Improved Global Convergence Probability Using Independent Swarms

Jaco F. Schutte* and Raphael T. Haftka†
University of Florida, Gainesville, Florida, 32611-6250

In global optimization it may sometimes be more efficient to perform multiple independent optimization runs with a limited number of fitness evaluations in place of a single run with an equal number of total fitness evaluations. This is especially true for problems with a number of widely separated local optima in the design space because a single optimization has a larger probability of becoming entrapped in a local optimum than multiple independent runs. This approach can be further exploited by utilizing parallel computation. Some preliminary results of an investigation on the feasibility of multiple independent optimizations are reported.

I. Nomenclature

\[ P_i \] = individual optimization global convergence probability
\[ P_c \] = multiple optimization cumulative global convergence probability
\[ C_r \] = convergence ratio
\[ N \] = number of optimization runs
\[ N_c \] = number of globally converged optimization runs
\[ n_{fe} \] = number of fitness evaluations
\[ n_b \] = budget of fitness evaluations allocated to solving problem
\[ n_i \] = allowed fitness evaluations for each independent optimization
\[ n_l \] = number of fitness evaluations required by optimization algorithm to find minima
\[ s_e \] = standard error

II. Introduction

GLOBAL optimization algorithms, no matter how sophisticated, may become entrapped in a local minimum when attempting to solve challenging large-scale optimization problems. When applying population based optimizers such as Genetic algorithms or the Particle Swarm algorithm the usual steps taken to improve the probability of convergence to the global optimum are one of the following: Increasing the population size – this allows for a higher sampling density per generation or iteration in the design space which increases the probability of sampling near the global optimum. Extending the number of fitness evaluations – running the optimization for a higher number of generations or iterations gives the algorithm more opportunity to explore the design space. Finally, the global algorithm parameters may be fine tuned to a specific problem by repeating an optimization with different parameter sets. All these strategies require substantial additional fitness evaluations, and may only have limited success. This paper explores the possibility of using multiple runs to improve the probability of reaching the global optimum.

* Graduate Research Assistant, Department of Mechanical and Aerospace Engineering, MAE-A, AIAA Student Member.
† Distinguished Professor, of Mechanical and Aerospace Engineering, MAE-A, AIAA Fellow
III. Multiple Run Strategy

We follow an approach proposed by Le Riche and Haftka for use with genetic algorithms [1]. It entails running multiple optimizations with a reduced number of fitness evaluations, either by limiting the number of generations or reducing the population while keeping the number of iterations constant. Individually, the convergence probability of such a limited optimization may only be a fraction of a single full optimization run. However, the cumulative convergence probability obtained by combining the limited runs can be significantly higher than that of the single run.

The optimizer applied in this study is the Particle Swarm Optimizer. It has strong global search capabilities and, being a stochastic population based optimizer, is an ideal candidate for parallelization [2]. The multiple run strategy can however be applied with any global or local optimization algorithm.

IV. Effect of Population Size and Allowed Iterations

The impact of population size on the performance of the PSO algorithm was previously investigated by Carlisle and Dozier [3] and Shi and Eberhart [4], among others. Carlisle and Dozier showed that, for the set of test problems evaluated in their paper, a certain minimum threshold population size is required for reliable convergence. They also showed that increasing the swarm size excessively beyond this value results in a decrease in algorithm efficiency.

We investigate these properties further by optimizing a set of test problems with different population sizes for a very high number of function evaluations.

V. Methodology

A. Analytical test set

The convergence behavior of the PSO algorithm was analyzed with the Griewank [6], Shekel [8] and Hartman [8] set of analytical problems (see Appendix). To obtain reasonably accurate convergence ratio/rate information, 1000 optimization runs are performed for each problem, with each run limited to a very high 500000 fitness evaluations. This was done to verify if an upper limit in the convergence probability exists for each problem. These optimization runs were performed with identical parameter settings, with the exception of a different random number seed for each optimization run to start the population in a different initial configuration. The 1000 optimizations are repeated for population sizes of 10, 20, 50 and 100 particles in the swarm. It is assumed that convergence to the global optimum is achieved when the fitness value approaches to within a predetermined fixed tolerance [Table 1] of the known optimum. For the Shekel and Hartman problems the selected tolerance values ensure that the global optimum for the problem is found. For the Griewank problem this is however not the case, since this noisy function

Figure 1. Multiple local minima for Griewank analytical problem surface plot in 2 dimensions
has several local minima for very small tolerances. An arbitrary value tolerance was selected which made the problem moderately difficult for the optimizer. A convergence ratio, \( C_r \), is then calculated as follows:

\[
C_r = \frac{N_c}{N}
\]  

where \( N_c \) is the number of globally converged optimizations, and \( N \) is the total number of optimizations, in this case 1000. For a large number of total optimizations, as in this case, we can assume (by the law of large numbers) that the probability of convergence to the global optimum for an individual run is \( P_i = C_r \) with limited accuracy.

The accuracy can be quantified using the standard error, (which is an estimate of the standard deviation of \( C_r \))

\[
s_e = \sqrt{\frac{P_i(1-P_i)}{N}}
\]

For example, if we obtain a convergence probability of \( P_i = 0.5 \) for example, with \( N = 1000 \) optimizations, the standard error would be \( s_e = 0.016 \).

<table>
<thead>
<tr>
<th>Table 1. Problem convergence tolerances</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem</td>
</tr>
<tr>
<td>--------------</td>
</tr>
<tr>
<td>Griewank</td>
</tr>
<tr>
<td>Shekel</td>
</tr>
<tr>
<td>Hartman</td>
</tr>
</tbody>
</table>

Figure 2 shows a sample of 20 fitness histories for the Griewank problem with a population size of 20 particles. It can be seen that after about 20,000 to 40,000 fitness evaluations most of the optimizations have reached a steady value, with only few optimizations jumping to lower values at higher numbers of fitness evaluations. The jump to a lower fitness value indicates an optimization which was able to escape from a local minimum. In Figure 3 we observe that the PSO require a high number of particles and fitness evaluations to obtain convergence ratios close to unity for the Griewank problem. This is consistent with the findings of [3]. Similar results were obtained for the Shekel and Hartman problem cases (omitted for the sake of brevity).

![Griewank G2, 20 particles](image-url)

Figure 2. Typical Griewank fitness history plots of 20 optimizations (sampled out of 1000)
B. Multiple-run convergence probability

If we denote the probability of a single individual optimization run converging to the global optimum as $P_i$, and the combined or cumulative probability of $N$ multiple independent optimization runs converging to the solution as $P_c$, using the fact that the convergence events are uncorrelated

$$P_c = 1 - \prod_{i=1}^{N} P_i$$

If we assume that individual optimization runs with similar parameter settings, as in the case of the following study, have an equal probability of convergence $P_i$, we can simplify Eq. (3) to

$$P_c = 1 - (1 - P_i)^N$$

The increase in cumulative probability $P_c$ with fixed values of $P_i$ for increasing number of optimization runs $N$ is illustrated in Figure 4. It must be stressed that the above relations are only valid for uncorrelated optimizations. This may not be true when a poor quality random number generator is used to generate initial positions in the design space, because certain generators exhibit a tendency to favor isolated regions in the design space, increasing the probability that local minima in only these regions may be found.

Figure 3. Griewank convergence ratio for 1000 optimizations as a function of fitness evaluations

![Figure 3](image-url)
C. Parallel Processing

An additional benefit to this strategy is the ability to exploit modern parallel processing resources. Although this approach does not require the runs to be executed in a parallel fashion the time requirements can be reduced to that of the longest single run utilizing a multiprocessor machine such as a Beowulf cluster.

D. Example

From Figure 3 it is clear that using a particle swarm with larger swarm sizes and/or allowing an increased number of fitness evaluations will deliver consistently higher convergence probabilities. On the other hand it can also be seen that minimizations with a small number of particles reach moderate convergence probabilities at significantly less fitness evaluations than large swarms. This is observed for all the problems in the test set. To exploit this behavior we propose replacing a single optimization with several PSO runs, each with a limited population and number of iterations. These individual optimizations utilize the same amount of resources allocated to the original single optimization (in this case the number of fitness evaluations).

Taking the Hartman problem as an example to illustrate the merit of such an approach we observe that the probability of convergence can only be slightly improved by allowing more fitness evaluations, or by increasing the population size (Figure 5). We also observe that an optimization with 10 particles quickly attains a probability of convergence $P_i = 0.344$ after 10,000 fitness evaluations. Using a multiple run strategy with limited optimizations of 10,000 fitness evaluations performed 10 times yield the theoretical $P_c$ values reported in Table 2 (calculated using Eq. [4] with $P_i = 0.344$ and $n = 1,\ldots,10$). These values are indicated as circled data points in Figure 5. The cumulative convergence probability obtained in this way is superior to that of using a single optimization run of up to 100 particles indefinitely.
To verify the cumulative probability values predicted with Eq. 3, a Monte Carlo approach is followed, sampling random pairs, triplets, quintuplets etc. of optimizations in the pool of 1000 runs. For example, to estimate the experimental global convergence probability of two runs we select a large number of random pairs among the 1000 runs. Applying Eq. (1), the number of cases in which either or both of the pairs converged $n_c$ for the limited optimization run, divided by $n$ (total number of pairs selected) will yield the experimental global convergence probability. These sampled convergence ratios are indicated as triangles in Figure 6 to Figure 9 along with the predicted values indicated by circles (as calculated using Eq. (4)). It can be seen that the points sampled by the Monte Carlo scheme reflect the theoretically predicted values closely.

In order to investigate the most efficient manner in which to divide a budget of fitness evaluations (which would have been used up by a single optimization) between multiple independent optimizations we execute sets of 2, 5, 10 and 12 independent optimizations, with a budget of 200,000 fitness evaluations divided equally among them. This

**Table 2. Convergence probability results for Hartman problem**

<table>
<thead>
<tr>
<th>Number of runs</th>
<th>Cumulative convergence probability $P_c$</th>
<th>Cumulative fitness evaluations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.344</td>
<td>10000</td>
</tr>
<tr>
<td>2</td>
<td>0.570</td>
<td>20000</td>
</tr>
<tr>
<td>3</td>
<td>0.718</td>
<td>30000</td>
</tr>
<tr>
<td>4</td>
<td>0.815</td>
<td>40000</td>
</tr>
<tr>
<td>5</td>
<td>0.879</td>
<td>50000</td>
</tr>
<tr>
<td>6</td>
<td>0.920</td>
<td>60000</td>
</tr>
<tr>
<td>7</td>
<td>0.948</td>
<td>70000</td>
</tr>
<tr>
<td>8</td>
<td>0.966</td>
<td>80000</td>
</tr>
<tr>
<td>9</td>
<td>0.978</td>
<td>90000</td>
</tr>
<tr>
<td>10</td>
<td>0.985</td>
<td>100000</td>
</tr>
</tbody>
</table>

**Figure 5. Cumulative convergence probability $P_c$ (Eq. (4)) as a function of the number of optimization runs with constant $P_i$ for the Hartman problem. Multiple independent runs with 10 particles.**

**VI. Numerical Results**

To verify the cumulative probability values predicted with Eq. 3, a Monte Carlo approach is followed, sampling random pairs, triplets, quintuplets etc. of optimizations in the pool of 1000 runs. For example, to estimate the experimental global convergence probability of two runs we select a large number of random pairs among the 1000 runs. Applying Eq. (1) the number of cases in which either or both of the pairs converged $n_c$ for the limited optimization run, divided by $n$ (total number of pairs selected) will yield the experimental global convergence probability. These sampled convergence ratios are indicated as triangles in Figure 6 to Figure 9 along with the predicted values indicated by circles (as calculated using Eq. (4)). It can be seen that the points sampled by the Monte Carlo scheme reflect the theoretically predicted values closely.

In order to investigate the most efficient manner in which to divide a budget of fitness evaluations (which would have been used up by a single optimization) between multiple independent optimizations we execute sets of 2, 5, 10 and 12 independent optimizations, with a budget of 200,000 fitness evaluations divided equally among them. This
results in each optimization in the set being stopped at 100,000, 40,000, 20,000 and 16,500 fitness evaluations. The results of these sets of multiple optimizations are also given in Figure 5 to Figure 9.

![Graph showing convergence ratio vs. fitness evaluations for 2 independent optimization runs for the Griewank problem. Each optimization run is limited to 100,000 fitness evaluations with 20 particles.](image1)

**Figure 6.** Extrapolated (Eq. (4)) and actual convergence probability $P_c$ for 2 independent optimization runs for the Griewank problem. Each optimization run is limited to 100,000 fitness evaluations with 20 particles.

![Graph showing convergence ratio vs. fitness evaluations for 5 independent optimization runs for the Griewank problem. Each optimization run is limited to 40,000 fitness evaluations with 20 particles.](image2)

**Figure 7.** Extrapolated (Eq. (4)) and actual convergence probability $P_c$ for 5 independent optimization runs for the Griewank problem. Each optimization run is limited to 40,000 fitness evaluations with 20 particles.
It can be seen that using a combination of runs at very high $P_c$ values (with a high number of the associated of fitness evaluations), Figure 6, or multiple runs with low $P_c$ values will not yield the same efficiency (as defined by $P_c$ per fitness evaluation). If the actual $P_c$ values are plotted on the same graph (Figure 10) it can be seen that the two combinations of 5 and 10 optimizations within the allocated budget yields the highest $P_c$ values for a given number of fitness evaluations.

Figure 8. Extrapolated (Eq. (4)) and actual convergence probability $P_c$ for 10 independent optimization runs for the Griewank problem. Each optimization run is limited to 20,000 fitness evaluations with 20 particles.

Figure 9. Extrapolated (Eq. (4)) and actual convergence probability $P_c$ for 12 independent optimization runs for the Griewank problem. Each optimization run is limited to 16,000 fitness evaluations with 20 particles.
E. Correlation between convergence probability and fitness history

Without performing a substantial number of optimizations (at significant computational expense) the exact individual convergence probability $P_i$ cannot be accurately determined. Our interest is finding the number of fitness evaluations where $P_i$ starts leveling off. The results in section D show that limiting the independent optimizations to this specific number yields the most efficient increase in $P_c$ value for a set of multiple optimizations. A way of obtaining this point where the curves flattens is by the observation that if the optimization fitness histories and the convergence probability $P_i$ are plotted for identical number of fitness evaluation scales, both types of graphs start leveling off at similar abscissa values. This trend is visible for very few optimizations, allowing us to obtain this information at little computational expense.
Ideally, as shown in section E, for the greatest efficiency independent optimizations in this strategy should be stopped at a number of fitness evaluations $n_i$ where the convergence ratio/probability graph starts leveling off (Figure 8). The exact convergence probability data is not available unless a substantial number of optimizations are performed, and must be estimated. This may be done by examining the fitness history of an optimization of the problem, since a correlation between the point where the fitness history levels off and the convergence history levels off (Figure 11) exists. For a global algorithm such as the PSO we propose using 10% of the total budget of fitness evaluations to obtain a fitness history.

As an example, a budget of $2 \times 10^5$ fitness evaluations will allow for $2 \times 10^4$ evaluations to be used to explore the problem/algorithm interaction. In most cases a single run will suffice for exploration. It is important to realize that the final fitness value of the exploratory run is inconsequential at this stage, we are only interested in the number of fitness evaluations required for the algorithm to converge to an optima (be it local or global). Any jumps out of local minima after the algorithm fitness history levels off, as observed for the Shekel problem in Figure 11, are disregarded. The remainder of the fitness evaluations is then distributed among a number of $N$ independent optimizations, which may be calculated as follows

$$N = \left\lfloor \frac{0.9n_b}{n_i} \right\rfloor \quad (5)$$

with an allowed number of fitness evaluations per run calculate by

$$n_i = \left\lfloor \frac{0.9n_b}{N} \right\rfloor, \quad n_i \geq n_i \quad (6)$$

If this methodology is applied to the Shekel problem with a budget of $2 \times 10^5$ fitness evaluations we observe from a single optimization fitness history (with $2 \times 10^4$ evaluations, Figure 12) that the algorithm reaches a minimum at $n_i = 10,000$ evaluations. If we apply Eq. (5) and (6) we obtain $N = 18$, and $n_i = 10,000$. 

Figure 11. Fitness history and convergence probability $P_c$ plots for Griewank, Hartman and Shekel problems.

VII. Discussion

Ideally, as shown in section E for the greatest efficiency independent optimizations in this strategy should be stopped at a number of fitness evaluations $n_i$ where the convergence ratio/probability graph starts leveling off (Figure 8). The exact convergence probability data is not available unless a substantial number of optimizations are performed, and must be estimated. This may be done by examining the fitness history of an optimization of the problem, since a correlation between the point where the fitness history levels off and the convergence history levels off (Figure 11) exists. For a global algorithm such as the PSO we propose using 10% of the total budget of fitness evaluations to obtain a fitness history.

As an example, a budget of $2 \times 10^5$ fitness evaluations will allow for $2 \times 10^4$ evaluations to be used to explore the problem/algorithm interaction. In most cases a single run will suffice for exploration. It is important to realize that the final fitness value of the exploratory run is inconsequential at this stage, we are only interested in the number of fitness evaluations required for the algorithm to converge to an optima (be it local or global). Any jumps out of local minima after the algorithm fitness history levels off, as observed for the Shekel problem in Figure 11, are disregarded. The remainder of the fitness evaluations is then distributed among a number of $N$ independent optimizations, which may be calculated as follows

$$N = \left\lfloor \frac{0.9n_b}{n_i} \right\rfloor \quad (5)$$

with an allowed number of fitness evaluations per run calculate by

$$n_i = \left\lfloor \frac{0.9n_b}{N} \right\rfloor, \quad n_i \geq n_i \quad (6)$$

If this methodology is applied to the Shekel problem with a budget of $2 \times 10^5$ fitness evaluations we observe from a single optimization fitness history (with $2 \times 10^4$ evaluations, Figure 12) that the algorithm reaches a minimum at $n_i = 10,000$ evaluations. If we apply Eq. (5) and (6) we obtain $N = 18$, and $n_i = 10,000$. 

Figure 11. Fitness history and convergence probability $P_c$ plots for Griewank, Hartman and Shekel problems.
Using 18 multiple optimizations of the Shekel problem, each limited to 10,000 fitness evaluations yield the cumulative probably $P_c$ given in Figure 13. If the initial exploratory run of 20,000 fitness evaluations is included in the Monte Carlo sampling for calculating the convergence probability, the circled data points are obtained, if this run is disregarded the triangle data points are obtained. Both types of curves reflect the true amount of fitness evaluations (including the exploratory phase) used for obtaining the corresponding data points.

If parallel computational resources are available, and each limited optimization is assigned to a separate node the multiple run approach will be constrained differently. Rather than the number of multiple optimizations being...
limited by a fixed budget of fitness evaluations (which is divided equally among the set of multiple independent optimizations using Eq. [6]), the budget will be defined by the number of computational nodes available to the user. This implies that, if a great number of nodes are available, very high $P_c$ values may be reached, even if severely limited multiple optimizations are used. Optimization time requirements for very complex and demanding large scale problems may be vastly reduced in this manner.

VIII. Conclusions

For the set of large scale optimization problems evaluated with the PSO, the multi-run strategy with small PSO populations delivers higher global convergence probability than a single run with a large population and an equal number of fitness evaluations. Using 10 percent of an allocated budget of fitness evaluations to evaluate the optimizer/problem behavior allows for the most efficient distribution of the remainder of the budget among the multiple independent optimizations. This multiple run strategy also allows for extremely efficient global optimization when parallel resources are available, an option which will be investigated in a future study.

IX. Appendix

1. **Griewank [7]:**
   Objective function:
   \[
   f(x) = \sum_{i=1}^{n} \frac{x_i^2}{d} - \prod_{i=1}^{n} \frac{x_i}{\sqrt{i}} + 1
   \]  
   with $n = 10$ and $d = 4000$
   Search domain $D = \{(x_1, x_2, \ldots, x_{10}) \in R^{10} : -600 \leq x_i \leq 600, i = 1, 2, \ldots 10\}$
   Solution: $x^* = (0.0, 0.0, \ldots, 0.0), f^* = 0.0$

2. **Shekel [8]:**
   Objective function:
   \[
   f(x) = \sum_{i=1}^{m} \frac{1}{(x-a_i)^T (x-a_i)} + c_i
   \]  
   \[
   \begin{array}{cccc}
   m & a_i & c_i \\
   1 & 4.0 & 4.0 & 4.0 & 4.0 & 0.1 \\
   2 & 1.0 & 1.0 & 1.0 & 1.0 & 0.2 \\
   3 & 8.0 & 8.0 & 8.0 & 8.0 & 0.2 \\
   4 & 6.0 & 6.0 & 6.0 & 6.0 & 0.4 \\
   5 & 3.0 & 7.0 & 3.0 & 7.0 & 0.4 \\
   6 & 2.0 & 9.0 & 2.0 & 9.0 & 0.6 \\
   7 & 5.0 & 5.0 & 3.0 & 3.0 & 0.3 \\
   8 & 8.0 & 1.0 & 8.0 & 1.0 & 0.7 \\
   9 & 6.0 & 2.0 & 6.0 & 2.0 & 0.5 \\
   10 & 7.0 & 3.6 & 7.0 & 3.6 & 0.5 \\
   \end{array}
   \]
   Search domain $D = \{x_i \in R^4 : 0 \leq x_i \leq 10, i = 1, \ldots, 4\}$
   Solution (with $m = 10$): $x^* = (4.00074671, 4.00059326, 3.99966290, 3.99950981), f^* = -10.536410$

3. **Hartman [8]:**
   Objective function:
   \[
   f(x) = \sum_{i=1}^{m} c_i \exp \left( -\sum_{j=1}^{n} a_{ij} \left( x_j - p_{ij} \right)^2 \right) \]  
   \[
   \begin{array}{cccc}
   m & a_{ij} & p_{ij} \\
   1 & 4.0 & 4.0 & 4.0 & 4.0 & 0.1 \\
   2 & 1.0 & 1.0 & 1.0 & 1.0 & 0.2 \\
   3 & 8.0 & 8.0 & 8.0 & 8.0 & 0.2 \\
   4 & 6.0 & 6.0 & 6.0 & 6.0 & 0.4 \\
   5 & 3.0 & 7.0 & 3.0 & 7.0 & 0.4 \\
   6 & 2.0 & 9.0 & 2.0 & 9.0 & 0.6 \\
   7 & 5.0 & 5.0 & 3.0 & 3.0 & 0.3 \\
   8 & 8.0 & 1.0 & 8.0 & 1.0 & 0.7 \\
   9 & 6.0 & 2.0 & 6.0 & 2.0 & 0.5 \\
   10 & 7.0 & 3.6 & 7.0 & 3.6 & 0.5 \\
   \end{array}
   \]
Search domain \( D = \{ (x_1, \ldots, x_6) \in \mathbb{R}^6 : 0 \leq x_i \leq 1, i = 1, \ldots, 6 \} \)

Solution (with \( m = 10 \)): \( \mathbf{x}' = (0.2017, 0.1500, 0.4769, 0.2753, 0.3117, 0.6573, 3.322368) \), \( f^* = -3.322368 \)

X. References