text{iv)} \quad \text{2nd Improvement = A} \end{aligned} \tag{8}$$

When this occurs, one of the points in the newest generation of the population dominates at least one point in the non-dominated population and will be replacing it. This desired improvement differs from the 1st improvement in that the final size of the non-dominated set need not change.

IV.3 3rd Desired Improvement

The improvement is accomplished if the hyper volume of the dominated space changes from one generation's non-dominated population to the next. This improvement is suggested by Deb[6] in his text on Multi-Objective optimization. In the implementation presented here, a worst case objective vector is created from the initial population. This worst case vector forms one vertex in a hyper cube. The opposing diagonal of the hypercube is formed by the objective vectors of the non-dominated set population making one hyper cube for each member in the non-dominated population. The volumes of the hyper cubes are merged in union operation. This is a Boolean union operation as described by constructive solid geometry[21] principles used to make CAD software. The figure below shows the process applied to a 2 dimensional objective. In this case hypo volumes (areas) are merged. The software can handle any objective space dimension.

Changes in the dominated hyper volume indicate that the algorithm being used is somehow contributing to the non-dominated set.

IV.4 4th Desired Improvement

For this improvement the average distance of the population in the non-dominated set from the objective space origin is calculated. If the average distance of the new non-dominated population from the objective origin is different from the average distance of last generation's non-dominated set then this improvement has been made. As opposed to the last improvement, this improvement tracks the geometric configuration of the dominated set in objective space.

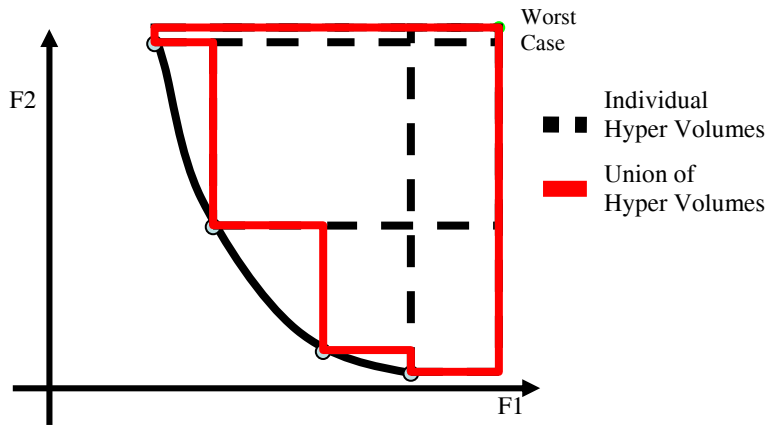

Figure 4. Hyper volume calculation method.

IV.5 5th Desired Improvement

Lastly, the 5th improvement involves checking to see if the latest Pareto approximation (non-dominated set) has increased its spread in objective space. The equation for the spread is attributed to Zitzler and the version used here appears in Deb[6].

$$D = \sqrt{\sum_{m=1}^{M} \left(\max_{i=1}^{|Q|} f_m^i - \min_{i=1}^{|Q|} f_m^i \right)^2} \tag{9}$$

Equation 9 calculates how far into the extremes of each objective function the non-dominated set has progressed.

V. Automatic Switching Algorithm

On first inspection, each of five improvements outlined above seem as if they can measure Pareto approximation progress on their own. This is not the case. Improvements 3 and 4 are linked and no real improvement can be detected without considering both. The reader can imagine a case of two volumes in objective space that dominate the same "amount" of objective space but have different shapes. By considering both improvements 3 and 4, the difference between these two "equal" volumes can be detected because the difference shape is also taken into account. As mentioned earlier, the software uses clustering to limit the size of the non-dominated set to a user specified size. Clustering alone can trigger an improvement without any appreciable improvements to Pareto approximation. These issues all echo from the work of Zitzler et al.[3] that suggests no one quality factor can be reliably used to measure which of two Pareto approximations is better. To avoid these situations, combinations of improvements must be used when determining if a search algorithm is to be allowed to continue to run or if the software needs to switch to another search algorithm.

The Multi-Objective Hybrid Optimization presented here uses a five point grading scheme. When a search algorithm creates a new search population generation, the new non-dominated set (Pareto, approximation is made) and the tests needed to determine if the 5 desired improvements are accomplished. For each improvement the algorithm accomplishes the search algorithm's new generation is assigned one grade point. A grade of two or better and the search algorithm runs again. If this minimum is not met, the program switches to another search algorithm. The hybrid software described here switches between the SPEA, NSDE, and MOPSO search algorithms. The algorithm will also switch out an algorithm if it reaches the, user specified, individual algorithm iteration limit. This iteration limit is in place so that other search algorithms which may work the problem at hand in a more efficient manner, have time to search. A block diagram of the Multi-Objective Hybrid software is below.

VI. Software Performance

The Multi-Objective Hybrid Optimization software was tested with two standard test problems. In the first test the software is allowed to perform the search on the test problem. The resulting Pareto approximation is compared to known solutions for the test problems. The equations for the test problems and the known good Pareto approximations were supplied by IOSO Technologies. Their standard Pareto approximations were made by performing random searches on the test problems for millions of objective function evaluations. Thus, there is no chance that these "good" Pareto approximations are tainted by being generated by another algorithm which converged improperly.

For the second test, the switching algorithm in the hybrid optimizer is disabled. Each individual search algorithm in the program is run for each test problem in the same fashion that the hybrid optimizer was run in the first test. The non-dominated set at 1/5 of the total allowed function evaluations is plotted. The performance of the hybrid optimizer is then compared to the single search algorithm.

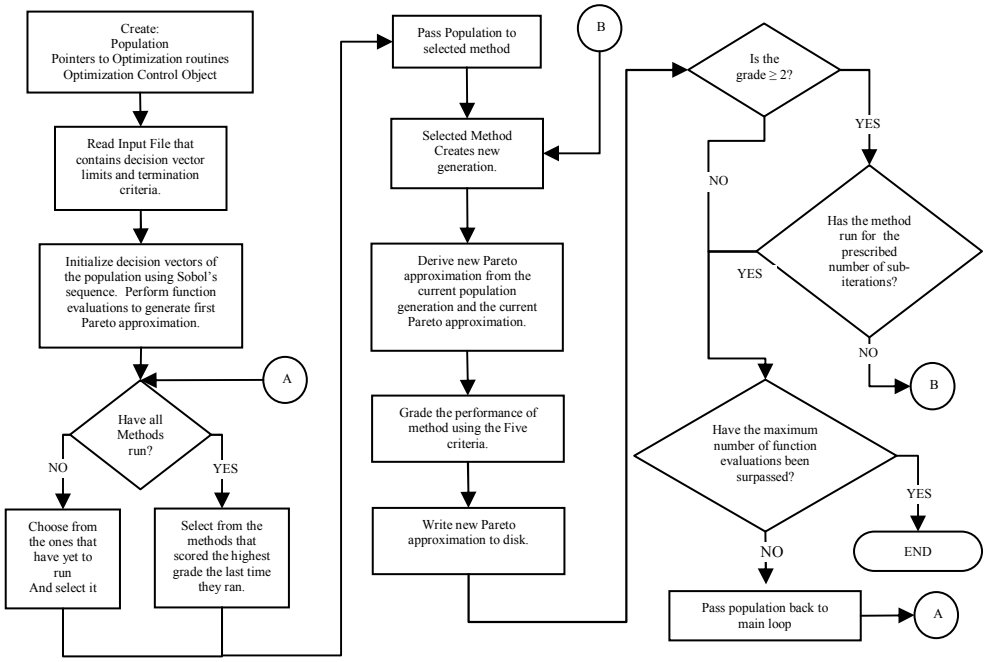

Figure 5. Multi-Objective Hybrid Optimizer Block Diagram.

VI.1 Test Problems

Two test problems are used in testing the hybrid optimizer. These are well known problems and a detailed explanation of these problems can be found in the IOSO user's manual[22].

1. Poloni Test Function[22]

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

2. Miele-Cantrell test problem combined with Bohachevsky problem #2 in the IOSO users' manual[22]

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

VI.2 Hybrid Optimizer Performance

The optimizer was run on the two test problems detailed above. The figures below show the progress of the optimizer with respect to the number of objective function evaluations. For the Poloni Test function the optimizer was limited to 500 objective function evaluations, maximum. In this work an objective function evaluation is the

number of individual function evaluations needed to make a complete objective vector. The population size was set to 20.

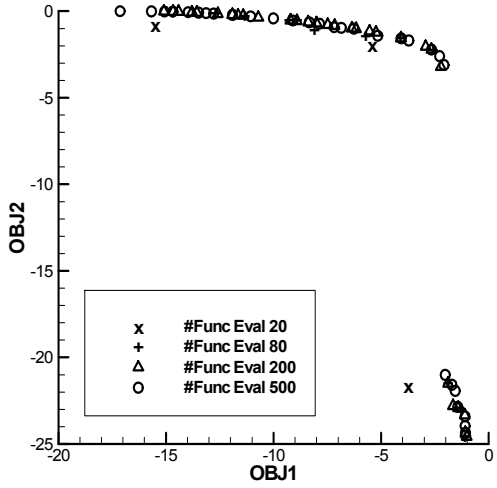

Figure 6. Multi-objective hybrid optimizer convergence for the Poloni test problem.

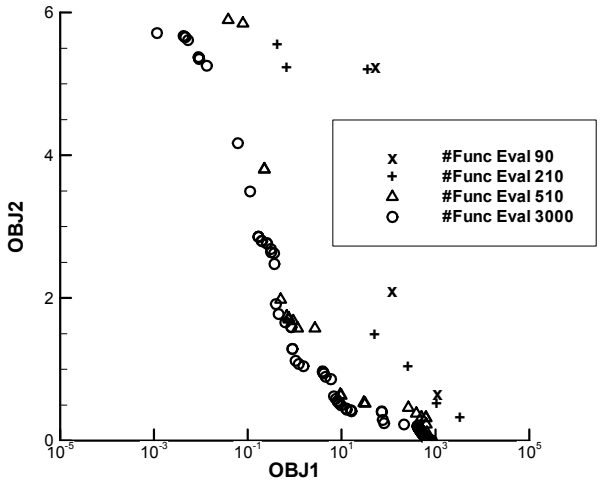

Figure 7. Multi-objective hybrid optimizer convergence for the Miele test problem.

Figure 6 shows the progress of the optimizer after 1, 4, 10, and 25 generations. The results at 25 generations compares remarkably well with the known solution for this problem. The issue facing an optimizer in this problem is its ability to accurately cluster in the objective domain so an accurate representation of the discontinuity in the Pareto front can be defined and maintained.

For the second test problem, two difficult issues are faced by the optimizer. First, the actual non-dominated set has small and numerous discontinuities. Secondly, the magnitude of the first objective functions spans a range from 0.000001 to 1000. This is a nine order of magnitude range, and most of the range is comprised of values less than one. This last issue makes it difficult to cluster points correctly regardless of which domain they are clustered in. Figure 7 shows that the hybrid optimizer was able to capture the discontinuities of the Miele test problem for large values of objective function 1. As the value of objective function one decreases the Pareto approximation thins. This also occurred with IOSO's Pareto front standard which is a random search based on one billion objective function evaluations. The values of Pareto approximation are in agreement with the known good Pareto front.

VI.3 Comparative Performance

In this test the switching algorithm was disabled and the optimizer ran each search algorithm on each test problem. The non-dominated set for all four optimization methods (hybrid, SPEA only, NSDE only, and MOPSO only) at 1/5 of the total function evaluations were then plotted. Figure 8 shows a non-dominated set for the four methods at 100 function evaluations for the Poloni test problem.

The hybrid optimizer has already defined the discontinuity in the Pareto front. The SPEA has best defined the extents of the left hand side of the Pareto front for this problem and does not intrude on the Pareto front's discontinuity. The Pareto approximation with the hybrid optimizer is no worse than with the SPEA approximation alone.

In Figure 8 the Pareto approximation for all four methods are plotted at 500 function evaluations of the Miele test problem. On the left side of the Pareto front the hybrid optimizer performs no worse than the SPEA only and MOPSO only searches. This is the difficult portion of Miele problem because of the small values of objective function number one, as mentioned earlier. On the right side of the Pareto front the hybrid optimizer performs as well as the NSDE only search.

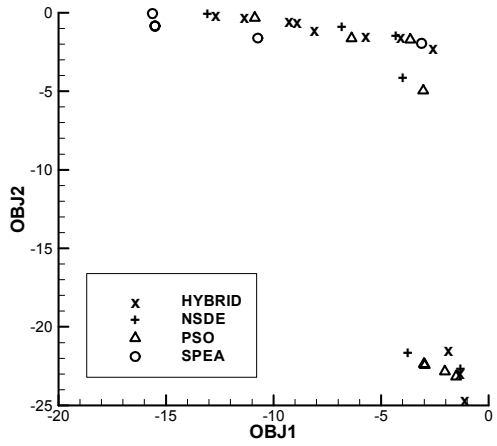

Figure 8- Comparison of different routines with the hybrid optimizer for Poloni test case.

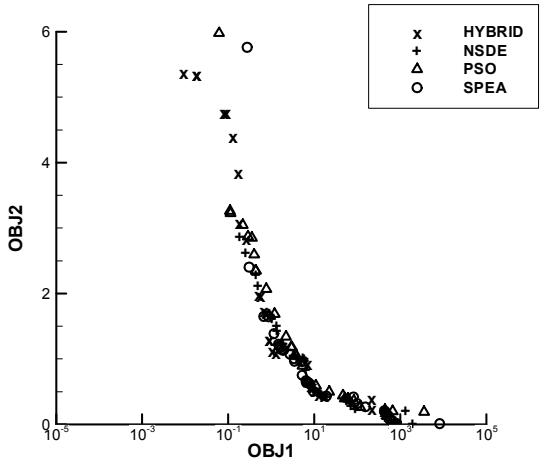

Figure 9- Comparison of different routines with the hybrid optimizer for Miele test case.

VII. Summary

The Multi-Objective Evolutionary Hybrid Optimization software presented here is able to find an accurate Pareto approximation for two standard test problems. The Poloni problem Pareto front approximation was established in 25 population generations. For the Miele problem, 100 generations were needed to establish a good approximation to the Pareto front. The less generations needed, the less computational power needed to solve these problems. This is encouraging at this point in this research.

When compared to the three single search algorithms that comprise it, the hybrid optimizer performed no worse than any of the single algorithms. This result is the most encouraging of all. In light of "no free lunch" theorems, the best that can be hoped for is that the hybrid optimizer switches out poorly performing search algorithms in favor of algorithms that can keep the search going at a brisk pace. Thus, the hybrid algorithm will be no better than the best performing constituent algorithm for a given class of objective functions, but it will be significantly more robust than any individual algorithm.

VIII. Future Work

During our next investigation, the software will be tested against more test problems and some real world design optimization problems. Both the first and second test, as described above, will be done with all problems. The effort to see if the hybrid optimizer can really perform no worse than the best single constituent algorithm for any given class of objective function must continue.

Two sets of improvements will be made to the software. First, another evolutionary search algorithm will be added to hybrid optimizer. Careful consideration will be made to insure the new search algorithm's principal of operation differs as much as possible from the three existing algorithms. This will be done in an attempt to broaden the classes of objective functions that the hybrid algorithm will be able to handle efficiently. The final improvement will be to the clustering algorithm. An attempt to normalize the clustering distance will be made. This will improve the hybrid optimizer by combating the Pareto front thinning problem many optimizers have with problems like the Miele test problem.

Acknowledgements

The authors are grateful for the financial support provided for this work by the US Air Force Office of Scientific Research grant FA9550-06-1-0170 monitored by Dr. Todd E. Combs and by the US Army Research Office/Materials Directorate under the contract number DAAD 19-02-1-0363 monitored by Dr. William M. Mullins.

References

[1]Colaço, J. M., Dulikravich, G. S., Orlande, H. R. B. and Martin, T. J., "Hybrid Optimization With Automatic Switching Among Optimization Algorithms", a chapter in *Evolutionary Algorithms and Intelligent Tools in*

Engineering Optimization (eds: W. Annicchiarico, J. Périaux, M. Cerrolaza and G. Winter), CIMNE, Barcelona, Spain/WITpress, UK, 2005.

[2]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, May 30 - June 3, 1999, pp. 233-259.

[3]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.

[4]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, 1999, pp. 257-271.

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

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

[7]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.

[8]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.

[9]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, 2002, pp. 182-197.

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

[11]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.

[12]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.

[13]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.

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

[15]Wolpert, D. H. and Macready, W., G., "No Free Lunch Theorems for Search," Technical Report SFI-TR-95-02-010, Santa Fe Institute, Sante Fe, NM, USA, 1995.

[16]Wolpert, D. H. and Macready, W., G., "No Free Lunch Theorems for Optimization," *IEEE Transactions on Evolutionary Computation*, 1, 1997, pp. 67-82.

[17]Koppen, M., "On the Benchmarking of Multiobjective Optimization Algorithm, Knowledge-Based Intelligent Information and Engineering Systems, Pt. 1," Proceedings *Lecture Notes in Artificial Intelligence*, 2773, 2003, pp. 379-385.

[18]Dulikravich, G. S., Moral, R. J. and Sahoo, D., "A Multi-Objective Evolutionary Hybrid Optimizer," EUROGEN 05 - Evolutionary and Deterministic Methods for Design, Optimisation and Control with Applications to Industrial and Societal Problems, (eds: R. Schilling, W. Haase, J. Periaux, H. Baier, G. Bugeda), Munich, Germany, September 12-14, 2005.

[19]Algorithm 659 in the Netlib's repository of software from the ACM's Transactions on Mathematical Software

[20]Bratley, P. and Fox, B., "ALGORITHM 659: Implementing Sobol's Quasirandom Sequence Generator", *ACM Transactions on Mathematical Software*, Vol. 14, Issue 1, March 1988.

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

[22]IOSO NM Version 1.0, User's Guide, IOSO Technology Center, Moscow, Russia, 2003.