V International Conference on Computational Methods for Coupled Problems in Science and Engineering
COUPLED PROBLEMS 2013
S. Idelsohn, M. Papadrakakis and B. Schrefler (Eds)

EFFECTIVE MODIFICATIONS TO DIFFERENTIAL EVOLUTION OPTIMIZATION ALGORITHM

ERIC J. INCLAN* AND GEORGE S. DULIKRAVICH*

*Florida International University (FIU)
Multidisciplinary Analysis, Inverse Design, Robust Optimization and Control (MAIDROC) Lab.
Department of Mechanical & Materials Eng., 10555 West Flagler St., Miami, Florida 33174, USA
e-mail: eincl001@fiu.edu; dulikrav@fiu.edu; web page: http://maidroc.fiu.edu

Key words: Optimization, Differential Evolution, Single Objective

Summary. Many inverse problems utilize optimization algorithms to perform minimization of least squares norms in an accurate, reliable and computationally efficient manner. Due to the cost of evaluating real-world objective functions, optimization algorithms must be both fast and robust. Differential Evolution (DE) algorithm is known for its robustness, but not its speed. This paper proposes four simple modifications to DE and compares their performance to Particle Swarm (PS) algorithm using a subset of the Schittkowski & Hock test cases. With these techniques, DE is observed to converge to the global minimum up to three times faster than PS, while maintaining robustness in some cases, and generally performs better than the original forms of DE.

1 INTRODUCTION

Researchers have utilized optimization algorithms when solving inverse problems and performing inverse shape design because a variety of inverse problems can be formulated as optimization problems [1]. In these situations, the effectiveness of the optimization algorithm will dictate the quality and computational cost of the solution. Over the past ten years two optimization algorithms, namely DE and PS, have enjoyed widespread use due to their outstanding performance in a variety of applications. As with any stochastic method, there is always room for improvement but the difficulty often lies in justifying the increased complexity of the new subroutine. This paper proposes four simple modifications to DE that improve its speed, robustness, or both. Since PS is generally considered to be the faster of the two [12], it will be used as the benchmark method.

2 OPTIMIZATION ALGORITHMS

2.1 Differential Evolution (DE)

Differential Evolution (DE) utilizes an equation (referred to here as the “update equation”) in order to generate a new solution, and replaces an existing solution with the new one if the new solution is superior.

The standard DE algorithm can be described as follows.

1) Create initial population of candidate solutions.
2) Evaluate objective function(s) for each solution.

```
3) Begin main loop:
   a) Copy original population to temporary population.
   b) For each solution in the temporary population,
      i. Create a new solution:
         1. Randomly select one dimension, j, of solution and apply update equation.
         2. For each dimension (excluding j) of the current solution,
            a. If R < CR, apply update equation, otherwise, leave unchanged.
      ii. Evaluate objective function(s) for new solution.
      iii. Compare new solution to corresponding solution from original population
      iv. If new solution is superior to original solution, replace the original with the
          new solution.
4) End main loop once population converges or maximum number of iterations is
   reached.
```

In the above algorithm, CR is a user-defined scalar $\in$ [0,1] known as the "crossover rate," and R is a uniformly distributed, random number $\in$ [0,1].

There are many forms of DE currently in use, several of which vary only by the update equation used. Three particularly successful forms are the so-called rand/1/bin, and best/2/bin proposed in [10] as well as Donor3 proposed in [2]. Their respective update equations are as follows,

$$Y_k = X_{r1,k} + F\left(X_{r2,k} - X_{r3,k}\right) \tag{1}$$

$$Y_k = X_{best,k} + F\left(X_{r1,k} + X_{r2,k} - X_{r3,k} - X_{r4,k}\right) \tag{2}$$

$$Y_k = \frac{\lambda_1 X_{r1,k} + \lambda_2 X_{r2,k} + \lambda_3 X_{r3,k}}{\lambda_1 + \lambda_2 + \lambda_3} + F\left(X_{r2,k} - X_{r3,k}\right) \tag{3}$$

where Y is the resulting coordinate in the kth dimension of the new solution, and F is a weighting factor (a user-defined scalar $\in$ [0,2]). The variable X denotes a coordinate from an existing solution vector, and the subscript r indicates that it was randomly selected. Therefore, the component Y in Eq. 1, for rand/1/bin, is a linear combination of components from three distinct, randomly selected solutions, while Eq. 2, for best/2/bin is the linear combination of four distinct, randomly selected solutions and the global best solution. The Donor3 method utilizes a weighted average of three components, where λ_1, λ_2, and λ_3 are uniformly distributed, random numbers $\in$ [0,1].

2.2 Particle Swarm (PS)

Published within two years of DE, Particle Swarm (PS) has become very popular due to its simplicity and speed. It is based on the social behavior of various species [4] and, like DE, uses an update equation to generate new solutions.

The basic PS algorithm is given below.

```
1) Create initial population of solutions.
2) Evaluate objective function(s) for each solution.
```

3) Store copy of initial population to serve as "individual best" vectors and store the "global best."
4) Begin main loop:
   a) For each solution in the population,
      i. Apply update equation.
      ii. Evaluate objective function(s) using new solution.
      iii. Replace "individual best" with new solution if new solution is superior.
   b) Replace "global best" if best new solution is superior to previous "global best."
5) End main loop once population converges or maximum number of iterations is reached.

The update equations are,

$$\vec{X}_i^{g+1} = \vec{X}_i^{g} + \vec{V}_i^{g} \tag{4}$$

$$\vec{V}_i^{g} = \alpha\vec{V}_i^{g-1} + \beta R_1\left(\vec{X}_{best,i} - \vec{X}_i\right) + \beta R_2\left(\vec{X}_{best,G} - \vec{X}_i\right) \tag{5}$$

where $\vec{V}$ is the so-called "velocity vector," α and β are user defined scalars, g is the iteration number, and R_1 and R_2 are uniformly distributed random numbers □ [0,1]. The vector $\vec{X}_{best,i}$ is the vector corresponding to the best value ever held by the ith solution, and $\vec{X}_{best,G}$ is the best solution ever found (also known as the "global best"). The first term on the right of Eq. 5 is the "inertia," which is effectively a scalar multiple of the velocity from the previous iteration. Since the velocity magnitude can grow each iteration, it is possible for the inertia to become so large that PS oscillates erratically or diverges altogether.

To help mitigate this risk the following condition was imposed (written in C-style pseudo-code),

```
while ($V_k > domain/3$)
    $V_k = V_k / 2$                                   (6)
```

where V_k is the kth coordinate of the velocity vector. The condition above causes the velocity component to be halved until its magnitude is less than one-third the width of the search space for that coordinate. This restriction is applied before the velocity is added to the individual vector. A second restriction was also added to the velocity vector, as follows (again in pseudo-code)

```
while ($X_{i,k} + V_{i,k}$ outside domain)
    $V_{i,k} = V_{i,k} \times 0.9$                    (7)
```

This restriction causes the entire velocity vector to be scaled down further if it would cause the new solution to move outside of the search domain.

3 MODIFICATIONS

Over the years many modifications to DE have been proposed. Some of these modifications relate to the update equation [6], while others involve merging DE with some other technique [11,3]. This paper proposes four modifications that draw more performance out of DE while minimizing the changes made to the implementation of DE.

3.1 Randomly Varying Parameters

It is widely known that the values of F and CR greatly impact the performance of DE. Some authors have suggested values based on their experience [10], and others have used meta-optimization to determine what values to use [7]. However, no single value provides optimal performance for all problems, and many values are only good for specific problems. To date, two authors have discussed randomly varying one parameter, F, between a fixed range [11, 5]. It will be shown that randomly varying both F and CR will dramatically improve the convergence speed and robustness of the DE methods in many test functions.

3.2 Special Vectors

Since DE involves the weighted average of a collection of vectors, it may be beneficial to include two special vectors in the population after each iteration. This paper proposes including an average vector, and a weighted average vector. The average vector is simply the average of all vectors in the population. Similarly, the weighted average vector is scaled based on the superiority of each vector (the better the solution the larger the weight). In this paper, each solution was non-uniquely ranked in reverse order (the best solution had a rank equal to the population number, and the worst solution had a rank equal to one), and the weights were simply the value of the rank. In order to include these two vectors, the population was increased by two regardless of the problem dimensionality. At the end of each generation, the two worst solutions from the population were replaced by the average vector and weighted average vector in order to make them available for selection during the following update step of DE. This is expected to increase the convergence speed of DE whenever the population converges within a convex region, which it often does at some point in most optimization problems.

3.3 Sorted Comparisons

The standard practice in DE is to compare a given new solution to a corresponding existing solution, but the selection of solutions to compare is purely arbitrary. Due to the computational expense of evaluating the objective function for each new solution, sorting the mutant population may help maximize the benefit of each function evaluation. In this paper, two sorting methods are proposed: “Most” and “Best”. The Most method sorts the new solutions so that the maximum possible number of replacements is achieved. While conducting this research, the Most method did significantly improve DE in the majority of test cases, therefore, those results will be withheld. The Best method sorts the existing population from best to worst, and the new solution population from worst to best. This way, only the best solutions of each population are preserved after each comparison. This method will be shown to greatly increase convergence speed but reduce robustness, causing the algorithm to converge to local minima more frequently.

3.4 Dynamic Random Generation Rate

Sometimes, when an algorithm is too greedy, it is possible to increase the algorithm's robustness by including a probability of randomly generating a new solution. DE and PS do not typically include this. However, since the Best method may be too greedy, such a change is warranted. To ensure that the Random Generation Rate (RGR) is used only when the algorithm converges, the RGR will vary dynamically based on the population's convergence rate. This rate will be measured by storing the average objective function values of the population for the five most recent iterations.

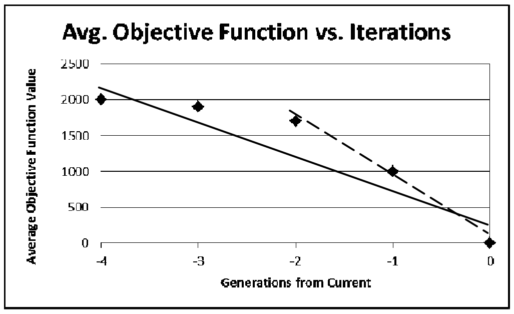


Figure 1: Example of RGR data and slopes

Using those values, a least-squares-fit linear regression will be drawn through the five points (shown in Fig. 1). Since one might want to consider the effects of concavity on the RGR, a second line will be fit through the last three points in order to implicitly capture this information (albeit imperfectly). Finally, a weighted average of the two slopes will be taken. This final value will be the slope used to calculate the RGR. The RGR itself will be calculated using the binary sigmoid equation,

$$\mu = \frac{2\mu_{\max}}{1+e^{-\sigma m}} \tag{8}$$

where μ is the RGR, $\mu_{\max}$ is the maximum rate set to 0.1, σ is a user-defined parameter set to 1.5, and m is the slope, which is assumed to be nonpositive since DE converges monotonically. Note that in order to make this method valid for objective function values of any order of magnitude, the slopes are first translated so that the most recent value is always zero, and then scaled using a simple division operation. Additionally, since this mutation method will cause DE to no longer decrease monotonically, the random generation rate is set to zero whenever the slope of either linear regression is positive.

Of all the methods proposed, this is by far the most complicated. Nevertheless, it will be shown that it can improve the overall performance of DE when used in conjunction with the Best method.

4 RESULTS

In order to evaluate the impact of these modifications, the algorithms were applied to a subset of the Schittkowski and Hock test cases [8][9], listed in Table 1. Constraints were enforced using a penalty function.

Table 1: Dimension and Population of Various Test Problems

Dimension	Population	Test Problem Number
2	22	1, 4, 5, 8, 12, 14, 21, 23, 57, 88, 201
3	30	28, 29, 30, 32, 35, 65, 245, 249
4	36	38, 40, 41, 43
5	41	45, 48, 86, 266
6	45	54, 55, 87, 93, 96, 271
7	49	100
9	56	108
10	59	110, 281, 283, 291
15	72	384, 386
20	83	288, 394
30	101	292, 391
50	130	293, 395
100	184	302

Each test problem was optimized 50 times, for 200 generations using the population sizes in Table 1. The algorithm used to generate these population values has been withheld because it is still undergoing research. The unmodified versions of DE have a weighting factor F = 0.8 and crossover rate CR = 0.9. The randomly varying versions are defined as follows: rand/1/bin uses F □ [0.4,0.8], CR □ [0.7,0.9], best/2/bin uses F □ [0.2,0.8], CR □ [0.6,1.0], and Donor3 uses F □ [0.4,0.8], CR □ [0.4,0.8]. In PS, the user-defined scalars α, and β are set to 0.5 and 2, respectively. To save space in the following graphs and tables, rand/1/bin is denoted STD, best/2/bin is denoted BST, and Donor3 is denoted DN3. The letters following the acronym correspond to the subsections of Section 3 (A = randomly varying parameters, B = special vectors, C = sorted comparisons, and D = dynamical RGR). Note that the special vectors were only used with best/2/bin.

Table 2 shows the aggregate accuracy of each method (note PS is shown in bold next to rand/1/bin). The first row for each method, labeled “Mean,” lists the percentage of test problems for which a given method obtained the lowest average minimum. In some test problems, multiple methods converged to the global minimum, therefore, the sum of these rows exceeds 100%. The second row is the percentage of test problems for which a given method obtained the lowest standard deviation of minima. As indicated in Table 2, BST-A had the best performance (mean and standard deviation) of any method for the 48 test problems selected. The third row lists the percentage of test problems for which a given method found the best single result, while the fourth row relates to the worst single result. The use of randomly varying parameters dramatically improves the performance of best/2/bin. The other modifications also improve its performance but usually at the cost of robustness.

Nevertheless, all modified versions of best/2/bin outperform PS. Donor3's robustness generally worsens, while the STD-A,C method rivals PS in robustness and accuracy.

Table 2: Comparison of Method Accuracy

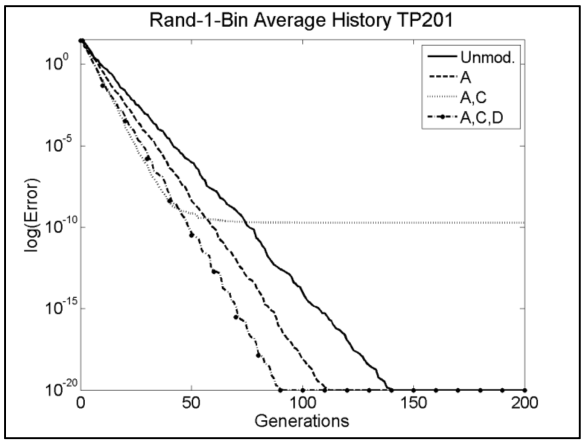
	Method												
Value	STD	STD - A	STD - A,C	STD - A,C,D	**PS**	DN3	DN3 - A	DN3 - A,C	DN3 - A,C,D	BST	BST - A	BST - A,C	BST - A,C,D	BST - A,B,C, D
Mean	8.3%	8.3%	**22.9%**	12.5%	**22.9%**	12.5%	4.2%	6.3%	0.0%	8.3%	**50.0%**	**33.3%**	**35.4%**	**33.3%**
Std. Dev.	8.3%	8.3%	**20.8%**	16.7%	**22.9%**	12.5%	4.2%	6.3%	0.0%	8.3%	**47.9%**	**33.3%**	**37.5%**	**33.3%**
Best	16.7%	25.0%	**31.3%**	29.2%	**41.7%**	18.8%	27.1%	29.2%	22.9%	10.4%	**56.3%**	**70.8%**	**58.3%**	**79.2%**
Worst	0.0%	0.0%	**0.0%**	0.0%	**10.4%**	0.0%	2.1%	39.6%	12.5%	31.3%	**0.0%**	**4.2%**	**0.0%**	**0.0%**

Table 3 shows the aggregate speed of each method. Here, the "Mean" lists the percentage of test problems for which a given method obtained the highest average rate of convergence (including premature convergence to local minima). The second row is the percentage of test problems for which a given method obtained the lowest standard deviation of convergence rates. The third row lists the percentage of test problems for which a given method converged the fastest, while the fourth row relates to the slowest convergence. The convergence rate was based on the best individual of the population for each generation of the optimization run. The convergence rate is the slope of the secant line, subtracting the final objective function value of the global best from the initial objective function of the global best and dividing by the number of generations.

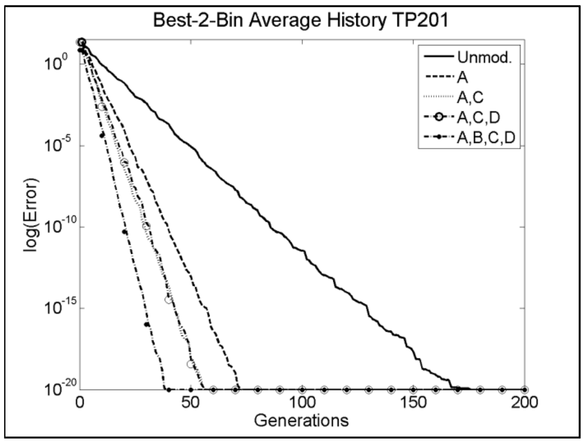
Table 3: Comparison of Method Convergence Rates

	Method													
Value	STD	STD - A	STD - A,C	STD - A,C,D	**PS**	DN3	DN3 - A	DN3 - A,C	DN3 - A,C,D	BST	BST - A	**BST - A,C**	BST - A,C,D	**BST - A,B,C, D**
Mean	0.0%	0.0%	0.0%	0.0%	**12.5%**	0.0%	0.0%	10.4%	0.0%	0.0%	8.3%	**35.4%**	0.0%	**33.3%**
Std. Dev.	27.1%	14.6%	4.2%	0.0%	**4.2%**	6.3%	4.2%	0.0%	2.1%	27.1%	6.3%	**2.1%**	0.0%	**2.1%**
Best	2.1%	0.0%	0.0%	0.0%	**10.4%**	0.0%	2.1%	12.5%	0.0%	4.2%	0.0%	**18.8%**	4.2%	**45.8%**
Worst	4.2%	0.0%	0.0%	0.0%	**10.4%**	2.1%	2.1%	33.3%	16.7%	27.1%	0.0%	**2.1%**	0.0%	**2.1%**

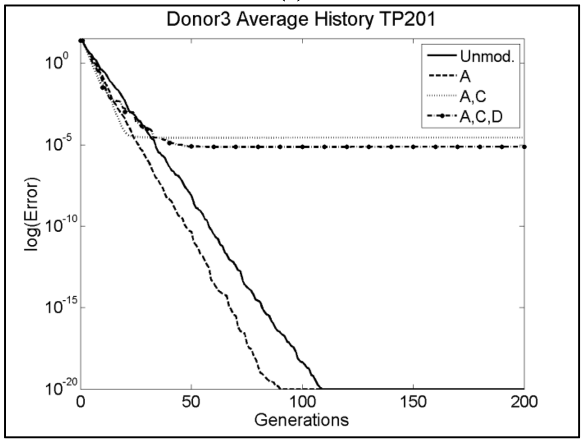
From Table 3 we see that, when applied to best/2/bin, the true benefit of the B, C, and D modifications is speed. In nearly half of the test problems BST-A,B,C,D had the single highest speed of any method, and in one third of problems had the highest average convergence rate of any method. One of the most striking examples of improvement in best/2/bin is test problem (TP) 201. Each convergence history, like those in Fig. 2, contains the average convergence history for a particular method. As shown in Fig. 2, the unmodified versions of rand/1/bin and best/2/bin are slower than PS, while Donor3 is slightly faster. The fully modified best/2/bin (BST- A,B,C,D in Fig. 2b) is roughly four times faster than the unmodified version, and roughly three times faster than PS. The STD-A,C,D (Fig. 2a) is the fastest of the rand/1/bin versions, and is nearly 35% faster than PS. The DN3-A version of Donor3 (Fig. 2c) is slightly faster than the original version. While the other modifications are faster, they also cause Donor3 to converge to local minima as indicated by the curve's rapid initial descent followed by a region of near-zero slope. The test cases TP 5, 8, and 88 show similar behavior.



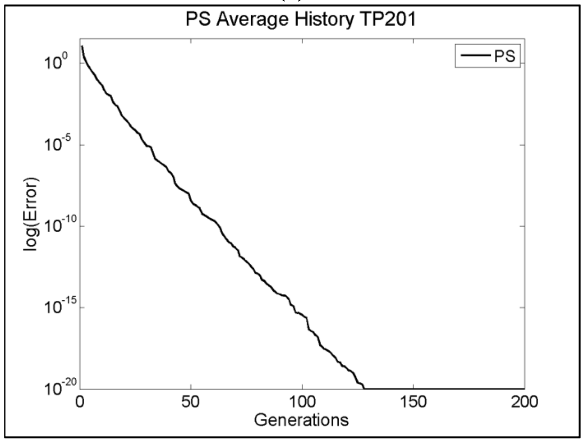
(a)



(b)



(c)



(d)

Figure 2: Convergence Histories for TP 201

For some test cases like TP 45, or TP 4 shown in Fig. 3, PS dominates all variations of DE. Here, PS is over four times faster than any DE variation, and much more robust. However, in several cases the proposed modifications of DE can make methods that normally perform worse than PS perform much better. As shown in Fig. 4, each modification causes best/2/bin to outperform PS in speed and robustness, whereas the unmodified version could not. While randomly varying parameters increases its speed, adding sorted comparisons dramatically improves its robustness. The special vectors provide an additional speed boost while slightly decreasing robustness. The rand/1/bin method shows similar improvements to a lesser degree. However, STD-A,C,D shows slower convergence than STD-A,C (Fig. 4a). In best/2/bin, the

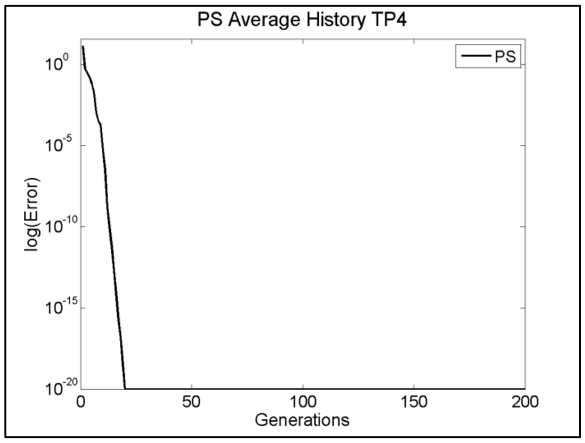
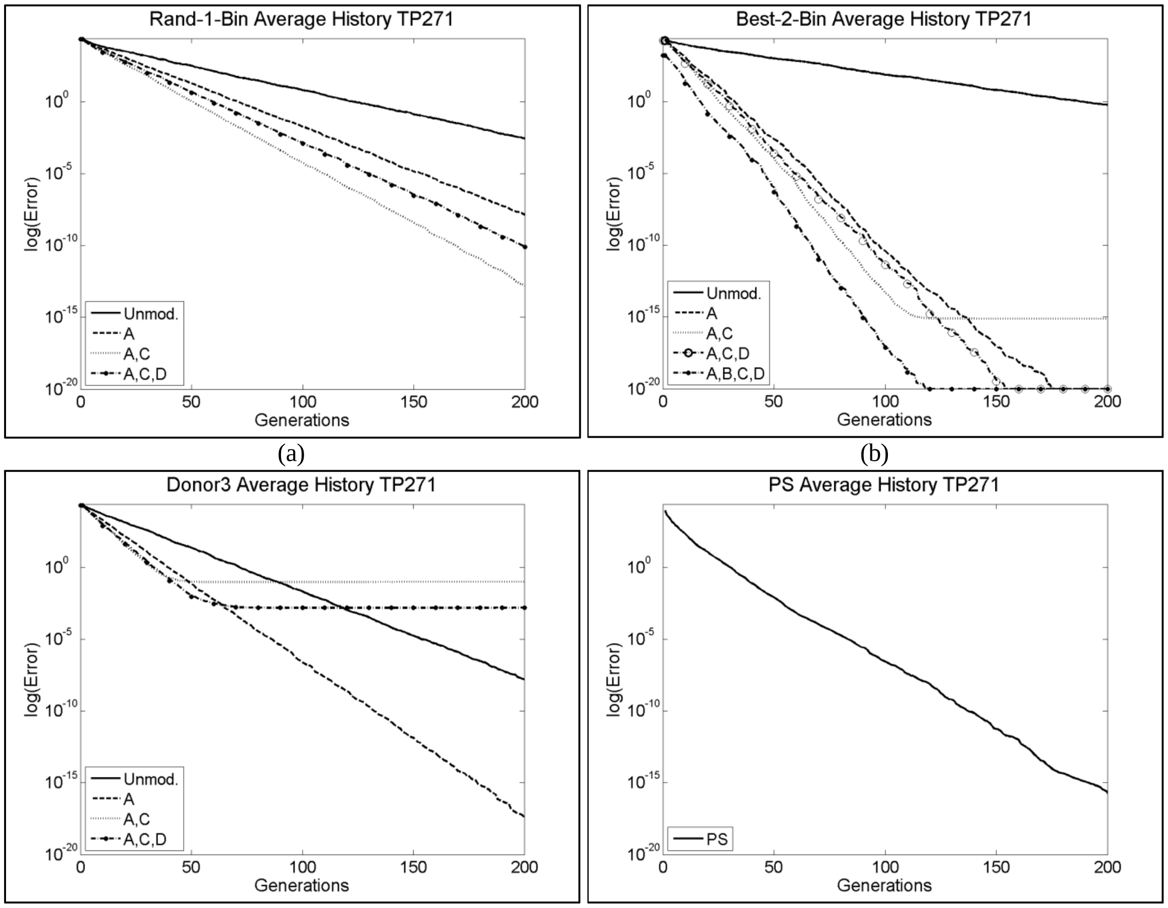


Figure 3: PS Convergence History for TP 4

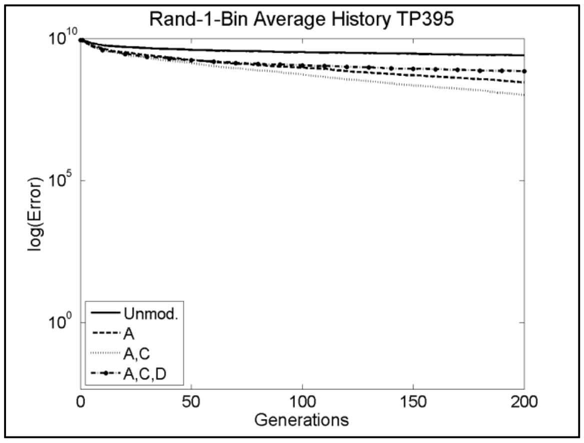


(a) (b) (c) (d)

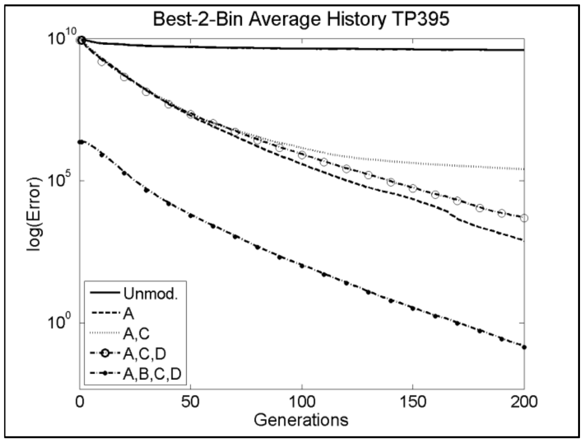
Figure 4: Convergence Histories for TP 271

BST-A,C,D version is just as fast, but less robust than BST-A,C (Fig. 4b). In Donor3, only randomly varying parameters increases speed and, in this case, preserves robustness. The other variations increase speed, but reduce robustness. Similar behavior can be observed in TP 110, 249, 266, 281, 283 and 288.

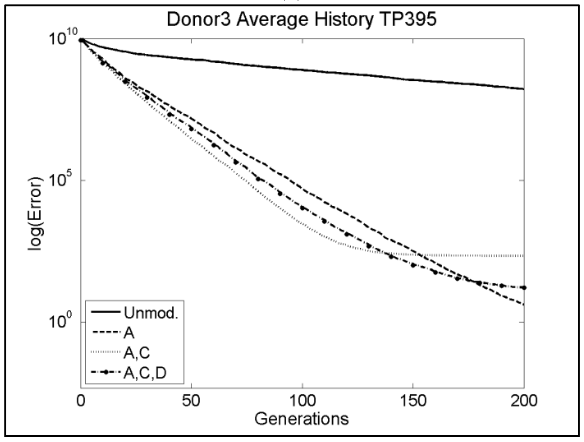
In cases where the objective function is convex, using special vectors can provide an advantage early on as shown in Fig. 5. The DN3-A (Fig. 5b) version converges faster than any best/2/bin variation, but the use of special vectors compensates for that disadvantage. Similar behavior is observed in TP 291, 292, 293, 302, 384, 391 and 394. Note that the objective function need not be strictly convex in order for the special vectors to improve performance.



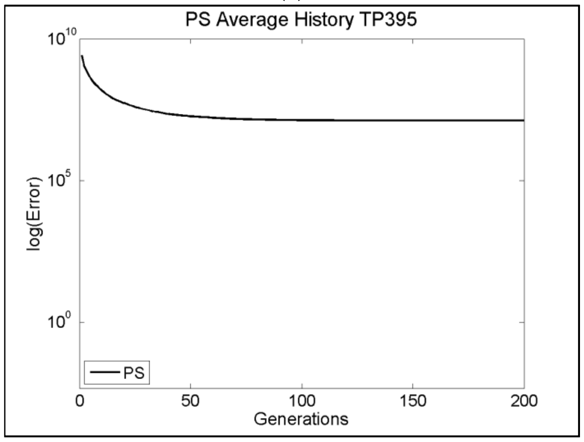
(a)



(b)



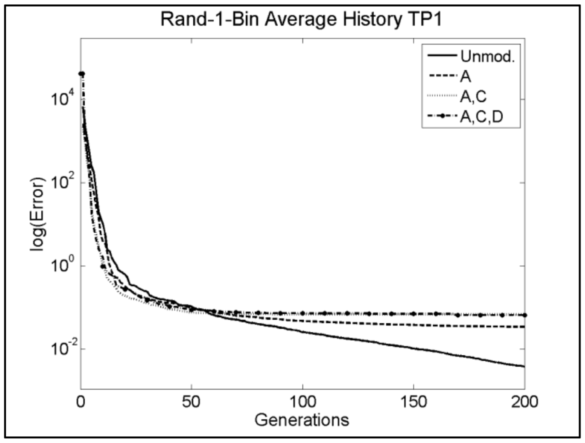
(b)



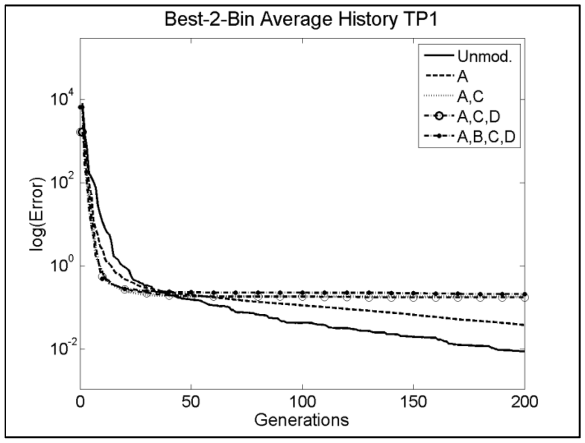
(b)

Figure 5: Select Convergence Histories for TP 395

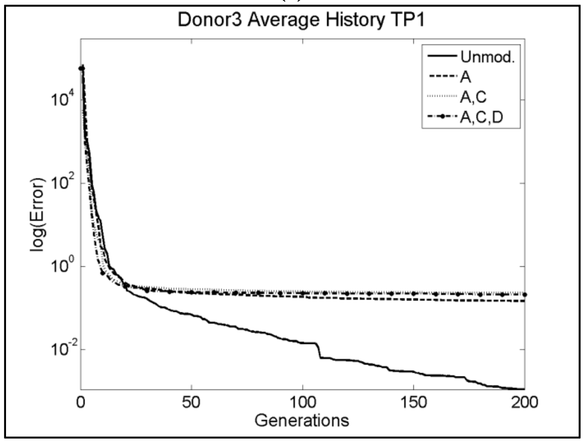
As with any optimization algorithm, these modifications are not guaranteed to improve DE's performance for every function. Sometimes, although improvements to speed can be observed, DE can still be trapped in local minima. Fig. 6 shows improvements to each DE method's speed, but not robustness. Again best/2/bin (Fig. 6b) can outpace PS, and rand/1/bin (Fig. 6a) can match PS. Furthermore, these modifications do not always improve behavior when applied sequentially (i.e., A,B,C will not always be superior to A,B, which will not always be superior to A).



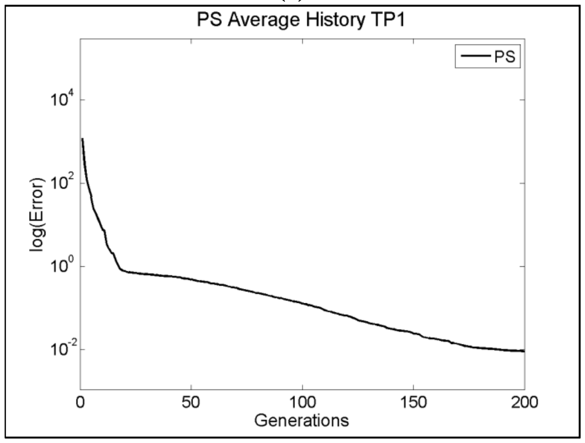
(a)



(b)



(c)



(d)

Figure 6: Convergence Histories for TP 1

5 CONCLUSIONS

The single most effective modification to rand/1/bin and best/2/bin is randomly varying parameters based on the ranges provided in this paper. The use of additional modifications has been shown to greatly improve the performance of best/2/bin, making it faster and more robust than particle swarm algorithm in many cases. The effects of these modifications on Donor3 are mixed.

ACKNOWLEDGEMENTS

This work was partially funded by the US Air Force Office of Scientific Research under grant FA9550-12-1-0440 monitored by Dr. Ali Sayir. 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 US Air Force Office of

Scientific Research 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] Cuco, A. P., Neto, A. J., Velho, H. F., & de Sousa,F. L. (2007). Solution of an Inverse Adsorption Problem with an Epidemic Genetic Algorithm and the Generalized Extremal Optimization Algorithm. (eds: G. S. Dulikravich, M. J. Colaco, H. R. Orlande, and M. Tanaka), *Proceedings of the International Symposium on Inverse Problems, Design and Optimization*, vol. 2, pp. 430-437. Miami, Florida, USA.

[2] Fan, H.-Y., Lampinen, J., and Dulikravich, G. S. (2003). 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.

[3] Iba, H., and Noman, N. (2008). Accelerating Differential Evolution Using an Adaptive Local Search. *IEEE Transactions on Evolutionary Computation, 12*(1), pp. 107-125.

[4] Kennedy, J., & Eberhart, R. (1995). Particle Swarm Optimization. *Proceedings., IEEE International Conference on Neural networks*, pp. 1942-1948.

[5] Konar, A., Chakraborty, U. K., and Das, S. (2005). Two Improved Differential Evolution Schemes for Faster Global Search. In H.-G. Beyer (Ed.), *Proceedings of the 2005 Conference on Genetic and Evolutionary Computaton (GECCO '05)* (pp. 991-998). New York: ACM.

[6] Lampinen, J., and Fan, H.-Y. (2003). A Trigonometric Mutation Operation to Differential Evolution. *Journal of Global Optimization*, pp. 105-129.

[7] Pedersen, M. E. (2010). Good Parameters for Differential Evolution. *Magnus Erik Hvass Pedersen*. Retrieved 2 2012, from Hvass Labs: http://www.hvass-labs.org/people/magnus/publications.php

[8] Schittkowski, K. (1987). *More Test Examples for Nonlinear Programming Codes*. Berlin: Springer-Verlag.

[9] Schittkowski, K., and Hock, W. (1981). *Test Examples for Nonlinear Programming Codes - All Problems from the Hock-Schittkowski-Collection*. Bayreuth: Springer.

[10] Storn, R., and Price, K. (1997). Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *Journal of Global Optimization*, pp. 341–359.

[11] Das, S., Abraham, A., Konar, A., Liu, Y., Sun, A., Loh, H. *et al*. (2008). Particle Swarm Optimization and Differential Evolution Algorithms: Technical Analysis, Applications and Hybridization Perspectives. In *Advances of Computational Intelligence in Industrial Systems, v*ol. 116, pp. 1-38, Berlin / Heidelberg: Springer.

[12] Thomsen, R., and Vesterstrom, J. (2004). A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems. *Congress on Evolutionary Computation, CEC2004.*, vol. *2*, pp. 1980-1987.