George S. Dulikravich
Full Professor
Florida International University
Dept. of Mechanical & Materials Eng.
MAIDROC Lab., Miami, Florida, USA

Thomas J. Martin
Senior Design Engineer
Hot Section Advanced Methods
Pratt & Whitney, a division of UTC
East Hartford, Connecticut, USA

Marcelo J. Colaço
Associate Professor
Federal University of Rio de Janeiro
Faculty of Mechanical Engineering
Rio de Janeiro, Brazil

Eric J. Inclan
Research Assistant
Florida International University
Dept. of Mechanical & Materials Eng.
MAIDROC Lab., Miami, Florida, USA

Automatic Switching Algorithms in Hybrid Single-Objective Optimization

Hybrid optimization algorithms consist of a number of proven constituent optimization algorithms and a control algorithm that performs automatic switching among the constituent algorithms at each stage during the optimization when the rate of convergence becomes unsatisfactory, the process tends towards a local minimum, or some other undesirable aspect of the iterative process appears. Thus, hybrid optimization algorithms that utilize a number of gradient based and non-gradient based constituent optimizers are more robust and converge better than individual constituent optimization algorithms. The logic of designing the automatic switching algorithms in hybrid optimizers is surveyed in this paper focusing on the research performed by the authors in the area of hybrid single-objective optimization initiated in 1997.

Keywords: *optimization algorithms, hybrid optimization, switching algorithms, minimization, single-objective optimization.*

1. INTRODUCTION

Realistic engineering problems always involve interaction of several disciplines such as fluid dynamics, heat transfer, elasticity, electromagnetism, dynamics, *etc*. Thus, realistic problems are always multidisciplinary and the geometric space is typically arbitrarily shaped and three-dimensional. Each of the individual disciplines is governed by its own system of differential equations or integral equations of different degree of non-linearity and based on often widely disparate time scales and length scales. All of these factors make a typical multidisciplinary optimization problem highly non-linear and interconnected. Consequently, an objective function space for a typical multidisciplinary problem could be expected to have a number of local minima. A typical multidisciplinary optimization problem, therefore, requires the use of optimization algorithms that can either avoid the local minima or escape from the local minima.

Optimization algorithms are *de facto* minimization algorithms, that is, algorithms that search for a global minimum and can be divided in three groups: a) gradient-based algorithms, b) non-gradient based (evolutionary algorithms), and c) hybrid otimization algorithms that combine the gradient-based and the non-gradient-based algorithms via an automatic switching algorithm. The objective of this survey paper is to elaborate on these switching algorithms.

Addition of constraints of both equality and inequality type to a typical multidisciplinary optimization problem reduces significantly the feasible domain of the objective function space. To find such often-small feasible function space, the optimizer should be able to initially search as large portion of the objective function space as possible. Non-gradient based optimizers are capable of performing this task. When equality constraints are to be enforced, the gradient-based optimizers can perform this task very accurately.

One of the primary concerns of any optimization algorithm is the computational effort required to achieve convergence. Except in the case of certain sensitivity based optimization algorithms and genetic algorithms with extremely large populations, the computer memory is not an issue. Typical constrained optimization problems in engineering require large number of objective function evaluations. Each function evaluation involves a very time-consuming computational analysis of the physical processes involved.

An equally important issue is the ability of an optimization algorithm to converge to the global minimum rather than immediate neighborhood of the minimum. Non-gradient based optimizers have these capabilities. On the other hand, once the neighborhood of the global minimum has been found, the non-gradient based optimizers have difficulty converging to the global minimum. For this purpose, it is more appropriate to use gradient-based optimizers.

2. OPTIMIZATION PROBLEM STATEMENT

The general single-objective, constrained, optimization problem can be mathematically stated as follows. Minimize the scalar objective function,

$$\mathrm{F(V)} \tag{1}$$

of a set of design variables,

$$\{V\} = \left\{ V_1 \quad V_2 \quad \cdots \quad V_{N\,\mathrm{var}} \right\}, \tag{2}$$

limited to their extreme individual ranges

$$\{V_{\min}\} < \{V\} < \{V_{\max}\} \tag{3}$$

subject to inequality constraints

Received: June 2013, Accepted: July 2013
Correspondence to: Prof. George S. Dulikravich
Dept. of Mechanical & Materials Eng., MAIDROC Lab.
10555 West Flagler St., Miami, Florida 33174, USA
E-mail: dulikrav@fiu.edu

$$g_m\left(V\right) \le 0\,,\; m = 1,N_{inc} \tag{4}$$

and equality constraints

$$\left|h_n\left(V\right)\right| \le \varepsilon_n\,,\; n = 1,N_{eqc} \tag{5}$$

Here, $\mathrm{F}\left(\vec{\mathrm{V}}\right)$ is the objective function, $\{V\}$ is the vector of $\mathrm{N_{var}}$ design variables, $\{V_{\min}\}$ is the vector of lower limit constraints, $\{V_{\max}\}$ is the vector of upper limit constraints, g_m is the set of N_{inc} inequality constraint functions, h_n is the set of N_{eqc} equality constraint functions, and ε is a very small number called the constraint thickness. The solution of an optimization problem is the set of design variables for which the objective function takes on its global minimum value,

$$F^*\left(V^*\right).$$

A set of design variables that does not violate any constraints is said to be feasible, while design variables that violate one or more constraints are infeasible. If a constraint is on the verge of being violated, it is said to be an active constraint. That is, active constraints satisfy the following relationships. Equality constraints are always active.

$$g_m\left(V\right) \le \varepsilon_m \tag{6}$$

$$\left|h_n\left(V\right)\right| \le \varepsilon_n \tag{7}$$

3. INDIVIDUAL PERFORMANCES OF SOME OF THE MOST COMMON OPTIMIZATION ALGORITHMS

As a demonstration of the superior performance of a hybrid optimizer when compared to individual optimization algorithms, we will first show the performance of several of the most common optimizers to find the optimum of the Griewank's function #8 [1], which is defined as

$$f = \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \qquad x \in\ ]-600,600[\tag{8}$$

The global minimum for this function is located at **x** = 0 and is $f(\mathbf{x}) = 0$. For a two-dimensional test case, it is shown in Figure 1 in three levels of local resolution. One can see that this function has an extremely large number of local minima, making the optimization task of finding the global minimum a serious challenge.

The following single-objective minimization algorithms [2-4] were individually tested to access their individual performance when attempting to find the global minimum and its location for this test function.

Figure 2 shows the results for this optimization task using separately: (a) Broyden-Fletcher-Goldfarb-Shanno (BFGS) quasi-Newton method [5], (b) Differential Evolution (DE) algorithm [6], (c) Simulated Annealing (SA) algorithm, [7] (d) Particle Swarm (PS) algorithm [8], and (e) our fourth generation hybrid optimization algorithm. Evolutionary methods performed somewhat better than the best gradient-based algorithm (BFGS).

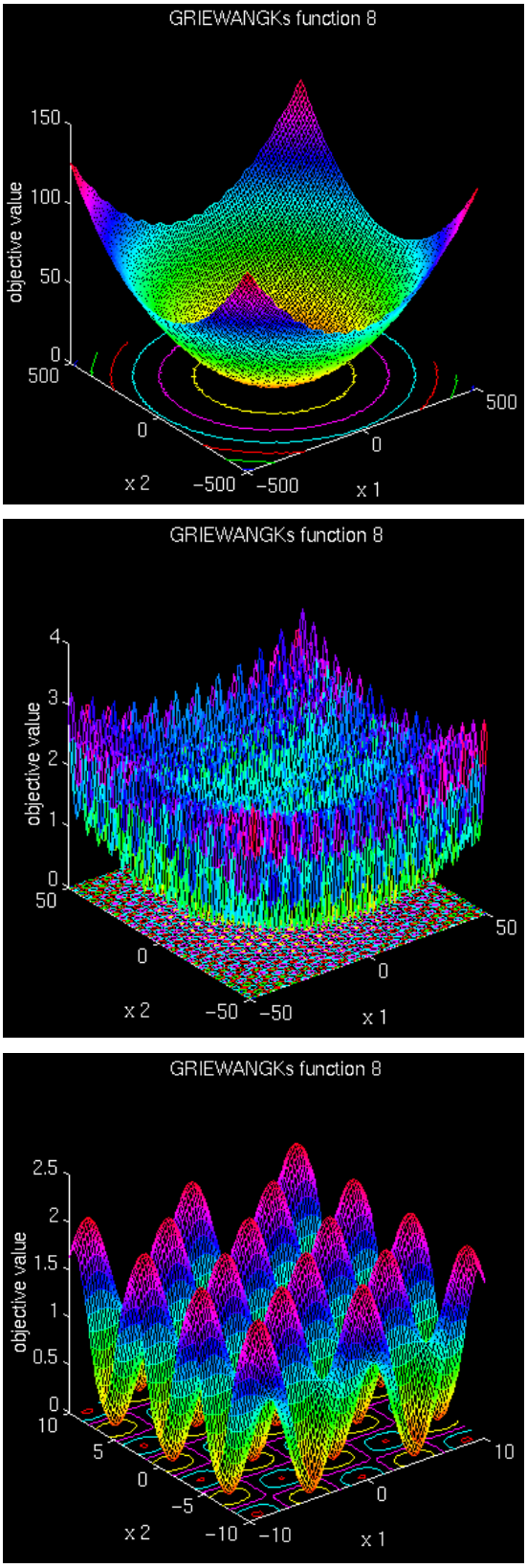
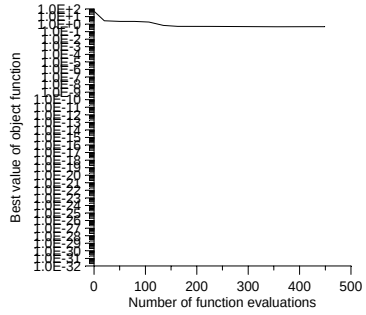
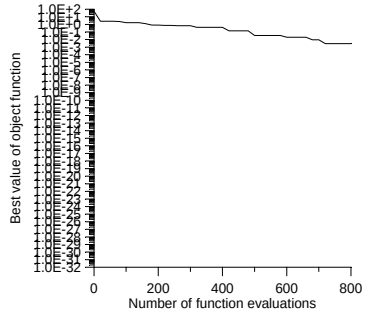


Figure 1: Griewank's function #8: global view, intermediate view, local view.

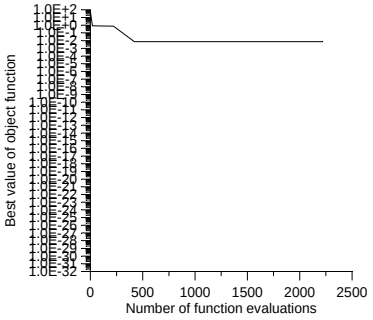
However, only the hybrid optimization algorithm was capable of locating the global optimum value of this function and to determine its global minimum value with satisfactory accuracy.



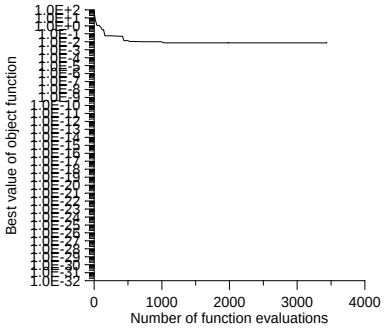
a) Convergence history for BFGS



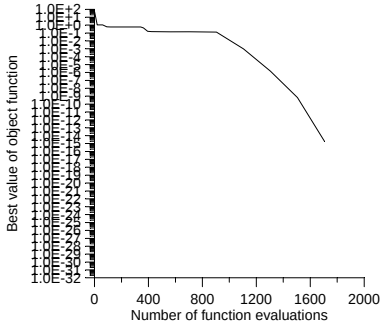
b) Convergence history for DE



c) Convergence history for SA



d) Convergence history for PS



e) Convergence history for hybrid optimizer

Figure 2: Comparison of performances of various optimizers for the Griewank's function #8 [1].

This suggests that it might be beneficial to utilize several different optimization algorithms during different phases of the optimization process, since "no free lunch theorem" [9] definitely holds, that is, no single optimization algorithm is better than all the other optimizers for all classes of optimization problems. Various optimization algorithms have been known to provide faster convergence over others depending upon the size and topology of the design space, the type of the constraints, and where they are during the optimization process. Each algorithm provides a unique approach to optimization with varying degrees of convergence, reliability, and robustness at different stages during the iterative optimization process.

Hybrid optimization algorithms combine individual constituent optimization algorithms in a sequential or parallel manner so the resulting software can utilize the advantages of each constituent algorithm. That is, single-objective optimization constituent algorithms that rely on different principles of operation are combined with a set of measures to perform automatic switching among the constituent algorithms. This allows the software to choose the most effective constituent algorithm for the design problem at hand. The automatic back-and-forth switching among several optimization algorithms can be viewed as a backup strategy [10] so that, if one optimization method fails, another optimization algorithm can automatically take over. Following is a discussion of various automatic switching strategies among the constituent optimizers.

4. FIRST GENERATION OF SINGLE-OBJECTIVE HYBRID OPTIMIZATION ALGORITHMS WITH AUTOMATIC SWITCHING

The first hybrid, single-objective, constrained optimization algorithm was developed and used in the period 1995-1998 [10-13]. It had two constitutive optimization algorithms: Davidon-Fletcher-Powell (DFP) gradient search algorithm [14, 15] and genetic algorithm (GA) [16]. Initial search of the objective function space was performed using GA. Once the code was showing signs of slow convergence, it was automatically switched to DFP. When DFP algorithm's convergence rate dropped below a specified minimum value, this hybrid optimizer switched automatically back to GA algorithm. This back-and-forth automatic switching successfully avoided premature termination of the overall optimization process in a local minimum and continued to the global minimum.

Preliminary results obtained with different versions of a hybrid optimizer that uses a GA for the overall logic, a quasi-Newtonian gradient-search algorithm or a feasible directions method to ensure monotonic reduction of the objective function, and a Nelder-Mead sequential simplex algorithm or a steepest descent methodology of the design variables into feasible regions from infeasible ones has proven to be effective at avoiding local minima. Since the classical GA does not ensure monotonic decrease of the objective function, the hybrid optimizer could store information gathered by the genetic search and use it to determine the sensitivity derivatives of the objective function and all

constraint function. When enough information has been gathered and the sensitivity derivatives are known, the optimizer automatically switches to the feasible directions method (with quadratic subproblem) thus quickly proceeding to further acceleraate the iterative search process.

One possible scenario for such a hybrid algorithm can be summarized as follows:

1) Let the set of population (candidate solutions) members define a simplex like that used in the Nelder-Mead method.
2) If the fitness evaluations for all of the population members does not yield a better solution, then define a search direction as described by the Nelder-Mead method.
3) If there are active inequality constraints, compute their gradients and determine a new search direction by solving the quadratic subproblem.
4) If there are active equality constraints, project this search direction onto the subspace tangent to the constraints.
5) Perform line search.

5. SECOND GENERATION OF SINGLE-OBJECTIVE HYBRID OPTIMIZATION ALGORITHMS WITH AUTOMATIC SWITCHING

The second generation of our hybrid optimization algorithms was developed in the late 1990s [17-22]. It had four constitutive optimization algorithms: Davidon-Fletcher-Powell (DFP) gradient search [14, 15], Genetic Algorithm (GA) [16], Nelder-Mead (NM) simplex algorithm [23], and Simulated Annealing (SA) [7]. Automatic switching among the four constituent algorithms was performed using heuristics (Figure 3).

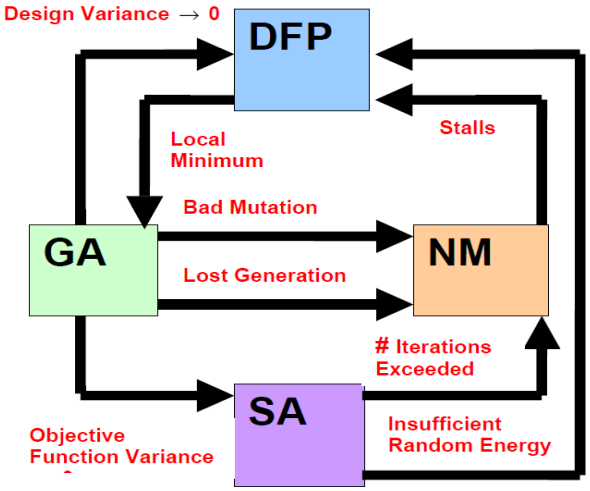


Figure 3. Automatic switching logic among constituent optimization algorithms in the second generation of our hybrid single-objective optimization algorithms.

6. THIRD GENERATION OF SINGLE-OBJECTIVE HYBRID OPTIMIZATION ALGORITHMS WITH AUTOMATIC SWITCHING

This version of the hybrid optimizer was developed and used briefly during 1998 [24, 25]. It incorporated five constitutive optimization algorithms: Davidon-Fletcher-Powell (DFP) gradient-based algorithm [14, 15], Genetic Algorithm (GA) [16], modified Nelder-Mead (NM) simplex algorithm [23], Simulated Annealing (SA) [7] and Sequential Quadratic Programming (SQP) [26].

7. FOURTH GENERATION OF SINGLE-OBJECTIVE HYBRID OPTIMIZATION ALGORITHMS WITH AUTOMATIC SWITCHING

The fourth generation of our hybrid optimization algorithms was developed in the early 2000s [17-22]. It had the following six constituent optimization modules: Davidon-Fletcher-Powell (DFP) gradient-based algorithm [14, 15], Genetic Algorithm (GA) [16], Nelder-Mead (NM) simplex algorithm [23], Differential Evolution (DE) algorithm [6], Sequential Quadratic Programming (SQP) [26] and quasi-Newton algorithm of Pshenichny-Danilin (LM) [27]. Thus, this hybrid optimizer had three gradient-based and three non-gradient-based constituent optimization algorithms that are automatically switching back-and-forth as depicted in Figure 4.

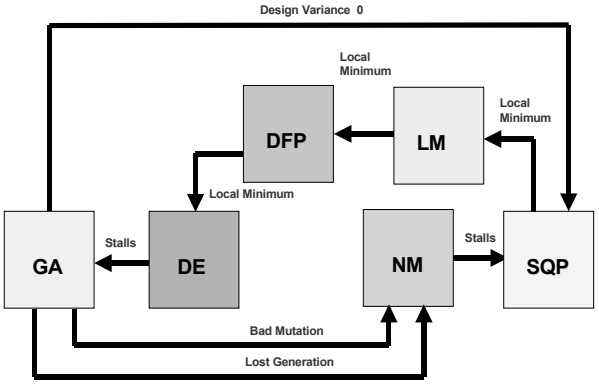


Figure 4: Switching procedure for the third generation of our hybrid single-objective optimization algorithms.

The evolutionary hybrid scheme handled the existence of equality and inequality constraint functions, $g(\vec{V})$ and $h(\vec{V})$, in three ways: Rosen's projection method, feasible search, and random design generation. Rosen's projection method [28, 2-4, 10] provided search directions which guided the descent direction tangent to active constraint boundaries. In the feasible search [10], designs that violate constraints were automatically restored to feasibility via the minimization of the active global constraint functions. If at any time this constraint minimization failed, random designs were generated within a Gaussian-shaped probability density cloud about a desirable and feasible design until a new design is reached.

Gradients of the objective and constraint functions with respect to the design variables, $\partial F/\partial \vec{V}$, $\partial g/\partial \vec{V}$, and $\partial g/\partial \vec{V}$ (also called design sensitivities), were calculated using either forward (first order) finite difference formulas, or by the efficient method of implicit differentiation of the governing equations [29].

The population matrix was updated every iteration with new designs and ranked according to the value of the objective function. As the optimization process

proceeded, the population evolved towards the global minimum. The optimization problem was completed when one of several stopping criterion was met: (1) the maximum number of iterations or objective function evaluations were exceeded, (2) the best design in the population was equivalent to a target design, or (3) the optimization program tried all four algorithms but failed to produce a non-negligible decrease in the objective function. The latter criterion was the primary qualification of convergence and it usually indicated that a global minimum had been found.

8. FIFTH GENERATION OF SINGLE-OBJECTIVE HYBRID OPTIMIZATION ALGORITHMS WITH AUTOMATIC SWITCHING

The hybrid optimization algorithm, called *H1*, was developed in 2004 by combining three of the fastest gradient-based and evolutionary optimization algorithms [2, 30]. It is quite simple conceptually, although its computational implementation is more involved. The global procedure is illustrated in Figure 5. It uses the concepts of three different methods of optimization, namely: the Broyden-Fletcher-Goldfarb-Shanno (BFGS) quasi-Newton algorithm [5], the Differential Evolution (DE) algorithm [6] and the Particle Swarm (PS) algorithm [8].

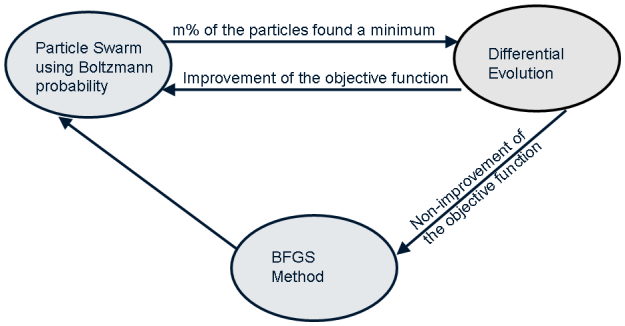


Figure 5: Switching procedure for the fifth generation of our hybrid single-objective optimization algorithms.

The most often automatically used constituent optimization module is the Particle Swarm (PS) algorithm [8]. When certain percent of the particles find a minimum, the algorithm switches automatically to the Differential Evolution (DE) algorithm [6] and the particles are forced to breed. If there is an improvement in the objective function, the algorithm returns to the Particle Swarm method, meaning that some other region is more likely to have a global minimum.

If there is no improvement in the objective function value, this can indicate that this region already contains the global value expected and the algorithm automatically switches to the BFGS algorithm [5] in order to quickly and accurately find the location of the minimum.

In order to speed-up the optimization, the procedure is repeated using sequential computational grid refinement [30] approach starting with PS algorithm.

In the hybrid optimizer *H1*, when a certain percent of the particles find a minimum, the algorithm switches automatically to the DA method and the particles are forced to breed. If there is an improvement in the objective function, the algorithm returns to the PS method, meaning that some other region is more prone to having a global minimum. If there is no improvement of the objective function, this can indicate that this region already contains the global value expected and the algorithm automatically switches to the BFGS method in order to find its location more precisely. In Figure 5, the algorithm returns to the PS method in order to check if there are no changes in this location and the entire procedure repeats itself. After some maximum number of iterations is performed (*e.g.,* five) the process stops.

9. SIXTH GENERATION OF SINGLE-OBJECTIVE HYBRID OPTIMIZATION ALGORITHMS WITH AUTOMATIC SWITCHING AND RESPONSE SURFACE

The hybrid optimizer *H2* [31, 4] is quite similar to the *H1*, except by the fact that is uses a response surface method at some point of the optimization task [4]. The global procedure is illustrated in Figure 6. It can be seen from this figure that after a certain number of objective functions were calculated, all this information was used to obtain a response surface. Such a response surface is then optimized using the same hybrid code defined in the *H1* optimizer so that it fits the calculated values of the objective function as closely as possible. New values of the objective function are then obtained very cheaply by interpolating their values from the response surface.

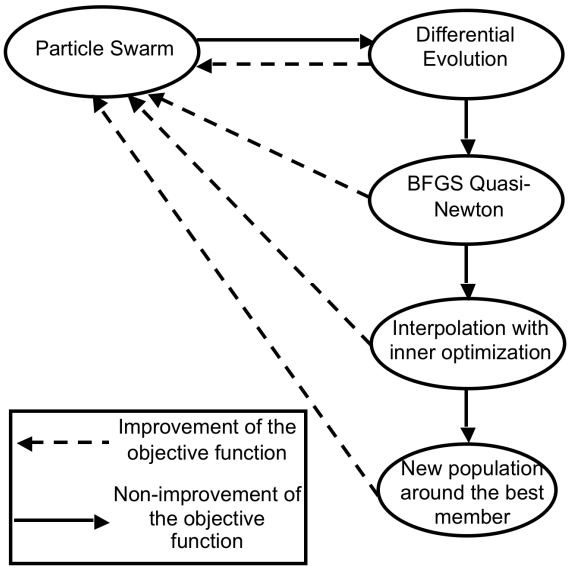


Figure 6: Global procedure for the hybrid optimization method *H2*

In Figure 6, if the BFGS cannot find any better solution, the algorithm uses a radial basis function interpolation scheme to obtain a response surface and then optimizes such response surface using the same hybrid algorithm proposed. When the minimum value of this response surface is found, the algorithm checks to see if it is also a solution of the original problem. Then, if there is no improvement of the objective function, the entire population is eliminated and a new population is generated around the best value obtained so far. The hybrid algorithm returns to the PS algorithm in order to check if there are no changes in this location and the

entire procedure repeats itself. After a specified maximum number of iterations is performed (*e.g.*, five) the process stops.

Hybrid optimizer called *H3* [31, 4] is an extension of *H1* and *H2*. The global procedure is outlined below:

1. Generate an initial population, using the real function (not the interpolated one) $f(x)$. Call this population P_{real}.
2. Determine the individidual that has the minimum value of the objective function, over the entire population P_{real} and call this individual x_{best}.
3. Determine the individual that is more distant from the x_{best}, over the entire population P_{real}. Call this individual x_{far}.
4. Generate a response surface [4] using the entire population P_{real} as support points. Call this function $s(x)$.
5. Optimize the interpolated function $s(x)$ using the hybrid optimizer *H1*, defined above, and call the optimum variable of the interpolated function as x_{int}. During the generation of the internal population to be used in the *H1* optimizer, consider the upper and lower bounds limits as the minimum and maximum values of the population P_{real} in order to not extrapolate the response surface.
6. If the real objective function $f(x_{int})$ is better than all objective function of the population P_{real}, replace x_{far} by x_{int}. Else, generate a new individual, using the Sobol's pseudo-random generator [32] within the upper and lower bounds of the variables, and replace x_{far} by this new individual.
7. If the optimum is achieved, stop the procedure. Else, return to step 2.

From the sequence above, one can notice that the number of times that the real objective function $f(x)$ is called is very small. Also, from step 6, one can see that the search space is reduced after each iteration. When it is no longer possible to find a better minimum on the current response surface, a new call to the real function $f(x)$ is made to generate a new point to be included in the pool of the support points and a new response surface is generated. Since the computing time to calculate the interpolated function (to read the approximate/interpolated value of $f(x)$ from the response surface $s(x)$) is very short, the *H2* optimizer is significantly faster than the *H1* optimizer which does not use the response surface.

The hybrid optimizer *H3* was compared against the optimizer *H1*, *H2* and the commercial code IOSO 2.0 for some standard test function which was the Levy #9 function [33]. It has 625 local minima and 4 variables. Such function is defined as

$$f(\mathbf{x}) = \sin^2(\pi - z_1) + \\ + \sum_{i=1}^{n-1}(z_i - 1)^2\left[1 + 10\sin^2(\pi z_{i+1})\right] + (z_4 - 1)^2 \tag{9}$$

$$z_i = 1 + \frac{x_i - 1}{4}, (i = 1, 4) \tag{10}$$

The function is defined within the interval $-10 \le x \le 10$ and its minimum is $f(x) = 0$ for $x = 1$. Figure 7 shows the optimization history of the IOSO, *H1*, *H2* and *H3* optimizers. Since the *H1*, *H2* and *H3* optimizers are based on random number generators (because of the PS module), we present the best and worst estimates for these three optimizers.

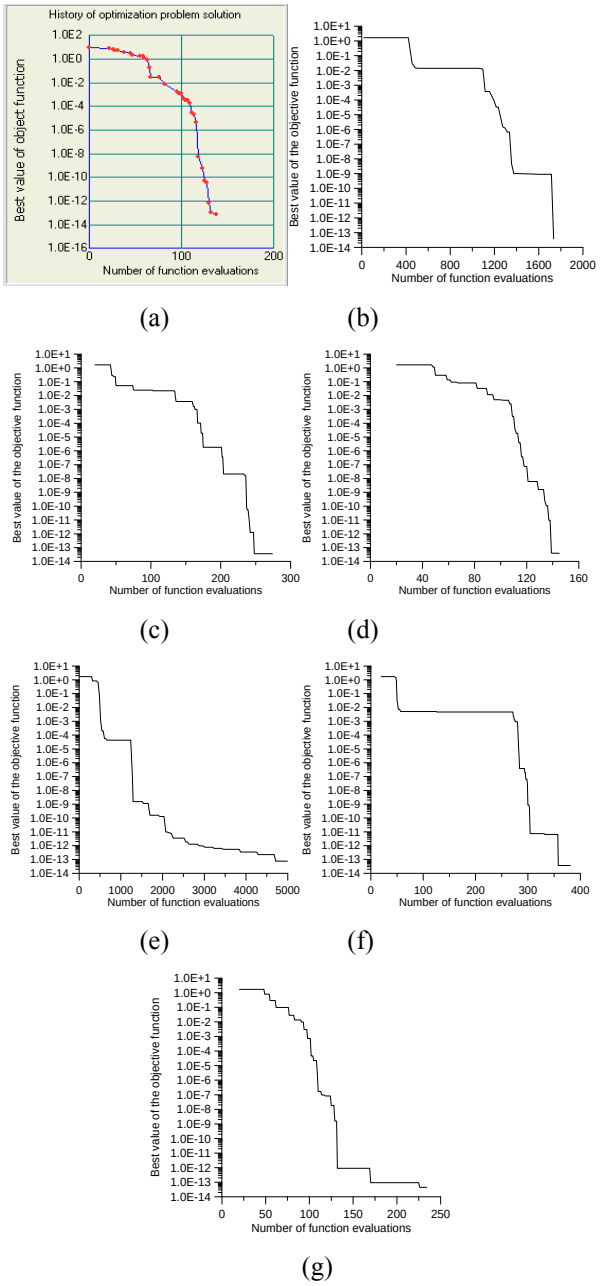


Figure 7 – Optimization history of the Levy #9 function using: (a)IOSO, (b)H1-best, (c)H2-best, (d)H3-best, (e)H1-worst, (f)H2-worst and (g)H3-worst hybrid optimizers of the fifth and sixth generation

From Figure 7, it can be seen that the performance of the *H3* optimizer is very close to the IOSO commercial code [34]. The *H1* code is the worst and the *H2* optimizer also has a reasonable good performance. It is interesting to note that the *H1* code is the only one that doesn't have a response surface model implemented.

10. AUTOMATED SWITCHING RULES AMONG THE CONSTITUENT OPTIMIZERS

A set of rules has been added to the hybrid constrained optimization system [22] in order to make the switching among the algorithms automatic, as well as to utilize some of the heuristic understanding of each algorithm's

behavior. The purpose of this switching was to increase the program's robustness and improve upon its convergence.

Each switching rule was based upon and incorporated with the unique behavior of each numerical optimization algorithm. The timing of the switching among the algorithms was forced to occur during those instances in which they performed badly, stalled, or failed. The first algorithm that the optimization process chose to switch to was determined by reasoning and by trial and error. If the subsequent algorithm also failed in its processes, the opportunity was made for every one of the algorithms in the system to have a try at the problem. When all the available algorithms had been tried on a particular population of feasible designs, and all failed at their given tasks, then the program was terminated.

The rules for switching will now be discussed for each algorithm in the hybrid optimization system. Figures 3, 4 and 5 demonstrate the major characteristics of this switching process in flowchart form.

Gradient based algorithms are the most time consuming of the optimization algorithms, but their convergence is both maximized and guaranteed, assuming the gradient calculation, $\partial F/\partial \vec{V}$, is accurate enough. These methods exhibit very rapid convergence early in the optimization process, especially when the initial guess starts with a poor design. But, the nature of their operation is based upon the local gradient. Therefore, they will always move in the direction that minimizes the function until they reach the minimum. The problem is that this minimum might not be the solution to the problem (global minimum), but only a local minimum. Another one of their limitations also comes from the fact that the gradient of the objective function tends to level off near a minimum. The DFP method provides for a modified second-order accurate convergence rate, so that the algorithm can home in on the local or global minimum a bit faster than the first order accurate steepest descent, but the convergence during these periods will be slow.

Local Minimum rule was developed for gradient-based methods. The gradient search optimization algorithm is switched whenever the change in the objective is less than a user-specified tolerance, $\overline{\Delta F}$.

$$\frac{\left|F^* - F_K\right|}{\left|F^* - F_0\right|} < \overline{\Delta F} \tag{11}$$

Here, F* is the value of the objective function at the new optimum, F_0 is the value of the objective function for the design at the start of the gradient search, and F_K is the value of the objective function from the previous optimization cycle. The genetic algorithm is the first algorithm chosen, then comes the simulated annealing and finally the Nelder-Mead algorithm (GA – SA - NM). The genetic algorithm is chosen first because its random search capabilities are useful in escaping local minima. This rule is also applicable whenever the program stalls on a constraint boundary.

Descent Direction rule is applied when the DFP search direction is not a descent direction, in which case the dot product of the search direction, $\vec{S}$, and the gradient of the objective function, ∇F, is greater than zero. Remember that the search direction may be different from the negative gradient direction, because of the DFP update formula and because search direction is projected onto the subspace of active constraints (Rosen's projection method).

$$\vec{S} \bullet \nabla F > 0 \tag{12}$$

When this condition is met, the inverse Hessian matrix is re-initialized to the identity matrix. If the inverse Hessian is already equal to the identity due to re-initialization, the program is automatically switched to simulated annealing (SA – NM - GA). The randomness and simplex searching methods of the simulated annealing process provide quick and effective ways of navigating through the irregular design spaces characteristic of the non-descent direction criterion.

New designs (optimum along the line search direction) created by the DFP are added to the population matrix and the DFP always works on the best member in the population matrix

The GA has several criteria that can qualify its performance and so several switching rules have been developed from these. The most often used switching criterion is based upon the variance in the population. As the genetic algorithm proceeds, it tends to select members in the population with like qualities to breed more often. The result is that the population tends to take on a similar set of characteristics and the variation in the population reduces. This can happen too quickly when the specified mutation rate is too infrequent. In the ideal situation, the design variables of the population will tend to collect around the global minimum, but may have difficulty in finding it.

Design Variance Limit rule was developed as the first rule for switching from GA.

$$\sigma_V = \frac{1}{N_{VAR} N_{POP}} \sum_{i=1}^{N_{VAR}} \frac{\sqrt{\sum_{j=1}^{N_{POP}} \left(P_{i,j} - \bar{V}_i\right)^2}}{\left(V_{\max,i} - V_{\min,i}\right)} < \sigma_{V\min} \tag{13}$$

In this equation, the non-dimensional standard deviation, σ_V, for all design variables in the population is measured with respect to the average design variable in the population.

$$\bar{V}_i = \frac{1}{N_{POP}} \sum_{j=1}^{N_{POP}} P_{i,j} \qquad i = 1,...,N_{VAR} \tag{14}$$

When the design variance in the population becomes less than the user specified limit, GA is switched to DFP. The reasoning is that the population variance is contracting around a minimum and the DFP can be used to quickly home in on that minimum. The order of the automatic switching is DFP – NM - SA.

Objective RMS Limit rule is similar to the aforementioned rule, except that the variance in the objective function is computed rather than the variance in the design variables.

$$\sigma_F = \frac{1}{N_{POP}} \frac{\sqrt{\sum_{j=1}^{N_{POP}} \left(F_j - \bar{F}\right)}}{\left(F^* - F^0\right)} \tag{15}$$

Here, the F_j is the objective function value of the jth population member, and the average function value of the population is computed from them.

$$\bar{F} = \frac{1}{N_{POP}} \sum_{j=1}^{N_{POP}} F_j \tag{16}$$

The difference between this rule and the design variance limit is that the population may be dispersed over a wide design space, but each may have very similar objective function values. This can occur is the objective function space is a large flat area with little or no gradient. The program is then switched first to the simulated annealing method (SA – NM – DFP).

Bad Mutation rule causes GA to switch to the Nelder-Mead algorithm if the average objective function increases from the previous optimization cycle that used GA. This will most likely occur if the random mutation rate is too large or if it produces one or more really bad designs. Since the Nelder-Mead algorithm specializes in bringing the worst members of the population closer to the centroid of the population, it is the most obvious first choice resulting in a sequence (NM – SA – DFP).

Lost Generation rule causes the hybrid optimizer to switch to the simulated annealing program (SA – NM – DFP). GA develops sequential populations of new 'childdren' that are entered into the population only if the population size is allowed to increase, or if the 'children' are better (have lower objective functions) than the worst (highest objective function) members in the population. If no new designs are entered into the population, the GA. This rule takes care of such situations.

Stalled Process rule is specifically developed for Nelder-Mead simplex searching algorithm that has only one failure mode. The NM is said to stall whenever the line search (produced by the direction from the worst design in the population through the centroid of the best designs) fails to improve itself so that it is better than the second worst member in the population.

$$F^*_{N_{POP}} \geq F_{N_{POP}-1} \tag{17}$$

This rule causes the program to switch to the DFP method (DFP – GA – SA). If the best design in the population was generated by the DFP, then the DFP is passed by and the GA takes over.

Population Less Fit rule has the purpose to assist the simulated annealing algorithm to add energy into the optimization system and allow it to escape local minima. The objective functions of the design variables in the population might get worse in this process. Therefore, there should be a limit set as to how much worse the population could get. That is, whenever the average objective function of the current population is less fit than some multiple of the average objective function of the previous optimization cycle's population, the program is switched.

$$\bar{F}^K > \left(1 + \varepsilon\right) \bar{F}^{K-1} \tag{18}$$

Insufficient Random Energy rule can be stated as follows.

$$\frac{\sqrt{\sum_{j=1}^{Npop} \left(\tilde{F}_j - \bar{F}\right)^2}}{N_{pop}\left(F^0 - F^*\right)} < \sigma_{F_{\min}} \tag{19}$$

Here, the algorithm ends whenever the variance of the objective function values with added random energy, $\tilde{F}$, are less than a user-specified limit. The program is switched from the simulated annealing to the DFP method (DFP – GA – NM). After several cycles, the random energy in the simulated annealing algorithm may have been reduced to a negligible amount, while the insufficient random energy criterion might not be met because of the large variance in the population. Therefore, the simulated annealing algorithm is switched to the Nelder-Mead (NM – GA – DFP) after $\tilde{K}_{\max}$ optimization cycles. The simulated annealing algorithm has, therefore, been programmed to end whenever the cooling protocol did not add sufficient energy into the system.

11. SEVENTH GENERATION OF SINGLE-OBJECTIVE HYBRID OPTIMIZATION ALGORITHMS WITH AUTOMATIC SEARCH VECTOR-BASED SWITCHING

The degree of success of any optimization process depends on the structure of the optimization algorithm, and the topology of the objective function. An optimization algorithm may be written such that it searches consistently in the direction of the current global best design vector (*e.g.,* DE best/2/bin [35]), but if the objective function topology is deceptive, it may direct the optimization algorithm into a local minimum. This rigidity in search strategy causes certain optimization algorithms to be better at optimizing some functions than others. The Search Vector-based hybrid (SVH) presented here [36] attempts to overcome this drawback by changing search directions during the optimization process.

It does so through the use of a predetermined set of search vectors (SV). Each iteration, the SVH will generate the SVs based on a predetermined formula or quality. The SVs are then evaluated. Some examples of the SVs formulations currently in use include:

1. Global Best Vector (GB): This is the fittest design vector currently in the population. It is implemented in the same way that the global best vector is implemented in PS [8] and DE best/2/bin [35].
2. Population Weighted Average (PWA): The population is ranked from best to worst, with the best receiving a rank equal to the population size, and the worst having a rank of one. The ranks are then used as weights, and the standard weighted arithmetic mean procedure is used to create this SV.

After the SVs have been evaluated, the fittest SV is selected as the SV for that iteration. At least one author has previously used the phrase "optimal search direction" [37], which is superficially similar to the concept of a "fittest SV," but the strategy presented here differs from any other known work in that it uses a collection of different search directions, each with their own unique formulation, and chooses between them. The method presented in [37], like most other algorithms, generates the search direction and keeps it fixed throughout the entire optimization process. The approach presented here will be called the "aggressive search strategy" since it assumes that the fittest SV must also be the best search direction, and selected for use.

The SVH automatically switches between a collection of optimization algorithms called constituent algorithms. Once the SV has been selected, the constituent algorithm selection process begins.

First, each constituent algorithm is executed so that it generates a temporary population. This temporary population will not be evaluated. Instead, it will be used as an indication of the behavior of the constituent algorithm for a given topology. For example, suppose the SVH has two constituent algorithms called CA1 and CA2 (see Figure 8).

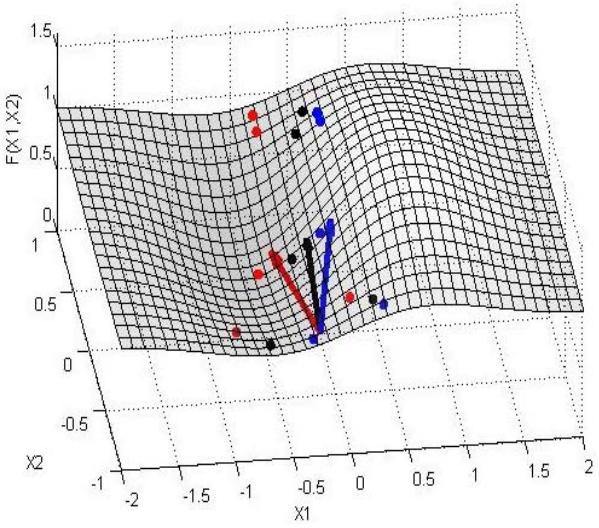


Figure 8: Example of population movements and their accompanying centroids using two fictitious constituent algorithms CA1 and CA2

It is possible that CA1 will use the current population (black dots) to generate a temporary population shifted to the left (red dots), while CA2 will create a temporary population shifted to the right (blue dots). The vectors in Figure 8 represent the centroids of each population. In order to select the constituent algorithm, the Euclidean distance between the endpoint of each centroid vector and the selected SV is calculated and stored. Secondly, each centroid is evaluated. The constituent algorithms are then ranked (using the Pareto dominance scheme in [38]) based on two objectives: (1) minimize distance between the centroid and the SV, and (2) minimum objective function value of the centroid. With this, a Pareto front can be created, and the constituent algorithm to be used is randomly selected from the Pareto front. In order for the centroids of the constituent algorithms to be statistically meaningful, the constituent algorithms are executed 10 times each iteration.

Once a constituent algorithm has been selected, it is then executed one last time. This time, the population is permanently changed, and the objective function for each design vector is evaluated. This completes one full iteration of the SVH.

The SVH can be summarized as follows:

1) Create an initial population.
2) Calculate fitness for each design vector.
3) Begin main loop:
   a. Calculate and evaluate SVs.
   b. Select fittest SV.
   c. Execute each constituent algorithm ten times and obtain a centroid of temporary populations for each constituent algorithm.
   d. Evaluate centroids of the constituent algorithms.
   e. Using (a) and (d), randomly select one of the Pareto optimal constituent algorithms (*i.e.*, minimum distance to SV and fittest centroid).
   f. Execute selected constituent algorithm normally.
4) End main loop once population converges or maximum number of iterations is reached.

Proper incorporation of constituent algorithms and maintenance of auxiliary vector populations such as the velocity of PS is not trivial. The SVH currently utilizes six constituent algorithms: PS [8], Particle Swarm with Random Differencing (PRD) [39], Modified Quantum Particle Swarm (MQP) [40], DE best/2/bin with randomized parameters (BST) [35], DE Donor3 with randomized parameters (DN3) [41], and Cuckoo Search (CKO) [42].

Algorithms like BST, DN3, and CKO do not utilize information from previous iterations and can be treated as independent modules. The PS [8] and PRD [39] constituent algorithms, however, utilize a velocity vector population that is a function of the previous iteration's velocity. As long as one of these modules is called sequentially, the velocity vector can be computed normally. However, if the SVH switches from PS or PRD to another method and then back to PS or PRD, the velocity calculation becomes meaningless. Therefore, the velocity must be reinitialized. As a first attempt, the velocity re-initialization equation can be used as

$$V_j = \left(D_{l,j} + R\left(D_{u,j} - D_{l,j}\right)\right)\frac{0.2}{1.02^G} \qquad (20)$$

where V_j is the j[th] coordinate of the velocity vector, $D_{l,j}$ and $D_{u,j}$ are the lower and upper limits of the search domain in the j[th] direction respectively, R is a uniformly distributed random number [0,1] and G is the generation (iteration) number.

Additionally, algorithms with DE-style comparisons have an implicit form of elitism. These comparisons prohibit the fitter design vectors like the GB from being replaced by inferior design vectors. This feature is not present in algorithms like PS. Therefore, if the SVH

switches from DE to PS and back to DE it is possible to lose fitter design vectors. In order to partially remedy this, the SVH stores the GB separately, but allows other vectors to be overwritten by constituent algorithms.

Since each constituent algorithm has its own logic, another issue inherent to this SVH is that the SV and constituent algorithm selection process overrides this logic. This is not necessarily a flaw, but it is an important consideration nonetheless. As suggested earlier, no optimization algorithm can solve all problems because no single search method is capable of optimizing all function topologies, which includes the presented SVH. Although automatically switching SVs partly resolves this issue, the strategy itself can still fall victim to this limitation.

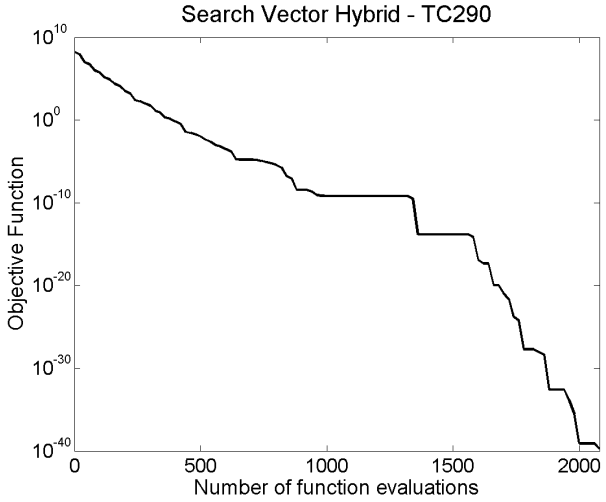
Therefore, a random parameter was introduced in the algorithm. Each iteration after the first, there is a 10% chance that the SVH will reselect the constituent algorithm used in the previous iteration. In this way, the SVH is not strictly bound to behave in a particular manner and can possibly escape the pitfalls of a strict search strategy. Another way to implement this feature could be to introduce a probability that the SVH will select a random constituent algorithm rather than the previous one, but this has not yet been tested.

Since the constituent algorithms are effectively modular, they can be added or removed as desired. SVs can also be added or removed as needed, thereby allowing the user of the SVH to incorporate the latest in optimization research into the SVH with minimal changes to the SVH code. Furthermore, it is widely known that changing the values of user-defined parameters can greatly impact the performance of an optimization algorithm. With this SVH, if a single optimization algorithm can be tailored to several classes of problems, then each tailored optimization algorithm can be incorporated as a distinct constituent algorithm. While increasing the number of constituent algorithms will increase the SVH's overhead computing costs, it can also increase its robustness. Since many real-world engineering problems have objective functions that require hours or even days to evaluate, the increased overhead of the SVH can be trivial by comparison.

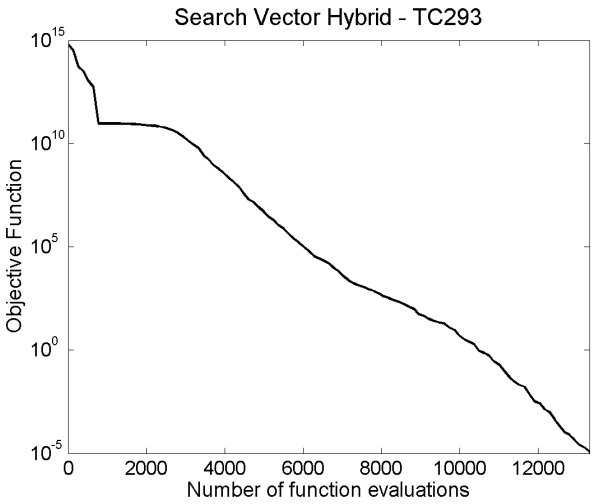
To benchmark this hybrid the Schittkowski & Hock test cases [43, 44] will be used. This set of analytical test functions contains over 300 test cases ranging from unconstrained, smooth and continuous objective functions to heavily constrained, discontinuous objective functions. For simplicity, constraints will be enforced using penalty functions. Test cases 290 – 293 use the same unconstrained function, given by the equation (21).

$$U(\vec{x}) = \left(\sum_{i=1}^{N} i x_i^2 \right)^2 \tag{21}$$

where N is the dimension of the function space. The dimension N for test cases 290, 291, 292, and 293 is 2, 10, 30, and 50, respectively, making them useful for examining the performance of an algorithm as the problem dimensionality increases.



a) 2 dimensional test case



b) 50 dimensional test case

Figure 9: SVH performance with increasing dimension

Figure 9, like most optimization algorithms, demonstrates that the hybrid optimizer requires more objective function evaluations with increased problem dimensionality. However, unlike many optimization algorithms, the increase in number of function evaluations needed to reach the same level of accuracy is not exponential. In order to obtain an accuracy on the order of 10^{-5} from the global minimum, the SVH requires roughly 760 evaluations for the 2-dimensional problem, and 13440 evaluations for the 50-dimensional problem. Therefore, although the dimensionality has increased by a factor of 25, the number of function evaluations increases by a factor of approximately 17.

Figure 10 demonstrates the performance of the SVH on the Griewank function [1] averaged over 50 optimization trials. In this case, the accuracy of the SVH is on par with DE, PSO, and SA, but slower than the hybrid of Figure 2e. This difference in performance can potentially be improved through the use of different SVs or different constitutive algorithms.

Overall, this hybrid optimizer outperforms its constitutive algorithms in accuracy and speed in over 60% of the Schittkowski and Hock test cases and performs competitively against other hybrid algorithms, including those previously presented.

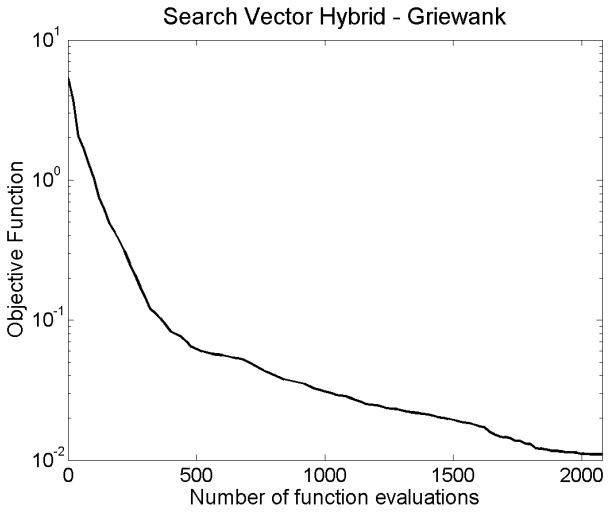


Figure 10: Convergence history of SVH optimizer applied to Griewank's function

CONCLUSION

A collection of standard gradient-based and evolutionary optimization algorithms was assembled in several generations of hybrid optimization tools where a set of heuristic rules was used to perform automatic switching among the individual optimizers in order to avoid local minima, escape from the local minima, converge on the global minimum, and reduce the overall computing time. The constraints were enforced either via penalty function or via Rosen's projection method. Lessons learned from these efforts are valuable. Most importantly, hybrid optimization is a very robust and cost-effective optimization concept. Automatic switching among the individual optimizers can be further improved by incorporating certain aspects of neural networks. Use of simplified models (surrogates) for evaluation of the objective function is highly cost effective, although progressively more complete physical models should be used as the global optimization process starts converging. Parameterization of the design space plays a crucial role in the hybrid constrained optimization.

ACKNOWLEDGMENTS

Authors are grateful for the pioneering work performed by Norman Foster in his M.Sc. Thesis in 1996 that lead to the first two generations of our hybrid single-objective constrained optimizers. Lead author dedicates this article to his alma mater – University of Belgrade.

REFERENCES

[1] GEATbx: Genetic and Evolutionary Algorithm Toolbox for use with MATLAB, Version 1.91, July 1997.

[2] 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, pp. 92-118, 2005.

[3] Colaço, M.J., Orlande, H.R.B. and Dulikravich, G.S.: Inverse and Optimization Problems in Heat Transfer, Journal of the Brazilian Society of Mechanical Science & Engineering, Vol. XXVIII, No. 1, pp. 1-23, January-March 2006.

[4] Colaço, J.M. and Dulikravich, G.S.: A Survey of Basic Deterministic, Heuristic and Hybrid Methods for Single-Objective Optimization and Response Surface Generation, Chapter 10 in Thermal Measurements and Inverse Techniques, (eds: Orlande, H.R.B., Fudym, O., Maillet, D. and Cotta, R.), Taylor & Francis, pp. 355-405, May 2011.

[5] Broyden, C.G.: Quasi-Newton Methods and Their Applications to Function Minimization, Math. Comp., Vol. 21, pp. 368-380, 1987.

[6] Storn, R. and Price, K.V.: Minimizing the real function of the ICEC'96 contest by differential evolution, Proc. of IEEE Conference on Evolutionary Computation, pp. 842-844, 1996.

[7] Corana, A., Marchesi, M., Martini, C. and Ridella, S.: Minimizing Multimodal Functions of Continuous Variables with the 'Simulated Annealing Algorithm', ACM Transactions on Mathematical Software, Vol. 13, pp. 262-280, 1987.

[8] Kennedy, J. and Eberhart, R.C.: Particle Swarm Optimization, Proc. of the 1995 IEEE International Conf. on Neural Networks, Vol. 4, pp. 1942-1948, 1995.

[9] Wolpert, D.H. and Macready, W.G.: No free lunch theorems for optimization, Evolutionary Computation, IEEE Transactions on Evolutionary Computing, Vol. 1, No. 1, pp. 67-82, 1997.

[10] Foster, N.F.: Shape Optimization Using Genetic Evolution and Gradient Search Constrained Algorithms, M.Sc. Thesis, Department of Aerospace Eng., The Pennsylvania State University, State College, PA, USA, August 1995.

[11] Foster, N.F. and Dulikravich, G.S.: Three-Dimensional Aerodynamic Shape Optimization Using Genetic and Gradient Search Algorithms, AIAA Journal of Spacecraft and Rockets, Vol. 34, No. 1, pp. 36-42, 1997.

[12] Dulikravich, G. S.: Design and Optimization Tools Development, chapters no. 10-15 in New Design Concepts for High Speed Air Transport, (eds.: H. Sobieczky), Springer, Wien/New York, pp. 159-236, 1997.

[13] Martin, T.J., Dennis, B.H. and Dulikravich, G.S.: Aero-Thermal-Structural Optimization, NASA / ICOMP 1996 Coolant Flow Management Workshop, Cleveland, OH, USA (eds: S. Hippensteele and J. Gladden), NASA Conference CP 10195, pp. 311-334, December 12-13, 1996.

[14] Davidon, W.C.: Variable Metric Method for Minimization, Atomic Energy Commission Research and Development Report, ANL-5990 (Rev.), November 1959.

[15] Fletcher, R. and Powell, M.J.D.: A Rapidly Convergent Descent Method for Minimization, Computer Journal, Vol. 6, pp. 163-168, 1963.

[16] Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, 1989.

[17] Martin, T.J. and Dulikravich, G.S.: Aero-Thermal Analysis and Optimization of Internally Cooled Turbine Blades, XIII International Symposium on Airbreathing Engines (XIII ISABE), (eds.: F.S. Billig) Chattanooga, TN, USA, ISABE 97-7165, Vol. 2, pp. 1232-1250, September 8-12, 1997.

[18] Dennis, B.H. and Dulikravich, G.S.: Thermo-Elastic Analysis and Optimization Environment for Internally Cooled Turbine Airfoils, XIII International Symposium on Airbreathing Engines (XIII ISABE), (eds.: F.S. Billig), Chattanooga, TN, USA, ISABE 97-7181, Vol. 2, pp. 1335-1341, Sept. 8-12, 1997.

[19] Martin, T.J., Dulikravich, G.S. and Han, Z.X.: Aero-Thermal Optimization of Internally Cooled Turbine Blades, Minisymposium on Multidisciplinary-Optimum Design, Fourth ECCOMAS Computational Fluid Dynamics Conference (eds.: Editors: K. Papailiou, D. Tsahalis, J. Periaux, D. Knoerzer), John Wiley & Sons, New York, Vol. 2, pp.158-161, Athens, Greece, September 7-11, 1998.

[20] Martin, T.J., Dennis, B.H., Dulikravich, G.S., Lee, S.S. and Han, Z.X.: Aero-Thermo-Structural Design Optimization of Internally Cooled Turbine Blades, AGARD - AVT Propulsion and Power Systems Symposium on Design Principles and Methods for Aircraft Gas Turbine Engines, NATO-RTO-MP-8 AC/323(AVT)TP/9, Ch. 35, Toulouse, France, May 11-15, 1998.

[21] Petrovic, M.V., Martin, T.J. and Dulikravich, G.S.: Axial Gas Turbine Efficiency Over a Range of Operating Conditions, Proceedings of 10TH Thermal & Fluids Analysis Workshop (TFAWS'99) (eds.: L.W. Griffin), NASA Marshall Space Flight Center, Huntsville, AL, USA, Sept. 13-17, 1999.

[22] Dulikravich, G.S., Martin, T.J., Dennis, B.H. and Foster, N.F.: Multidisciplinary Hybrid Constrained GA Optimization, a chapter 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), Jyvaskyla, Finland, John Wiley & Sons, pp. 233-259, May 30 - June 3, 1999.

[23] Nelder, J.A. and Mead, R.: A Simplex Method for Function Minimization, Computer Journal, Vol. **7**, pp. 308-313, 1965.

[24] Dulikravich, G.S., Dennis, B.H., Martin, T.J. and Egorov, I.N.: Multi-disciplinary Analysis and Design Optimization, Invited Lecture, Mini-Symposium on Inverse Problems - State of the Art and Future Trends, XXIV Brazilian Congress on Applied and Computational Mathematics, Sept. 10-13, 2001, Belo Horizonte, Brazil.

[25] Dulikravich, G.S., Dennis, B.H., Martin, T.J. and Egorov, I.N.: Multi-disciplinary Design Optimization, Invited Lecture, EUROGEN 2001 - Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems, (ed.: Giannakoglou, K., Tsahalis, D.T., Periaux, J. and Fogarty, T.), Athens, Greece, Sept. 19-21, 2001, Published by International Center for Numerical Methods in Engineering (CIMNE), Barcelona, Spain, pp. 11-18, 2002.

[26] Zhou, J.L. and Tits, A.: User's Guide for FFSQP Version 3.7: A Fortran Code for Solving Optimization Programs, Possibly Minimax, with General Inequality Constraints and Linear Equality Constraints, Generating Feasible Iterates, Institute for Systems Research, University of Maryland, Tech. Report SRC-TR-92-107r5, 1997.

[27] Pshenichny, B.N. and Danilin, Y.M.: Numerical Methods in Extremal Problems, MIR Publishers, Moscow, 1969.

[28] Hafka, R.T. and Gurdal, Z.: Elements of Structural Optimization, 3rd edition, Kluwer Academic Publ., Boston, MA, USA, 1992.

[29] Hafka, R.T. and Malkus, D.S.: Calculation of Sensitivity Derivatives in Thermal Problems by Finite Differences," Int. Journal of Numerical Methods in Engineering., Vol. 17, pp. 1811-21, 1981.

[30] Colaco, M.J. and Dulikravich, G.S.: A Multilevel Hybrid Optimization of Magnetohydrodynamic Problems in Double-Diffusive Fluid Flow, Journal of Physics and Chemistry of Solids, Vol. 67, pp. 1965-1972, 2006.

[31] Colaco, M.J. and Dulikravich, G.S.: Solidification of Double-Diffusive Flows Using Thermo-Magneto-Hydrodynamics and Optimization, Materials and Manufacturing Processes, Vol. 22, pp. 594-606, 2007.

[32] Sobol, I. and Levitan, Y.L.: The Production of Points Uniformly Distributed in a multidimensional Cube, Preprint IPM Akademia Nauk SSSR, No. 40, Moscow, Russia, 1976.

[33] More, J.J., Gabow, B.S. and Hillstrom, K.E.: Testing Unconstrained Optimization Software. ACM AQ13, Transactions on Mathematical Software, pp. 17–41, 1981.

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

[35] Inclan, E.J. and Dulikravich, G.S.: Effective Modifications to Differential Evolution Optimization Algorithm, International Conference on Computational Methods for Coupled Problem in Science and Engineering, (eds.: Idelsohn, S., Papadrakakis, M., Schrefler, B.), Santa Eulalia, Ibiza, Spain, June 17-19, 2013.

[36] Inclan, E.J. and Dulikravich, G.S.: A Hybrid Optimization Algorithm with Search Vector Based Automatic Switching, World Congress of Multidisciplinary Optimization, (eds.: Kim, N.-H., Haftka, R.), Orlando, FL, USA, May 15-19, 2013.

[37] Ahrari, A., Shariat-Panahi, M. and Atai, A.A.: GEM: A Novel Evolutionary Optimization Method with Improved Neighborhood Search. Applied Mathematics and Computation, Vol. 210, no. 2, pp. 376–386, 2009.

[38] Deb, K.: Multi-Objective Optimization using Evolutionary Algorithms. Chichester: Wiley & Sons, 2009.

[39] Inclan, E.J., Dulikravich, G.S. and Yang, X.-S.: Modern Optimization Algorithms and Particle Swarm Variations, Symposium on Inverse Problems, Design and Optimization-IPDO2013, (eds.: Fudym, O., Battaglia, J.-L.), Albi, France, June 26-28, 2013.

[40] Sun, J., Lai, C.H., Xu, W., and Chai, Z.: A Novel and More Efficient Search Strategy of Quantum-Behaved Particle Swarm Optimization. ICANNGA '07 Proceedings of the 8th international conference on Adaptive and Natural Computing Algorithms, pp. 394 – 403, 2007.

[41] Fan, H.-Y., Lampinen, J. and Dulikravich, G. S.: Improvements to Mutation Donor of Differential Evolution. EUROGEN2003 - International Congress on Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems, (eds: G. Bugeda, J. A-Désidéri, J. Periaux, M. Schoenauer and G. Winter), CIMNE, Barcelona, Spain, September 15-17, 2003.

[42] Yang, X.-S. and Deb, S.: Engineering Optimisation by Cuckoo Search. International Journal Mathematical Modelling and Numerical Optimisation, Vol. 1, No. 4, pp. 330-343, 2010.

[43] Schittkowski, K.: More Test Examples for Nonlinear Programming Codes. Berlin: Springer-Verlag, 1987.

[44] Schittkowski, K. and Hock, W.: Test Examples for Nonlinear Programming Codes - All Problems from the Hock-Schittkowski-Collection. Bayeruth: Springer, 1981.

АЛГОРИТМИ ЗА АУТОМАТСКО ПРЕБАЦИВАЊЕ УНУТАР ХИБРИДНИХ ЈЕДНО-ЦИЉНИХ ОПТИМИЗАТОРА

George S. Dulikravich, Thomas J. Martin, Marcelo J. Colaco, Eric J. Inclan

Хибридни алгоритми за оптимизацију се састоје од неколико проверених основних оптимизационих алгоритама и једног контролног алгоритма који извршава аутоматско пребацивање између тих основних алгоритама у свакој фази оптимизације када брзина конвергенције постаје незадовољавајућа, када процес почиње да конвергира према локалном минимуму, или кад се деси неки други нежељени аспект итеративног процеса. Према томе, хибридни оптимизациони алгоритми који употребљавају неколико основних оптимизационих алгоритама који су базирани на градијенту и оних који нису базирани на градијенту су стабилнији и конвергирају брже него појединачни основни оптимизациони алгоритми. У овом раду је приказана логика коју треба применити за стварање алгоритама за аутоматско пребацивање у хибридни оптимизациони алгоритам са циљем да се прикажу истраживања које су аутори учинили у области хибридних једно-циљних оптимизатора почевши од 1997.