SELF-ADAPTIVE DIFFERENTIAL EVOLUTION WITH SQP LOCAL SEARCH

Janez Brest, Aleš Zamuda, Borko Bošković, Sašo Greiner,
Mirjam Sepesy Maučec, Viljem Žumer
Faculty of Electrical Engineering and Computer Science, University of Maribor
Maribor, Slovenia
{janez.brest; ales.zamuda; borko.boskovic; saso.greiner; mirjam.sepesy; zumer}@uni-mb.si

Abstract In this paper we present experimental results of self-adaptive differential evolution algorithm hybridized with a local search method. The results of the proposed hybrid algorithm are evaluated on a set of benchmark functions provided by the IEEE Congress on Evolutionary Computation (CEC 2008) special session on Large Scale Global Optimization. Performance comparison of our algorithm with other algorithms is reported.

Keywords: Differential Evolution, Local search, Optimization, Self-adaptation

1. Introduction

In recent years numerous stochastic optimization algorithms have been proposed to solve real-parameter optimization problems, such as evolution strategies, real-parameter genetic algorithms, simulated annealing, differential evolution, particle swarm optimization, ant-colony optimization, evolutionary algorithms, etc. The optimization problem is to find $\vec{x}$, which optimizes the objective function $f(\vec{x})$ where $\vec{x} = [x_1, x_2, ..., x_D]^T$ is a set of real variables. D is the dimensionality of the search space. Domains of the variables are defined by their lower and upper bounds: $x_{j,low}$, $x_{j,upp}$; $j \in \{1, ..., D\}$. A priori knowledge about the objective function is usually very limited and, in practice, the objective function is often nonlinear, multi-modal, etc.

In this paper we hybridize our self-adaptive differential evolution algorithm jDEdynNP [6] with a local search procedure. The performance of the new algorithm is evaluated on a set of benchmark functions provided by CEC 2008 special session on Large Scale Global Optimization

(LSGO) [21] at the IEEE World Congress on Computational Intelligence (WCCI 2008).

2. Background

In this section we give an overview of previous work. The references to source code of the original differential evolution (DE) algorithm are presented. Then the self-adaptive mechanism used in our DE algorithm is briefly outlined. In section 3 the SQP local search procedure is described.

To solve high-dimensional problems [21], cooperative coevolution [15, 19] can be used. Liu et al. [12] used FEP (Fast Evolutionary Programming) with cooperative coevolution (FEPCC) to speedup convergence rates on large-scale problems, Bergh and Engelbrecht [23] used a Cooperative Approach to Particle Swarm Optimisation (PSO), Yang, Tang and Yao used differential evolution with cooperative coevolution (DECC) [25], recently. Gao and Wang [9] used a memetic DE algorithm for high-dimensional problem optimization. There were 8 papers accepted to LSGO at CEC 2008: [26, 10, 6, 13, 22, 27, 28, 24] and many of them used DE.

2.1 The Differential Evolution Algorithm

DE is a population based evolutionary algorithm proposed by Storn and Price [20, 18]. The original DE has three control parameters: amplification factor of the difference vector – F, crossover control parameter – CR, and population size – NP. During one generation for each vector, DE employs the mutation, crossover and selection operations to produce a new vector for the next generation. DE [20, 16, 8] has been shown to be a simple yet powerful evolutionary algorithm for global optimization in many real problems [14].

In this paper we will skip a detailed description of the DE algorithm. The algorithm is widely used in many research areas and it is implemented in several programming languages (for source code of the algorithm see DE homepage: `http://www.icsi.berkeley.edu/~storn/code.html`).

2.2 The Self-adaptive DE Algorithm

In this subsection we revise our jDEdynNP-F algorithm [6], which was proposed at the CEC 2008 special session. The jDEdynNP-F algorithm applies self-adapted F and CR control parameters and a population size reduction method. Additionally, it implements a mechanism for sign

changing of F control parameter with some probability based on the fitness values of randomly chosen vectors, which are multiplied by the F control parameter (scaling factor) in the mutation operation of the DE algorithm.

The jDEdynNP-F algorithm uses the same self-adaptive control mechanism as it was first proposed in [4] and lately used in many other variants [7, 3, 5, 2]. This mechanism changes the control parameters F and CR during the run.

The jDEdynNP-F algorithm implements the method for gradually reducing population size [5] during the optimization process. The dynamic population size reduction mechanism is used in the jDEdynNP-F algorithm to start optimization with the greatest population at the beginning of the evolutionary process, and finishes optimization with the smallest population size. The population size is gradually reduced. In [5, 6] we proposed a reduction scheme where the new population size is equal to half of the previous population size.

The jDEdynNP-F algorithm applies a mechanism that changes the sign of the control parameter F with some probability ($prob = 0.75$) when $f(\vec{x}_{r_2}) > f(\vec{x}_{r_3})$ during the mutation operation as presented in Fig. 1. *rand* generates random numbers uniformly distributed between 0 and 1. This mechanism uses *rand/1/bin/* DE strategy and was proposed in [6].

```
// individuals' objective function values are stored in array named cost
prob = 0.75;                              // probability for changing the sign
if (rand < prob && cost[r2] > cost[r3])
    F = -F;                               // sign change
```

Figure 1. The control parameter F changes sign.

In this paper, the jDEdynNP-F algorithm is hybridized for the first time with a local search procedure, which is presented in the following section.

3. Sequential Quadratic Programming (SQP)

Sequential Quadratic Programming (SQP) [17, 1] is a non-linear optimization method based on gradient computation. It is a generalization of Newton's method to multiple dimensions, incorporating a quadratic approximation model for the objective function given an initial guess for the solution. Great strength of the SQP method is its ability to solve problems with nonlinear constraints. The approximation model is solved at each iteration to yield a step toward the solution of the original

Table 1. LSGO@CEC'08 benchmark functions

F_1	Shifted Sphere Function	uni-modal	separable
F_2	Shifted Schwefel's Problem 2.21	uni-modal	non-separable
F_3	Shifted Rosenbrock's Function	multi-modal	non-separable
F_4	Shifted Rastrigin's Function	multi-modal	separable
F_5	Shifted Griewank's Function	multi-modal	non-separable
F_6	Shifted Ackley's Function	multi-modal	separable
F_7	FastFractal "DoubleDip" Function	multi-modal	separable

problem. As with most optimization methods, SQP is not a single algorithm, but rather a conceptual method from which numerous specific algorithms have evolved [1].

The algorithm for SQP we have used is FSQP-AL and its implementation is given in CfSQP [11]. The norm of descant direction was set to ϵ =1.e–8. Constraints were enforced only in terms of bounds to search parameters, i.e. linear bound-constraints were used.

Hybridization of the chosen local search algorithm in our global optimization jDEdynNP-F algorithm was as follows. First, the global algorithm was run for 30% of maximum number of function evaluations (MAXFEs). Then after every 100 generations (note that we have dynamic population size during the evolutionary process), we employed the local search method on the fittest individual of the current population. The number of iterations of the SQP method that were used to refine the given solution was set to $\lfloor \frac{\sqrt{D}}{5} \rfloor$.

After the SQP local search procedure is called, we check whether the new obtained individual (result from SQP procedure) is better than the currently best individual. If SQP finds a better individual (in this case SQP returns a positive value), it will be stored, otherwise when SQP returns a negative value we do not use SQP in the rest of the evolutionary process. The suggested mechanism seems to work fine with fractal function F_7, when the SQP local search procedure usually could not make any improvement of the currently best individual.

4. Experimental Results

In this section we present results of experiments, which were made in order to present the performance of the proposed hybrid algorithm.

Table 1 shows characteristics of CEC 2008 benchmark functions.

Table 2 presents the obtained results of our hybrid algorithm on the benchmark functions. The error values ($f(\vec{x}) - f(\vec{x}^*)$) are presented

Table 2. Error values achieved for problems F_1–F_6, with $D = 1000$. Function value achieved for function F_7 with $D = 1000$

	jDEdynNP-F [6]		jDEdynNP-F with SQP	
	Mean	Std. dev.	Mean	Std. dev.
F_1	1.1369e–13	0.0000	1.1141e-13	1.1369e-14
F_2	1.9529e+01	2.2525	2.6582e+01	1.6529
F_3	1.3136e+03	1.3635e+02	3.2719e+02	1.7768e+02
F_4	2.1668e–04	4.0563e–04	8.3060e-12	3.6271e-12
F_5	3.9790e–14	1.4211e–14	5.0022e-14	1.2389e-14
F_6	1.4687e–11	2.4310e–11	3.7630e-13	3.8851e-13
F_7	–1.3491e+04	4.6038e+01	-1.3766e+04	8.2836e+01

in the table. The optimal solution results are known for benchmark functions F_1–F_6, while for function F_7 the optimal solution value is not given.

Figure 2 shows the convergence graphs for functions F_1–F_7 on $D = 100$ with and without SQP local search procedure for the best, median and worst individuals obtained at the end of the evolutionary process. Figures 3 and 4 show convergence graphs for functions when $D = 500$ and $D = 1000$, respectively.

The convergence graphs show that the algorithm with the SQP performs better than the algorithm without SQP in most cases, exception is function F_2 (*Schwefel's Problem 2.21*) when $D = 1000$. The algorithm with the SQP obviously gives better results on functions F_3 (*Rosenbrock's function*), and F_4 (*Rastrigin's function*) when $D = 1000$.

In this experiment we did not make fine tuning of the SQP's parameters, i.e. when starting SQP, how many iterations may be used by SQP, etc.

The summary result of the LSGO 2008 competition are available at `http://nical.ustc.edu.cn/papers/CEC2008_SUMMARY.pdf`, where results comparison on $D = 1000$ functions are presented. The jDEdynNP-F algorithm took third place, after [22] and [24].

The mean value obtained by our proposed algorithm with the SQP is lower than 1.e–10 (roughly speaking, a function *is solved*, when mean value drops under 1.e–10) for functions F_1 (6), F_4 (2), F_5 (6), and F_6 (4). In the parentheses after function we give a number of LSGO algorithms that also reached bound 1.e–10 (note, the competition included eight algorithms).

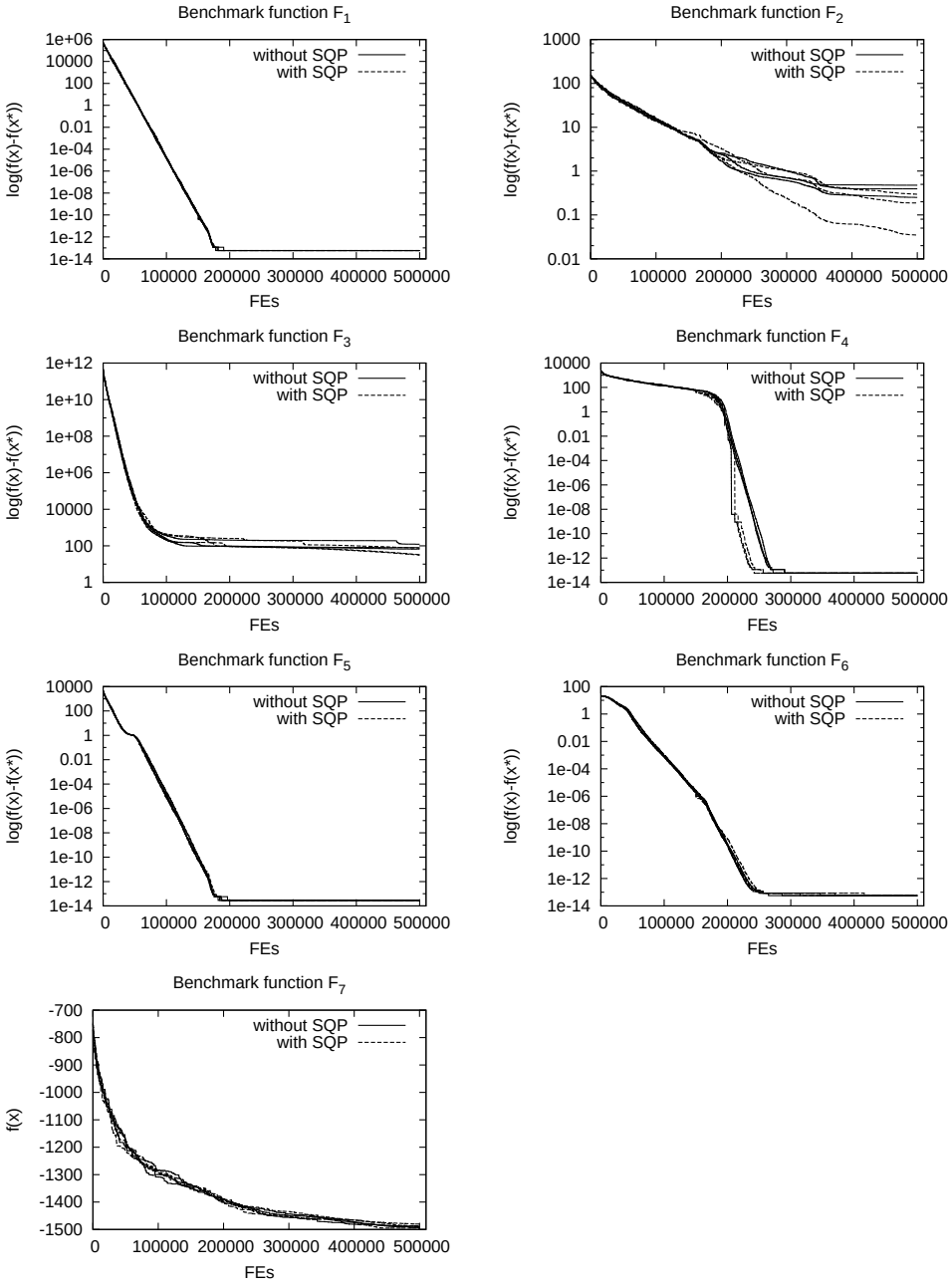

Figure 2. Convergence graphs for functions F_1–F_6 with $D = 100$.

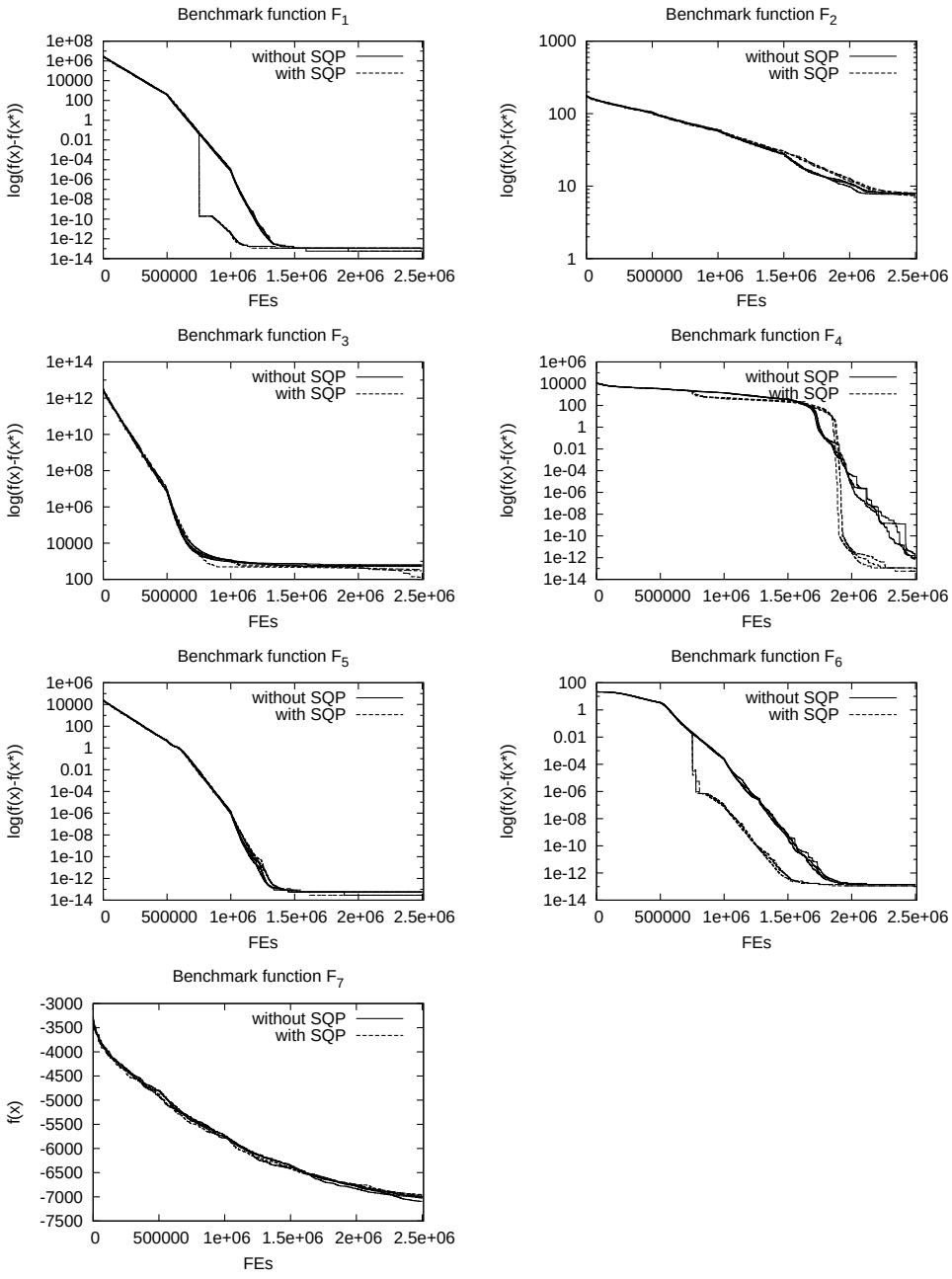

Figure 3. Convergence graphs for functions F_1–F_6 with $D = 500$.

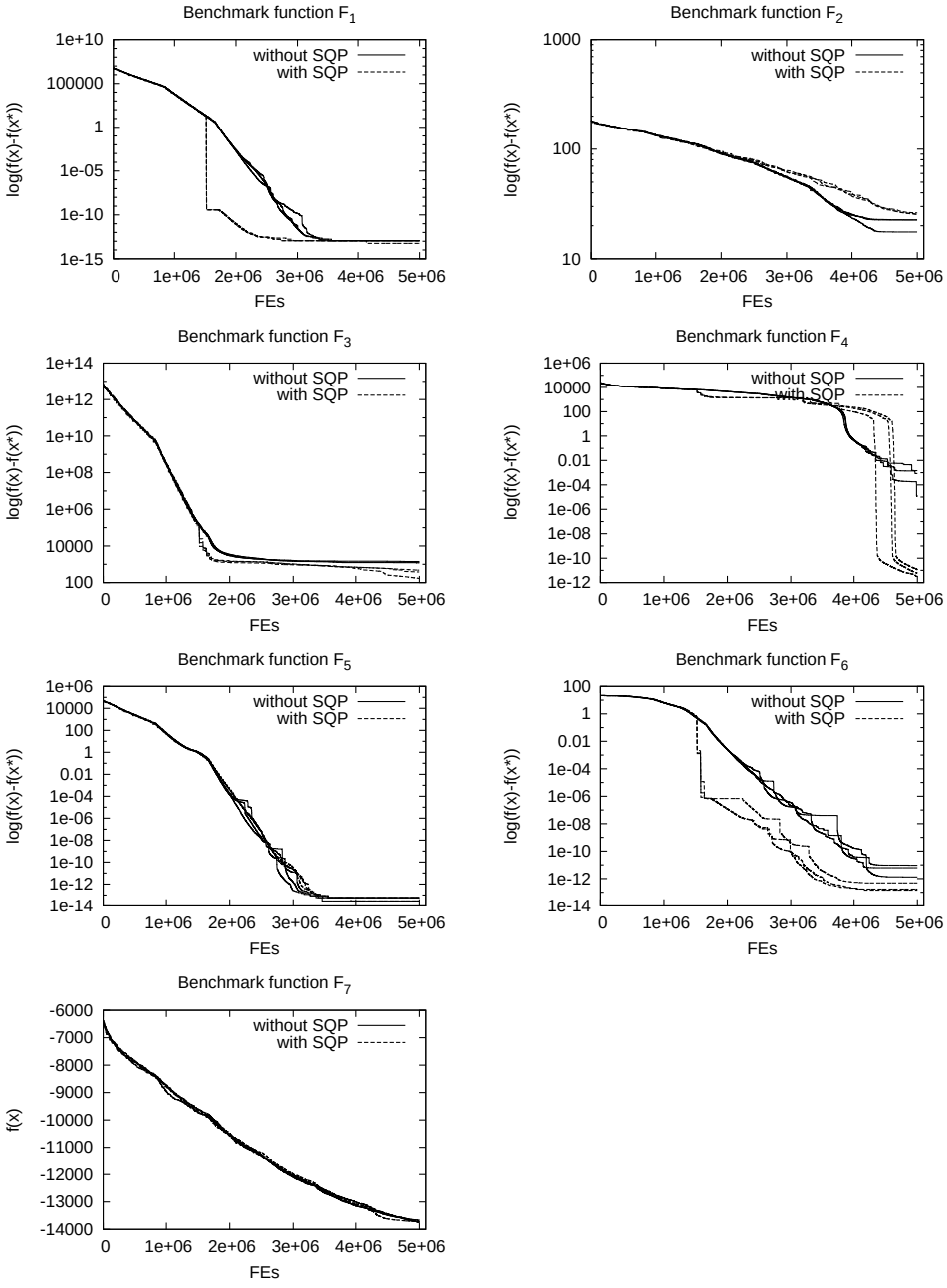

Figure 4. Convergence graphs for functions F_1–F_6 with $D = 1000$.

Our algorithm has rank 4, 2, and 3 for functions F_2, F_3, F_7, respectively. Based on this comparison of the jDEdynNP-F with SQP with the LSGO 2008 results, we can conclude that our algorithm is highly competitive on all LSGO 2008 functions when $D = 1000$.

5. Conclusions

This paper presents our attempt to hybridize self-adaptive differential jDEdynNP-F algorithm with SQP local search procedure. The experimental results confirm that the proposed hybrid algorithm might perform better than the algorithm without SQP. The better parameter setting for the SQP and deep insight of it are challenges for future work.

Acknowledgment

We thank the authors of CfSQP for providing us with the academic license of their software.

References

[1] P.T. Boggs and J.W. Tolle. Sequential quadratic programming for large-scale nonlinear optimization. *J. Comput. Appl. Math.*, 124(1-2):123–137, 2000.

[2] J. Brest. Differential Evolution with Self-Adaptation. In Juan Ramon Rabunal Dopico, Julian Dorado de la Calle, and Alejandro Pazos Sierra, editors, *Encyclopedia of Artificial Intelligence*, Information Science Reference, Hershey, 2008, pp. 488–493.

[3] J. Brest, B. Bošković, S. Greiner, V. Žumer, and M. Sepesy Maučec. Performance comparison of self-adaptive and adaptive differential evolution algorithms. *Soft Comput.*, 11(7):617–629, 2007.

[4] J. Brest, S. Greiner, B. Bošković, M. Mernik, and V. Žumer. Self-Adapting Control Parameters in Differential Evolution: A Comparative Study on Numerical Benchmark Problems. *IEEE T. Evol. Comput.*, 10(6):646–657, 2006.

[5] J. Brest, and M. Sepesy Maučec. Population Size Reduction for the Differential Evolution Algorithm. *Appl. Intell.*, DOI: 10.1007/s10489-007-0091-x.

[6] J. Brest, A. Zamuda, B. Bošković, M. Sepesy Maučec, and V. Žumer. High-Dimensional Real-Parameter Optimization using Self-Adaptive Differential Evolution Algorithm with Population Size Reduction. In *Proc. IEEE Congress on Evolutionary Computation (CEC 2008)*, pages 2032-2039, Hong Kong, 2008.

[7] J. Brest, V. Žumer, and M. Sepesy Maučec. Control Parameters in Self-Adaptive Differential Evolution. In B. Filipič and J. Šilc, editors, *Bioinspired Optimization Methods and Their Applications*, pages 35–44, Ljubljana, Slovenia, October 2006.

[8] V. Feoktistov. *Differential Evolution: In Search of Solutions (Springer Optimization and Its Applications)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[9] Y. Gao and Y.-J. Wang. A Memetic Differential Evolutionary Algorithm for High Dimensional Functions' Optimization. In *Proc. Third International Conference on Natural Computation (ICNC 2007)*, pages 188–192, 2007.

[10] S.-T. Hsieh, T.-Y. Sun, C.-C. Liu, and S.-J. Tsai. Solving Large Scale Global Optimization Using Improved Particle Swarm Optimizer. In *Proc. IEEE World Congress on Computational Intelligence*, pages 1777–1784, Hong Kong, 2008.

[11] C. T. Lawrence, J.L. Zhou, and A.L. Tits. User's Guide for CFSQP Version 2.5: A C Code for Solving (Large Scale) Constrained Nonlinear (Minimax) Optimization Problems, Generating Iterates Satisfying All Inequality Constraints. Technical report TR-94-16r1, Institute for Systems Research, University of Maryland, College Park, MD, 1997.

[12] Y. Liu, X. Yao, Q. Zhao, and T. Higuchi. Scaling Up Fast Evolutionary Programming with Cooperative Coevolution. In *Proc. IEEE Congress on Evolutionary Computation (CEC 2001)*, pages 1101–1108, 2001.

[13] C. MacNish and X. Yao. Direction Matters in High-Dimensional Optimisation. In *Proc. IEEE World Congress on Computational Intelligence (WCCI 2008)*, pages 2377–2384, Hong Kong, 2008.

[14] Z. Michalewicz and D.B. Fogel. *How to Solve It: Modern Heuristics*. Springer, Berlin, 2000.

[15] M.A. Potter and K. De Jong. A Cooperative Coevolutionary Approach to Function Optimization. In Y. Davidor, H.-P Schwefel, and R. Männer, editors, *Parallel Problem Solving from Nature – PPSN III*, pages 249–257, 1994.

[16] K.V. Price, R.M. Storn, and J.A. Lampinen. *Differential Evolution, A Practical Approach to Global Optimization*. Springer, 2005.

[17] G.V. Reklaitis, A. Ravindran, and K.M. Ragsdell. *Engineering Optimization: Methods and Applications*. Wiley-Interscience, 1983.

[18] J. Rönkkönen, S. Kukkonen, and K.V. Price. Real-Parameter Optimization with Differential Evolution. In *Proc. IEEE Congress on Evolutionary Computation (CEC 2005)*, pages 506–513, Edinburgh, UK, 2005.

[19] D. Sofge, K. De Jong, and A. Schultz. A Blended Population Approach to Cooperative Coevolution for Decomposition of Complex Problems. In *Proc. IEEE Congress on Evolutionary Computation (CEC 2002)*, pages 413–418, 2002.

[20] R. Storn and K. Price. Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *J. Global Optim.*, 11:341–359, 1997.

[21] K. Tang, X. Yao, P. N. Suganthan, C. MacNish, Y. P. Chen, C. M. Chen, and Z. Yang. Benchmark Functions for the CEC'2008 Special Session and Competition on High-Dimenasional Real-Parameter Optimization. Technical report, Nature Inspired Computation and Applications Laboratory, USTC, China, 2007. `http://nical.ustc.edu.cn/cec08ss.php`.

[22] L.-Y. Tseng and C. Chen. Multiple Trajectory Search for Large Scale Global Optimization. In *Proc. IEEE World Congress on Computational Intelligence (WCCI 2008)*, pages 3057–3064, Hong Kong, 2008.

[23] F. van den Bergh and A.P. Engelbrecht. A Cooperative Approach to Particle Swarm Optimisation. *IEEE T. Evol. Comput.*, 8(3):225–239, 2004.

[24] Y. Wang and B. Li. A Restart Univariate Estimation of Distribution Algorithm: Sampling under Mixed Gaussian and Lévy probability Distribution. In *Proc. IEEE World Congress on Computational Intelligence (WCCI 2008)*, pages 3918–3925, Hong Kong, 2008.

[25] Z. Yang, K. Tang, and X. Yao. Differential Evolution for High-Dimensional Function Optimization. In *Proc. IEEE Congress on Evolutionary Computation (CEC 2007)*, pages 3523–3530, Singapore, 2007.

[26] Z. Yang, K. Tang, and X. Yao. Multilevel Cooperative Coevolution for Large Scale Optimization. In *Proc. IEEE World Congress on Computational Intelligence (WCCI 2008)*, pages 1663-1670, Hong Kong, 2008.

[27] A. Zamuda, J. Brest, B. Bošković, and V. Žumer. Large Scale Global Optimization Using Differential Evolution With Self-adaptation and Cooperative Co-evolution. In *Proc. IEEE World Congress on Computational Intelligence (WCCI 2008)*, pages 3719–3726, Hong Kong, 2008.

[28] S.Z. Zhao, J.J. Liang, P.N. Suganthan, and M.F. Tasgetiren. Dynamic Multi-Swarm Particle Swarm Optimizer with Local Search for Large Scale Global Optimization. In *Proc. IEEE World Congress on Computational Intelligence (WCCI 2008)*, pages 3846–3853, Hong Kong, 2008.