AIAA JOURNAL
Vol. 46, No. 3, March 2008

Multi-Objective Hybrid Evolutionary Optimization with Automatic Switching Among Constituent Algorithms

Ramon J. Moral* and George S. Dulikravich†
Florida International University, Miami, Florida 33174

DOI: 10.2514/1.28926

In this work, a multi-objective hybrid optimizer is presented. The optimizer uses several multi-objective evolutionary optimization algorithms and orchestrates the application of these algorithms to multi-objective optimization problems, using an automatic internal switching algorithm. The switching algorithm is designed to favor those search algorithms that quickly improve the Pareto approximation and grades improvements using five criteria. A thorough testing of the reliability and accuracy of the multi-objective hybrid optimizer against a number of prominent multi-objective optimization algorithms and one hybrid optimizer confirmed that multi-objective hybrid optimizer performs reliably and accurately.

I. Introduction

HYBRID schemes for single-objective optimization have been used to create robust optimization software by combining known gradient-based and nongradient-based search algorithms [1,2]. What distinguishes single-objective hybrid optimization algorithms from one another is the switching methodologies employed. One method is to use a population-based search until a candidate optimal region in the objective space is determined. Then the search switches automatically to a gradient-based search to quickly close in on the optimum [1]. A second methodology is to monitor the currently used constituent optimization algorithm to see if it stops or fails to progress the search. Then switching is performed to another constituent search algorithm to counter the deficiency that the previous algorithm experienced [2,3]. A third, simpler, method is to switch between algorithms after a predetermined number of function evaluations, thus allowing each constituent algorithm to work on the problem for a guaranteed number of generations/function evaluations.

Hybrid optimization schemes for multi-objective optimization also exist and provide performance gains. Curteanu et al. [4] created a hybridized multi-objective optimizer by making a weighted sum of objective function values from each of the constituent optimization algorithms and applying a genetic algorithm to find a candidate minimum design vector. Then sequential quadratic programming was used to minimize this weighted sum of objective functions. Although these types of methods are useful, they do not optimize multiple objectives in a Pareto-optimal sense.

The multi-objective hybrid optimization (MOHO) [2,3] software described in this work optimizes multiple objectives in a Pareto-optimal sense. Many types of hybrid metaheuristics exist. In his thorough taxonomy of the subject, Talbi [5] pointed out many examples of this type of work. Using the notation developed by Talbi, the work presented here can be classified as an HRH($A_1 + A_2 + \cdots + A_n$) heterogeneous, global, general hybrid algorithm. HRH stands for high-level relay hybrid, meaning that each of the separate and complete n search algorithms run on their own in some sequential execution (nonparallelized) scheme. Vrugt and Robinson [6] presented an HTH(A1 + A2 + A3) (high-level-teamwork hybrid) metaheuristic algorithm called the genetically adaptive multiobjective (AMALGAM) method. In that algorithm, multiple searches run in parallel and contribute a portion of each new generation's population. The portion that each search contributes to the new generation is dependent on the success of the algorithm to provide past useful solutions to the search. Vrugt and Robinson's results show that AMALGAM, which uses nonsorting genetic algorithm (NSGA-II), can outperform NSGA-II on the test problems used in the original NSGA-II paper [7].

The next section of this paper describes the constituent algorithms and operating methodology for MOHO. This is followed by the section that presents the description and results of some numerical tests comparing the capabilities of MOHO with several well-known multi-objective optimization algorithms. Results showing the load distribution for each constituent algorithm are also shown. Finally, discussions of the results and conclusions will be presented, together with suggestions for future work.

II. Multi-Objective Hybrid Optimization Algorithm

The MOHO software is a high-level relay hybrid metaheuristic algorithm. In its current version, three different evolutionary multi-objective search algorithms are coordinated and applied to expedite the search for a Pareto front. Like many other evolutionary algorithms, MOHO runs in steps of population generations. At each generation, the algorithm that is selected makes a new generation using any or all of the information provided to it: the last generation's population and the latest nondominated set. Then the MOHO algorithm combines the new generation and the latest nondominated set to create a new nondominated set. MOHO keeps track of this process to detect five possible improvements to the dominated set (the Pareto approximation). If the particular search algorithm can achieve at least two of any of the five improvements, this algorithm is allowed to create the next generation.

If the search algorithm in question is not able to achieve at least two improvements or it has consecutively run for the user-specified limiting number of iterations, the latest population and nondominated set are passed to the next search algorithm. MOHO runs until the maximum number of function evaluations is performed. All optimization run parameters are specified by the user in an external (to the software) input file.

The population in MOHO (generation 1) is initialized using Sobol's [8] quasi-random sequence generator. The Sobol algorithm used in MOHO is available from netlib.org and originally appeared

Presented as Paper 6976 at the 11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Portsmouth, VA, 6–8 September 2006; received 18 November 2006; revision received 10 September 2007; accepted for publication 15 November 2007.

*Graduate Student, Department of Mechanical Engineering and Materials Engineering, Multidisciplinary Analysis, Inverse Design, Robust Optimization and Control, 10555 West Flagler Street.

†Professor and Chairman, Department of Mechanical Engineering and Materials Engineering, Director of Multidisciplinary Analysis, Inverse Design, Robust Optimization and Control, 10555 West Flagler Street. Associate Fellow AIAA.

in [9]. This algorithm is preferred because it evenly distributes populations with design vectors of over 1000 variables. When the design variable limit of the ACM Sobol [8] algorithm is reached, the sequence is just repeated over the remaining variable dimensions.

MOHO was successfully run on Microsoft Windows workstations, Linux workstations, and Beowulf-style clusters. The software can run in two modes: serial and parallel. In serial mode, the optimization and objective function evaluation occur on the same processor. In parallel mode, the optimizer runs on a processor (the cluster server in this case) and the objective function evaluations are then made into jobs that are submitted to the job manager of the cluster in question. This architecture is preferred by the authors to allow for large parallelized computational-fluid-dynamics- and finite-element-analysis-based objective function calls that consume significant amounts of computing time.

The remainder of this section is organized into two subsections. In the first subsection, the constituent multi-objective search algorithms are presented. In the second subsection, the automatic switching algorithm is discussed in greater detail.

MOHO uses three multi-objective optimization algorithms: strength Pareto evolutionary algorithm (SPEA-2) by Zitzler et al. [10], a multi-objective implementation of the single-objective particle swarm (referred to as MOPSO here) by Eberhardt et al. [11], and a nonsorting differential evolution (NSDE) algorithm, which is a low-level hybrid metaheuristic search combining NSGA-II by Deb et al. [7] and differential evolution by Storn and Price [12]. Each of these constituent search algorithms was modified, if needed, to fit into MOHO's system of maintaining the nondominated set and clustering outside of the search algorithms. Also, each search algorithm was set up to be able to accept populations and nondominated sets in a generalized format to allow the hybridization to run smoothly. These conditions do not adversely affect the application of the individual search algorithms.

SPEA-2 is used in the following manner for this work:

1) A population P and a nondominated set P' are handed to the algorithm from the centralized part of the MOHO software.

2) The fitness is then calculated for members of P and P'.

3) Binary tournament selection is used to select the new set of offspring from the mating pool of $P + P'$.

4) Two-point crossover and bit mutation are performed, but can be changed by the user from the input to better suit a given problem.

5) The new population and the old nondominated population are returned to the centralized portion of MOHO in which the new P' is generated and clustering is performed, if needed.

The SPEA-2 algorithm does not perform the merging of populations and nondominating sets. This step is performed by MOHO, external to the search, so that the switching algorithm can monitor the progress of the nondominated set.

The multi-objective particle swarm algorithm used for this work is derived from the original particle swarm optimization (PSO) algorithm developed by Eberhardt et al.[11]. Some MOPSO algorithms use weighted sums of PSO algorithms [13]. Although this type of application of PSO is effective in its own right, an optimization algorithm solving multi-objective problems in a Pareto-optimal sense was needed. To modify PSO for multi-objective optimization, the definitions of personal best value and global best values were modified. In MOPSO, the personal best value for a particle is the nondominated objective in the particle's search history. The global best value is the member of the current generation's nondominated set that is closest (in objective space) to the particle for which the velocity is being calculated. Other than the method for choosing the global and personal best for each particle, the rest of MOPSO remains true to the original PSO. The MOPSO algorithm is used in the following way for this work:

1) A population P and a nondominated set P' are handed to the algorithm from the centralized part of the MOHO software.

2) A velocity vector for each particle is calculated using the technique described earlier.

3) The displacement for each particle is calculated using the equations of motion and unit time step.

4) The new population and the old nondominated population are returned to the centralized portion of MOHO in which the new P' is generated and clustering is performed, if needed.

The NSDE created for this work is a low-level combination of NSGA-II [7] and differential evolution (DE) [12]. In particular, the mutation operator from DE replaces the mutation operator in the original NSGA-II. The rest of the algorithm is from NSGA-II. At the beginning of the algorithm, the last population and the nondominated population are used to play the roles of the offspring and parents from the last generation, respectively. At this point, NSDE continues on like NSGA-II until the new generation is created. Then the new population and the old nondominated set are handed back to the switching algorithm in MOHO.

III. Automatic Switching Among Constituent Search Algorithms

After each generation is created and the new nondominated set is identified, the software assigns the new nondominated set a score from zero to five. If the new nondominated set gets a score of two or better, the algorithm that generated the set is allowed to create the next generation of population points. An algorithm scores a point for each of five possible improvements that are achieved from one generation to the next. Also, if an algorithm has run consecutively beyond a user-specified iteration (generation) limit, the run switches to the next constituent search algorithm determined by the algorithm pointer array. This rule is used to allow all constituent search algorithms an opportunity to improve the Pareto approximation.

The switching algorithm compares the nondominated set from the current generation to the nondominated set of the previous generation. The comparison process consists of looking at five desired improvements to the Pareto approximation. The improvements are actually gains in five performance criteria (quality factors). More than one quality factor is used, because of the work of Zitzler et al. [14], in which it is shown that as opposed to the situation in a single-objective optimization, one quality factor alone should not be used to compare two Pareto approximations. This is why an algorithm must score a minimum of two to continue creating new generations of population points. This work also puts forth definitions of compatibility and completeness for quality factors. After the improvement metrics for the MOHO algorithm are presented, the applicability of the compatibility and completeness presented in Zitzler et al. will be discussed.

The main part of MOHO handles combining the new generation with the most recent nondominated set to form the new nondominated set by employing objective space-based clustering as needed and calculating the improvements. The clustering is employed only as a nondominated-set trimming tool when the software determines that the nondominated set will grow beyond the user-defined limit. Five improvements were developed and used in this work, because there is a concern that adding too many Pareto-based calculations would add considerable overhead to the software. Now that the algorithm switching logic has been discussed, the five improvements will be defined.

A. Improvement 1: Nondominated-Set Size Changes

For this improvement, the size of the nondominated set changes. This change can be either the nondominated set getting larger or smaller. The nondominated set grows in size when a new point on the Pareto approximation front is discovered and the old nondominated set is below the user-defined nondominated-set size limit. Shrinking of the nondominated set occurs when a new point(s) dominates a larger set of points from the old nondominated set. This indicates one of two things:

1) The new point(s) significantly redefine the geometry of the nondominated set.

2) Clustering has yet to be employed on the Pareto approximation and multiple points were clustered around each other.

B. Improvement 2: A Point from the New Generation Dominates

This improvement is satisfied if any population member of the new generation dominates any member in the last generation's nondominated set. Any opportunity to improve the Pareto approximation by removing a dominated point is considered an improvement. This can be expressed as follows:

1) Set A equal to false.

2) Let m equal the size of the population and n be the size of the nondominated set.

3) $(\text{POP}_j \succ \text{NonDom}_k \,?\, \text{True:False}) \vee A \quad (j = 1, \ldots, m \quad \text{and} \quad k = 1, \ldots, n)$.

4) The second improvement is equal to A.

C. Improvement 3: Change in the Dominated Hypervolume

The hypervolume quality factor has its roots in the hypervolume calculation presented by Deb [15]. When the first population generation is created using Sobol's [8] quasi-random sequence generator, the worst objective value for each objective from the entire population is collected into a worst-case objective vector. This worst-case objective vector is used as the common diagonal for all hypervolume calculations in the search: a static datum for the entire run. The improvement is considered fulfilled when the new generation's contribution to the nondominated set causes a change in the dominated hypervolume for the nondominated set.

C. Improvement 4: Average Distance Change

In this calculation, the average Euclidian distance of all the objective vectors of the new generation's nondominated set is calculated. This is basically the magnitude of the objective vectors, because the datum is the origin of the objective space. If the measure changes from the value of the old generation, this improvement was met. This improvement tries to capture changes in the geometry of the nondominated set.

E. Improvement 5: Spread of the Nondominated Set

The equation for the spread is found in a book by Deb [15] and its origins are attributed to Zitzler et al. [16]. This improvement is fulfilled if the new generation's nondominated set increases the spread over that of the last generation. The equation for the spread is as follows:

$$\text{SPRD} = \sqrt{\sum_{m=1}^{M}\left(\max_{i=1}^{|Q|} \text{f}_m^i - \min_{i=1}^{|Q|} \text{f}_m^i\right)^2} \tag{1}$$

Improvements 3 and 5 are common metrics for trying to determine if one Pareto approximation set is better than another. The other improvements are not. In fact, the other improvements would be useless to compare final Pareto approximations made by two different multi-objective algorithms.

This brings us back to the aforementioned work of Zitzler et al. [14]. In that work, the definitions of compatibility and completeness are made for the scenario of the accuracy and usefulness of indicators that determine which of two Pareto approximations, A or B, is better. In short, compatibility is the capability of a comparison method to accurately yield a true result when one approximation is better than another approximation, with respect to the comparison metric. A metric having only the compatibility property has a price: there can be sets that are better than others and the comparison method will yield a false result. Comparison methods that are complete yield a true result for all approximations that are compared. Obviously, the choice of quality indicators is important to create comparisons that have both the properties of completeness and compatibility.

In Zitzler et al. [14], the two Pareto approximations are made by two different optimization algorithms. Each algorithm comes to its own Pareto approximation independently from the other. In the case of the switching criteria used in this work, this scenario does not apply. At a given generation, the new population is combined with the last generation's nondominated set. A new nondominated set is determined from this union. If the size of the resulting nondominated set surpasses the user-set limit, clustering is applied to the set until the size of the set is acceptable. Using this methodology for progressing the Pareto approximation yields three possible outcomes for the new nondominated set, with implications about the desired improvements.

The first possibility is that the new generation provides no contribution to the nondominated set. Here, the algorithm detects no improvements. The old nondominated set is the new one.

The second possibility is that some of the new generation population becomes a part of the new nondominated set. In this situation, it is certain that the switching algorithm will detect at least one of the improvements. There is no chance that the newest nondominated set and the last one are incomparable, because a part of the old set still exists in the new set.

In the third possibility, members of the new generation (some or all) form a completely new nondominated set and retire the set from the last generation. Here, we are certain that more than one improvement will be detected. Regardless of which of the three situations occurs, detection of an improvement always means that the nondominated set has changed in a favorable manner, because detecting an improvement is not the method by which the Pareto approximation is developed. The Pareto approximation is developed using dominance calculations and objective-based clustering [15]. The improvements are used to benchmark how many of five aspects of the Pareto approximation change from generation to generation.

IV. Numerical Experiments

MOHO was tested to evaluate its capability by running it on test problems from three previously published works. The first work examined is the well-known multi-objective optimization comparison paper by Zitzler et al. [16] (referred to as ZDT from this point forward). A copy of the original data from the ZDT paper can be found on the Web site cited in that paper. In the present work, we will present this data again and inject the results from our MOHO into this original comparison. The second and third works that MOHO will be compared with are related to each other. In the original NSGA-II paper, Deb et al. [7] revisited some of the ZDT problems, presented results for some other unconstrained problems, and used NSGA-II to solve some constrained problems. In the present work, our MOHO will be used to solve the unconstrained problems of the original NSGA-II paper. Results from that paper will be compared with MOHO results. Finally, the paper by Vrugt and Robinson [6] compared their AMALGAM algorithm to the performance of NSGA-II for some unconstrained multi-objective optimization test problems. These results will also be included in this work and compared with results from MOHO using the same experimental conditions.

In the ZDT work, the SPEA, NSGA, vector-evaluated genetic algorithm, niched Pareto genetic algorithm (NPGA), Hajela and Lin's evolutionary algorithm, Fonseca and Fleming's evolutionary algorithm, and a single-objective evolutionary algorithm were studied. The algorithms were applied to six multi-objective problems designed by those authors specifically for that work. The six problems from the ZDT paper [15] are summarized in Eqs. (2–7). All test problems represent two-objective optimizations in which both objectives are to be minimized.

ZDT1:

$$f_1 = x_1 \qquad g = 1 + 9 \cdot \sum_{i=2}^{m} \frac{x_i}{m-1} \qquad h = 1 - \sqrt{\frac{f_1}{g}} \qquad f_2 = h \cdot g \tag{2}$$

where the number of variables is $m = 30$ and $x_i \in [0, 1]$, and the true Pareto front is found by setting $g = 1$.

ZDT2:

$$f_1 = x_1 \qquad g = 1 + 9 \cdot \sum_{i=2}^{m} \frac{x_i}{m-1} \qquad h = 1 - \left(\frac{f_1}{g}\right)^2 \qquad f_2 = h \cdot g \tag{3}$$

where the number of variables is $m = 30$ and $x_i \in [0, 1]$, and the true Pareto front is found by setting $g = 1$.

ZDT3:

$$f_1 = x_1 \qquad g = 1 + 9 \cdot \sum_{i=2}^{m} \frac{x_i}{m-1}$$
$$h = 1 - \sqrt{\frac{f_1}{g}} - \left(\frac{f_1}{g}\right) \cdot \sin(10\pi f_1) \qquad f_2 = h \cdot g \tag{4}$$

where the number of variables is $m = 30$ and $x_i \in [0, 1]$, and the true Pareto front is found by setting $g = 1$.

ZDT4:

$$f_1 = x_1 \qquad g = 1 + 10 \cdot (m-1) + \sum_{i=2}^{m} (x_i^2 - 10\cos(4\pi x_i))$$
$$h = 1 - \sqrt{\frac{f_1}{g}} \qquad f_2 = h \cdot g \tag{5}$$

where the number of variables is $m = 10$, $x_1 \in [0, 1]$, and $x_i \in [-5, 5]$ (where $i \neq 1$), and the true Pareto front is found by setting $g = 1$.

ZDT5:

$$f_1 = 1 + u(x_1) \qquad g = \sum_{i=2}^{m} v(u(x_1)) \qquad h = \frac{1}{f_1}$$
$$v(u(x_1)) = \begin{cases} 2 + u(x_1) & \text{if } u(x_i) < 5 \\ 1 & \text{if } u(x_i) = 5 \end{cases} \qquad f_2 = h \cdot g \tag{6}$$

where the number of variables is $m = 11$, $x_1 \in [0, 1]$ for 30-bit resolution, and $x_i \in [0, 1]$ for 5-bit resolution (where $i \neq 1$), the true Pareto front is found by setting $g = 10$, and $u(x_i)$ is the number of ones in the bit vector representation of x_i.

Table 1 Run parameters for ZDT comparison

Number of generations	250
Population size	100
Crossover rate	0.8
Mutation rate	0.01

ZDT6:

$$f_1 = 1 - \exp(-4x_1)\sin^6(6\pi x_1) \qquad g = 1 + 9 \cdot \left(\frac{\sum_{i=2}^{m} x_i}{m-1}\right)^{0.25}$$
$$h = 1 - \left(\frac{f_1}{g}\right)^2 \qquad f_2 = h \cdot g \tag{7}$$

where the number of variables is $m = 10$ and $x_1 \in [0, 1]$, and the true Pareto front is found by setting $g = 1$.

Table 1 shows the run conditions for MOHO. All seven algorithms were compared in two ways. First, a plot of the nondominated sets for each algorithm was made. To allow for performance fluctuations caused by the random number generators driving the initial population and genetic operators, all algorithms were run 30 times on each of the six problems. The nondominated plot is generated by making a union of the nondominated sets for the first five runs of each algorithm on each problem. The nondominated set of the unions is then plotted. In the figures presented here, the results for MOHO are injected into the plot results from ZDT and the random number data from the original plots are removed to provide a clearer view of the performances of the search algorithms. Figures 1–6 are the plots for ZDT test problems 1 through 6, respectively, using the original ZDT paper data and MOHO data run for this work.

From these figures, it is evident that our MOHO algorithm performs very well on almost all of the ZDT test cases as far as accuracy is concerned. In addition to being an accurate evolutionary search algorithm, these figures demonstrate that MOHO is also a reliable algorithm that consistently gives good results.

The second comparison mechanism is the "cover" function proposed in the ZDT work. The cover function evaluates what fraction of the nondominated set of algorithm A is either equal to or weakly dominates the nondominated set of algorithm B. The formula for the cover functions, as shown in the ZDT work, is

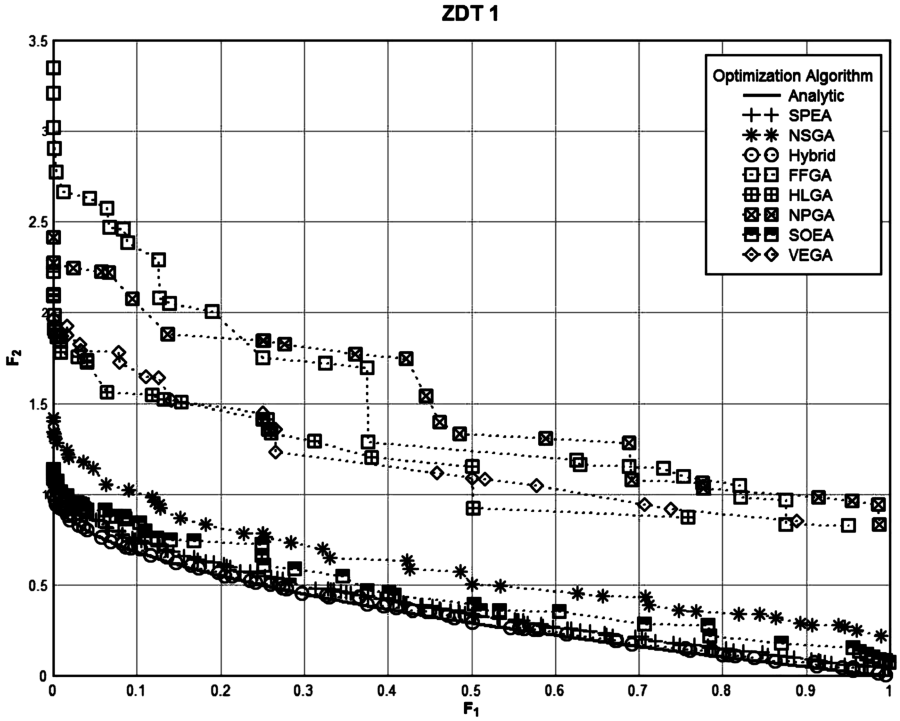

Fig. 1 Results for test problem 1 from ZDT with MOHO results added (circles).

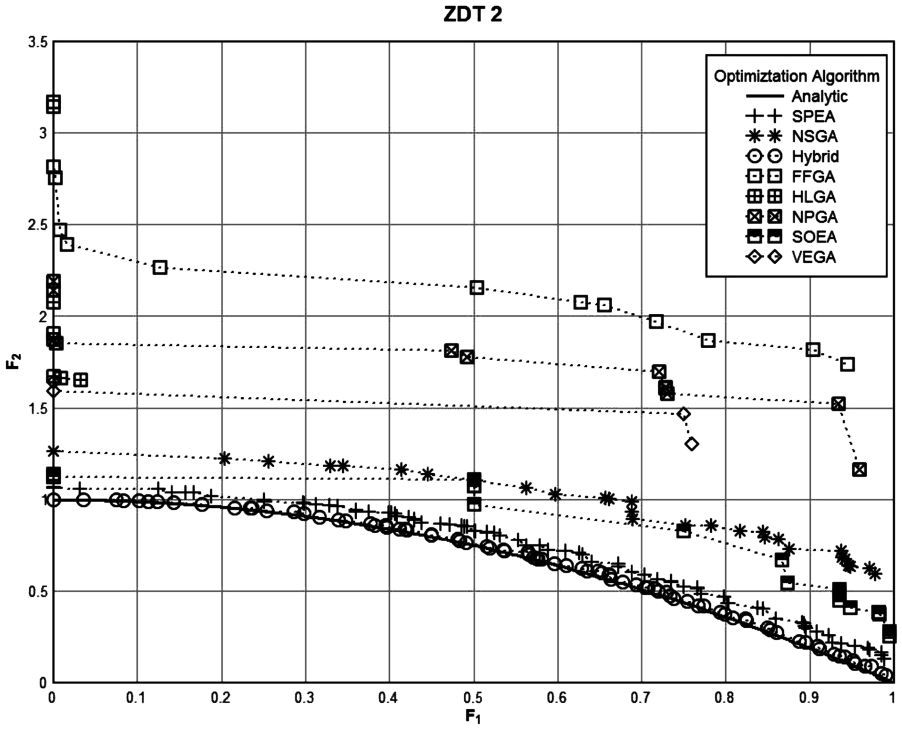

Fig. 2 Results for test problem 2 from ZDT with MOHO results added (circles).

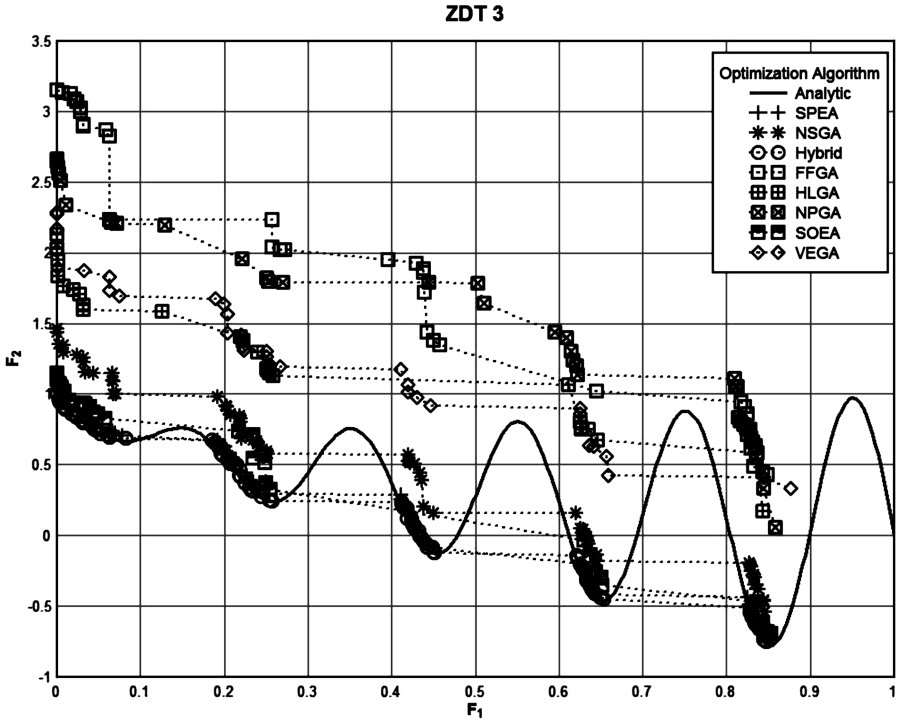

Fig. 3 Results for test problem 3 from ZDT with MOHO results added (circles).

$$C(A,B) = \frac{|\{a'' \in B; \exists a' \in A\colon a' \preceq a''\}|}{|B|} \tag{8}$$

Of note is the fact that $C(A,B)$ is not necessarily equal to $C(B,A)$.

Figure 7 shows the results for the cover function analysis for all test problems and optimization algorithms compared. In this analysis, all 30 runs of each algorithm are used. The reference and random results were removed, because the purpose of this work is not to show the effectiveness of evolutionary algorithms in general. Each cell in Fig. 7 is made by treating the algorithms in the row as algorithm A in Eq. (8). Each of the 30 Pareto approximations for algorithm A is compared with the algorithm B (column). So for each Pareto approximation of algorithm A, comparisons with 30 algorithm B Pareto sets are performed. This means that 30 cover function values are generated. Each column in the box plot (Fig. 7) represents the results for one of the ZDT problems. So each column in the box plot represents a total of 900 Pareto approximation set comparisons.

Using the data generated from solving the ZDT test problems, it was possible to perform another analysis of the MOHO algorithm's

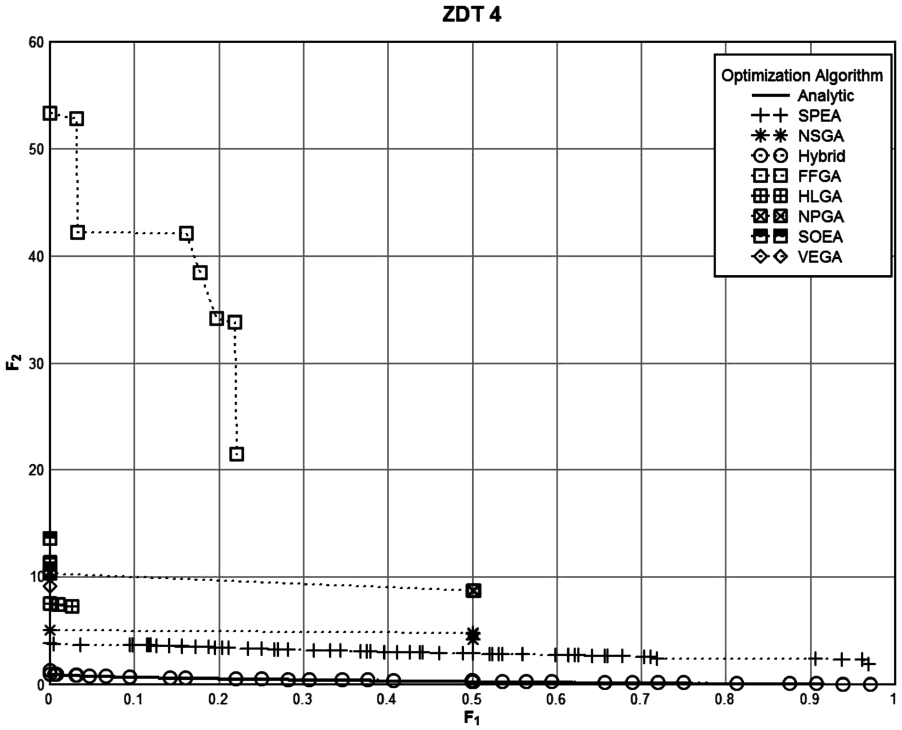

Fig. 4 Results for test problem 4 from ZDT with MOHO results added (circles).

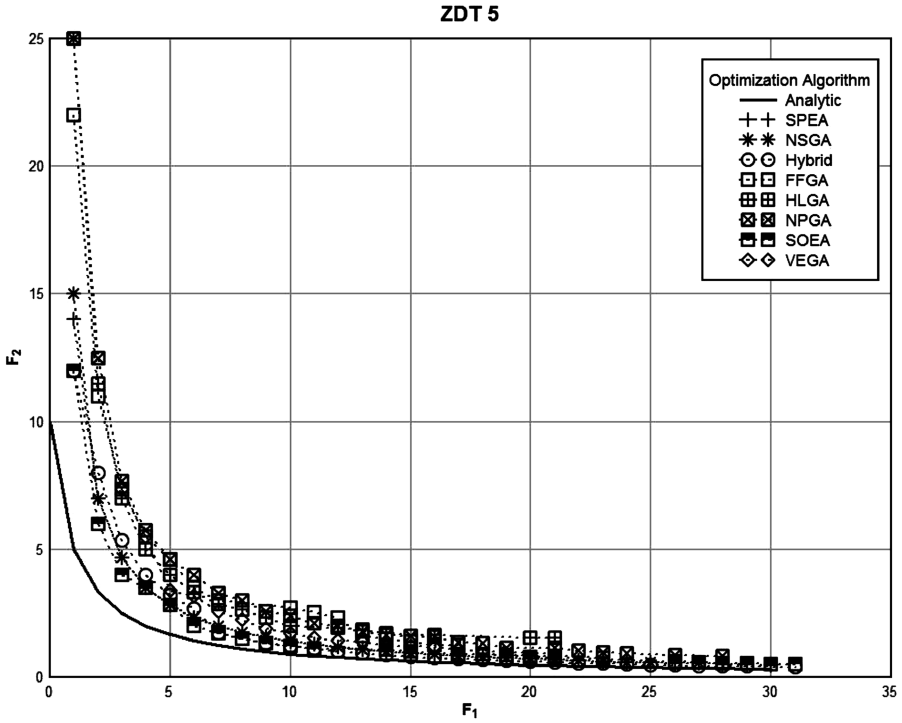

Fig. 5 Results for test problem 5 from ZDT with MOHO results added (circles).

performance. Figure 8 shows the average percent of the total number of function evaluations used by each of the three constituent search algorithms in MOHO when applied to a given ZDT test problem. Figure 8 takes into account all 30 runs used for the ZDT data comparison.

In the original NSGA-II paper, Deb et al. [7] presented two measures for evaluating the performance of multi-objective optimization algorithms. The gamma parameter measures the average Euclidian distance of each point in an algorithm's Pareto approximation from the actual Pareto front of the problem. This parameter is designed to give an idea of the accuracy of an algorithm's Pareto approximation.

The second measure for evaluating the performance is the diversity metric Δ, which measures the spread and uniformity of the Pareto approximation. The equation for this measure is given in Eq. (9):

$$\Delta = \frac{d_f + d_l + \sum_{i=1}^{N-1} |d_i - d_{\text{AVE}}|}{d_f + d_l + (N-1)d_{\text{AVE}}} \tag{9}$$

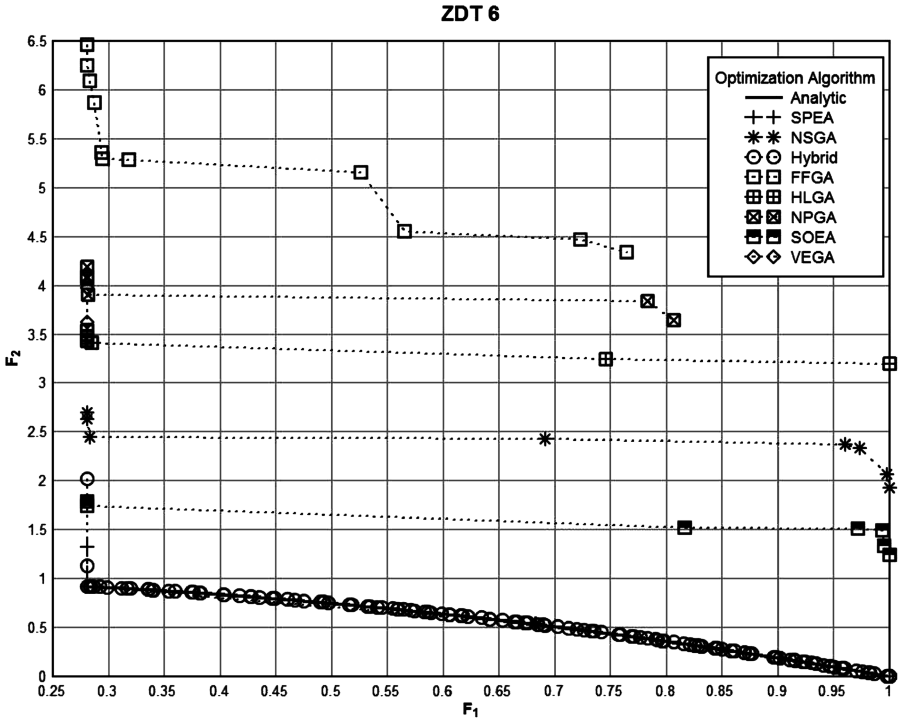

Fig. 6 Results for test problem 6 from ZDT with MOHO results added (circles).

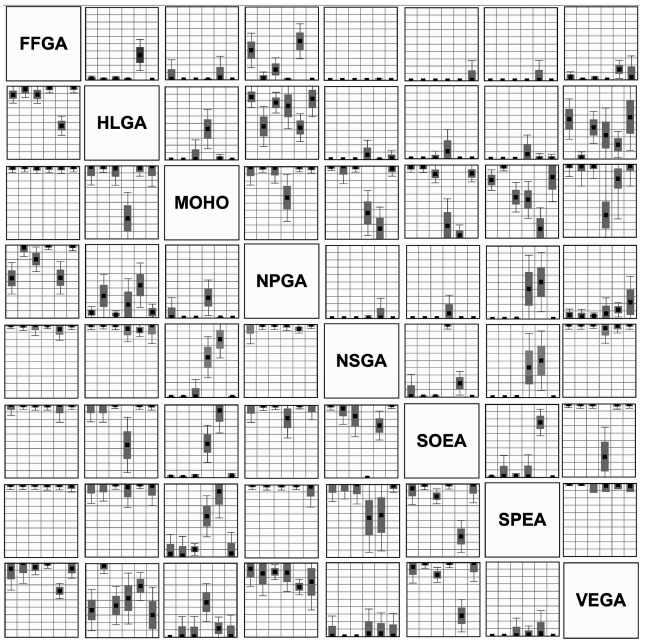

Fig. 7 Cover function analysis from ZDT with MOHO data inserted into the original analysis. As in the original analysis the value at the bottom of the graph is 0 and the value is 1 at the top. The black boxes in the box plots show the average cover value for all 30 runs. The shaded boxes and the whiskers display the one-standard-deviation spread and two-standard-deviation spreads, respectively, (i.e., ±0.5 standard deviations and ±1 standard deviation). The plots are read left to right: the leftmost box shows the results for ZDT1 and rightmost box shows the results for ZDT6.

Deb et al. [7] used this measure to analyze the Pareto approximations for two-objective optimization problems. To use this measure, the Pareto approximation to be analyzed must be sorted by the first objective in ascending order. The parameter d_f is the Euclidian distance between the first Pareto approximation point and the true Pareto front point with the minimum first objective value. In kind, the parameter d_l is the Euclidian distance from the last Pareto approximation point to the true Pareto point with the largest first objective value.

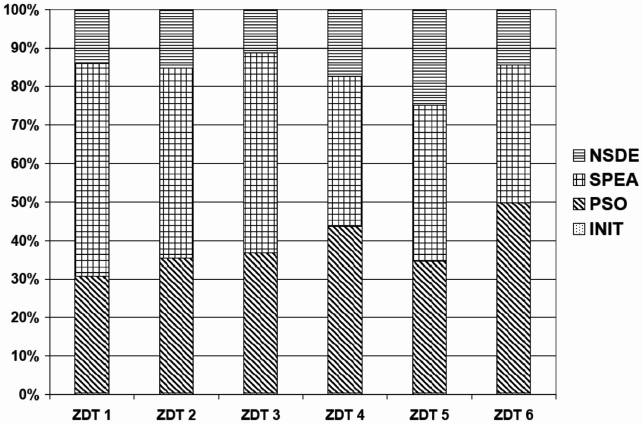

Fig. 8 Average percent of total execution time that each constituent optimization algorithm was used in MOHO for each of the six ZDT test problems. The average is taken over 30 runs used in the ZDT comparison.

A "perfect" Pareto approximation would have a Δ value of zero. This would mean that the first and last Pareto approximation points lie on the end points of the true Pareto front, $d_f = d_l = 0$ and that Pareto approximation is perfectly distributed so that all $d_i = d_{\text{AVE}}$.

Deb et al. [7] used these measures to analyze NSGA-II's performance on the ZDT1, ZDT2, ZDT3, ZDT4, and ZDT6 test problems previously defined. The performance of NSGA-II on the well-known Fonseca and Fleming [17], Kursawe [18], Poloni and Pediroda [19], and Schaffer [20] two-objective test problems was also analyzed in that work. These problems are defined in Eqs. (10–13).

Fonesca:

$$f_1 = x_1 \qquad g = 1 + 9 \cdot \sum_{i=2}^{m} \frac{x_i}{m-1} \qquad h = 1 - \sqrt{\frac{f_1}{g}} \qquad f_2 = h \cdot g \tag{10}$$

where the variable range is $m = 3$ and $x \in [-4, 4]$, and the ideal optima are $x1 = x2 = x3$ and $[-(1/\sqrt{3}), (1/\sqrt{3})]$.

Kursawe:

$$f_1 = \sum_{i=1}^{n-1}\left(-10\exp\left(-0.2\sqrt{x_i^2 + x_{i+1}^2}\right)\right)$$
$$f2 = \sum_{i=1}^{n}(|x_i|^{0.8} + 5\sin x_i^3) \tag{11}$$

where the variable range is $m = 3$ and $x \in [-5, 5]$, and the ideal optima are found using MOHO for 2×10^6 function evaluations.

Poloni:

$$\begin{aligned} f_1 &= [1 + (A_1 - B_1) + (A_2 - B_2)] \\ f_2 &= [(x_1 + 3)^2 + (x_2 + 1)^2] \\ A_1 &= 0.5\sin 1 - 2\cos 1 + \sin 2 - 1.5\cos 2 \\ A_2 &= 1.5\sin 1 - \cos 1 + 2\sin 2 - 0.5\cos 2 \\ B_1 &= 0.5\sin x_1 - 2\cos x_1 + \sin x_2 - 1.5\cos x_2 \\ B_2 &= 1.5\sin x_1 - \cos x_1 + 2\sin x_2 - 0.5\cos x_2 \end{aligned} \tag{12}$$

where the variable range is $m = 2$ and $x \in [-\pi, \pi]$ and the ideal optima are found using MOHO for 2×10^6 function evaluations.

Schaffer:

$$f_1 = x^2 \qquad f_2 = (x - 2)^2 \tag{13}$$

where the variable range is $x \in [-10^3, 10^3]$ and the ideal optima are $x \in [0, 2]$.

In Table 2, the results for the work by Deb et al. [7] are shown alongside MOHO results for the problems from that paper. The same conditions were used: population size of 100 processed through 250 generations, for a total of 25,000 function evaluations.

As mentioned earlier, Vrugt and Robinson [6] created a high-level-teamwork hybrid optimization algorithm named AMALGAM that works by running all of the constituent search algorithms simultaneously, thus requiring a multiprocessor. Notice that this is conceptually different from our MOHO, in which the constituent search algorithms are run sequentially, thus requiring only one processor. Each constituent search algorithm in AMALGAM is preassigned a number of new offspring to create. The offspring are then combined with the old population and the fast nondominated sorting from NSGA-II is applied to the combined population. In the work introducing AMALGAM, the algorithm was compared with NSGA-II (real-coded) using the gamma and delta metrics defined in the work by Deb et al. [7]. An analysis very similar to the one found in Deb et al. was performed, except that the algorithms were run for 150 generations, or 15,000 function evaluations. Table 3 shows the data from the AMALGAM compared with NSGA-II (real-coded). Added to this table are our MOHO results for the same test conditions used by Vrugt and Robinson [6].

V. Future Work

During our next investigation, the software will be tested against more test problems and some real-world design optimization problems. From this work, it became apparent that more algorithm measurement methodologies need to be investigated. This will be necessary to avoid the discrepancies with the NSGA-II results found in Tables 2 and 3. It is apparent that a researcher-prepared, well-distributed, ideal Pareto front can be interpreted to mean many things.

Two sets of improvements will be made to the software. First, another constituent evolutionary search algorithm will be added to our MOHO. Careful consideration will be made to insure that the new search algorithm's principle of operation differs as much as possible from the three constituent search algorithms already used in MOHO. This will be done in an attempt to broaden the types of objective functions that the hybrid algorithm will be able to handle efficiently. The final planned improvement will deal with the clustering algorithm. An attempt to normalize the clustering distance will be made so that Pareto fronts covering many orders of magnitude will not affect the searches.

Table 2 Performance comparison of NSGA-II and MOHO for 25,000 function evaluations

	NSGA-II (real) [5]		NSGA-II (binary) [5]		MOHO	
Problem Name	Gamma	Delta	Gamma	Delta	Gamma	Delta
Fonseca and Fleming [17]	0.0019	0.38	0.0025	0.45	0.0057	0.33
Kursawe [18]	0.0290	0.41	0.0290	0.40	0.0348	0.37
Poloni and Pediroda [19]	0.0155	0.45	0.0170	0.50	0.0501	0.88
Schaffer [20]	0.0034	0.48	0.0028	0.44	0.0038	0.34
ZDT1	0.0335	0.39	0.0009	0.46	0.0177	0.35
ZDT2	0.0724	0.43	0.0009	0.44	0.0130	0.34
ZDT3	0.1145	0.73	0.0434	0.58	0.6348	0.59
ZDT4	0.5130	0.70	3.2276	0.48	10.07	0.97
ZDT6	0.2966	0.67	7.8068	0.64	0.1696	0.95

Table 3 NSGA-II, AMALGAM, and MOHO performance comparison for 15,000 function evaluations

	NSGA-II (real) [3]		AMALGAM [3]		MOHO	
Problem name	Gamma	Delta	Gamma	Delta	Gamma	Delta
Fonseca and Fleming [17]	0.0026	0.38	0.0017	0.33	0.006	0.33
Kursawe [18]	0.0108	0.48	0.0099	0.47	0.037	0.37
Schaffer [20]	0.0036	0.49	0.0032	0.37	0.004	0.34
ZDT1	0.0053	0.34	0.0011	0.33	0.0158	0.35
ZDT2	0.0068	0.36	0.0009	0.35	0.0128	0.34
ZDT3	0.0027	0.56	0.0010	0.55	0.600	0.62
ZDT4	0.0523	0.73	0.0022	0.32	14.6	0.99
ZDT6	0.0504	0.53	0.0011	0.40	0.289	1.00

VI. Conclusions

For the first results comparing MOHO with the evolutionary multi-objective optimization algorithms from the ZDT work, it was shown that MOHO can outperform the classic evolutionary algorithms for all test problems except ZDT5. In their work, Zitzler et al. [16] point out that the test problem ZDT5, as originally stated, did not work out for the purposes of their numerical testing and it changed the target Pareto from the original problem definition, as shown in Table 1 and Fig. 5 of this work. Difficulties surrounding the ZDT5 test case are not trivial, as was confirmed by Deb et al. [7], who tested all the ZDT test problems except ZDT5.

In the second set, composed of four test cases (Table 2), MOHO's performance was comparable with the real-coded NSGA-II, except in the case of ZDT4. Figure 4 shows that ZDT4 has an almost vertical portion of the nondominated front at which f_1 goes to its minimum value. This portion of the front has an abrupt change in the value of the second objective. It is difficult to distribute ideal Pareto front points in this portion of the front. This accounts for the large gamma value for MOHO in ZDT4. The same phenomenon occurs in ZDT6, in which it affected the gamma value for the binary-coded NSGA-II.

Finally, when comparing MOHO with the method of Vrugt and Robinson [6], MOHO does not seem to perform as well as the AMALGAM algorithm or NSGA-II. These results are for 150 generations, or 15,000 function evaluations. What is curious is that the results show a tenfold improvement of performance in gamma when using NSGA-II on the ZDT problems, while using 40% fewer function evaluations. The only explanation for these results is that the methodology to prepare ideal Pareto fronts in the work of Vrugt and Robinson [6] was completely different from the methodology used by Deb et al. [7] and in this work.

Acknowledgments

The authors are grateful for the financial support provided for this work by the U.S. Air Force Office of Scientific Research under grant FA9550-06-1-0170, monitored by Todd E. Combs, Fariba Fahroo, and Donald Hearn, and by the U.S. Army Research Office/Materials Division under contract W911NF-06-1-0328, monitored by William M. Mullins. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the U.S. Air Force Office of Scientific Research, the U.S. Army Research Office, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for government purposes, notwithstanding any copyright notation thereon.

References

[1] Dulikravich, G. S., Martin, T. J., Dennis, B. H., and Foster, N. F., "Multidisciplinary Hybrid Constrained GA Optimization," *EUROGEN'99—Evolutionary Algorithms in Engineering and Computer Science: Recent Advances and Industrial Applications*, edited by K. Miettinen, M. M. Makela, P. Neittaanmaki, and J. Periaux, Wiley, New York, 30 May–3 June 1999, pp. 233–259.

[2] Dulikravich, G. S., Moral, R. J., and Sahoo, D., "A Multi-Objective Evolutionary Hybrid Optimizer," *Evolutionary and Deterministic Methods for Design Optimization and Control with Applications to Industrial and Societal Problems (EUROGEN 2005)* [CD-ROM], edited by R. Schilling, W. Haase, J. Periaux, H. Baier, and G. Bugeda, Technical Univ. of Munich, Munich, 12–14 Sept. 2005, Paper C4-1.

[3] Moral, R. J., Sahoo, D., and Dulikravich, G. S., "Multi-Objective Hybrid Evolutionary Optimization with Automatic Switching," 11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Portsmouth, VA, AIAA Paper 2006-6976, 6–8 Sept. 2006.

[4] Curteanu, S., Leon, F., and Gâlea, D., "Alternatives for Multiobjective Optimization of a Polymerization Process," *Journal of Applied Polymer Science*, Vol. 100, No. 5, 2006, pp. 3680–3695. doi:10.1002/app.23205

[5] Talbi, E.-G., "A Taxonomy on Hybrid Metaheuristics," *Journal of Heuristics*, Vol. 8, 2002, pp. 541–562. doi:10.1023/A:1016540724870

[6] Vrugt, J. A., and Robinson, B. A., "Improved Evolutionary Optimization from Genetically Adaptive Multimethod Search," *Proceedings of the National Academy of Sciences (U.S.)*, Vol. 104, No. 3, 2007, pp. 708–711. doi:10.1073/pnas.0610471104

[7] Deb, K., Pratap, A., Agarwal, S., and Meyrivan, T., "A Fast and Elitist Multi-Objective Genetic Algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, Vol. 6, No. 2, 2002, pp. 182–197. doi:10.1109/4235.996017

[8] Sobol, I. M., "Uniformly Distributed Sequences with an Additional Uniform Property," *USSR Computational Mathematics and Mathematical Physics* , Vol. 16, 1976, pp. 236–242. doi:10.1016/0041-5553(76)90154-3

[9] Bratley, P., and Fox, B., "Algorithm 659: Implementing Sobol's Quasirandom Sequence Generator," *ACM Transactions on Mathematical Software*, Vol. 14, No. 1, Mar. 1988, pp. 88–100. doi:10.1145/42288.214372

[10] Zitzler, E., Laumanns, M., and Thiele, L., "SPEA2: Improving the Strength Pareto Evolutionary Algorithm," Dept. of Electrical Engineering, Swiss Federal Inst. of Technology, Rept. TIK-103, Zurich, 2001, pp. 1–21.

[11] Eberhardt, R., Shi, Y., and Kennedy, J., *Swarm Intelligence*, Morgan Kaufmann, San Mateo, CA, 2001.

[12] Storn, R., and Price, K. V., "Minimizing the Real Function of the ICEC '96 Contest by Differential Evolution," *IEEE Conference on Evolutionary Computation*, Inst. of Electrical and Electronics Engineers, Piscataway, NJ, 1996, pp. 842–844.

[13] Parsopoulos, K., and Vrahatis, M., "Particle Swarm Optimization Method in Multi-Objective Problems," *Proceedings of the 2002 ACM Symposium on Applied Computing (SAC 2002)*, Association for Computing Machinery, New York, 2002, pp. 603–607.

[14] 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 Computation*, Vol. 7, No. 2, 2003, pp. 117–132. doi:10.1109/TEVC.2003.810758

[15] Deb, K., *Multi-Objective Optimization Using Evolutionary Algorithms*, Wiley, New York, 2002.

[16] Zitzler, E., Deb, K., and Thiele, L., "Comparison of Multiobjective Evolutionary Algorithms: Empirical Results," *Evolutionary Computation*, Vol. 8, No. 2, 2000, pp. 173–195. doi:10.1162/106365600568202

[17] Fonseca, C. M., and Fleming, P. J., "An Overview of Evolutionary Algorithms in Multi-Objective Optimization," *Evolutionary Computation*, Vol. 3, No. 1, 1995, pp. 1–16. doi:doi:10.1162/evco.1995.3.1.1

[18] Kursawe, F., "A Variant of Evolution Strategies for Vector Optimization," *Parallel Problem Solving from Nature (PPSN 1)*, Lecture Notes in Computer Science, Vol. 496, Springer–Verlag, Berlin, 1990, pp. 193–197.

[19] Poloni, C., and Pediroda, V., *GA Coupled with Computationally Expensive Simulations: Tools to Improve Efficiency*, *Genetic Algorithms and Evolution Strategy in Engineering and Computer Science: Recent Advances and Industrial Applications (EUROGEN 1997)*, edited by Quagliarella, D., Periaux, J., Poloni, C., and Winter, G., Wiley, New York, 1997, Chap. 13, pp. 267–288.

[20] Schaffer, J. D., "Multiple Objective Optimization with Vector Evaluated Genetic Algorithm," *Proceedings of the 1st International Conference on Genetic Algorithms*, Lawrence Erlbaum Associates, Mahwah, NJ, 1987, pp. 93–100.

A. Messac
Associate Editor