Parallelization of the Rosen-Suzuki Fist Function and Himmelblau Function Using a Two-Population Evolutionary Algorithm

Michael Scherger
Department of Computer Science
Texas Christian University
Fort Worth, TX, 76129, USA
Email: m.scherger@tcu.edu

Roja Ramani Molupoju
Department of Computing Science
Texas A&M University – Corpus Christi
Corpus Christi, TX, 78412, USA
Email: molupoju.roja@gmail.com

Abstract – *(ICAI'15) The general nonlinear programming (NLP) problem requires finding the optimum point (minimum or maximum) of a function of n real variables subjected to some given constraints. This research presents the parallel implementation of the Rosen-Suzuki Frst function and the Himmelblau function using a two-population genetic algorithm. There are numerous situations in science and engineering where the optimum is bounded which adds complexity to the optimization problem. The approach used in this research by [32] uses evolves two populations (male and female). One population is evolved inside the feasible domain of the design space and the second population is evolved outside this feasible domain. In this research a version of this parallel, two-population algorithm was implemented. The results of this research show that the two-population parallel genetic algorithm is effective and accurate for these functions.*

Keywords: Evolutionary algorithms, constrained optimization problems, parallel algorithms.

1 Introduction

A constrained optimization problem is usually written as a nonlinear programming (NLP) problem of the following type. The problem seeks to minimize:

$$F(x_1, \ldots, x_i, \ldots, x_n) \ (1 \le i \le n)$$

subject to side constraints:

$$x_{i\,min} \le x_i \le x_{i\,max} \ (1 \le i \le n)$$

and inequality and equality constraints:

$$g_j(x_1, \ldots, x_n) \le 0 \text{ where } (1 \le j \le m)$$
$$h_j(x_1, \ldots, x_n) = 0 \text{ where } (1 \le j \le p).$$

For these types of NLP problems, there are n variables, m inequality constraints, and p equality constraints. The function $F(X)$ is called the objective function, $g_j(X)$ is the j^{th} inequality constraint, and $h_k(X)$ is the k^{th} equality constraint. The i^{th} variable can varies in range from $[x_{imin} .. x_{imax}]$.

Searching the boundary between the feasible and the infeasible regions is critical for any function to find the global optimum.[30] The inability of evolutionary systems to search precisely the boundary area is the main reason for difficulty in locating the global optimum. Some of the constraints are active at the global optimum.

For this reason, in may constrained optimization problems, it is more difficult to locate the global optimum [23]. A two-population evolutionary computation approach was proposed in paper [31] where two populations of individuals are evolved: one of feasible and the other of infeasible by which the search pressure upon the boundaries of the feasible space can be increased. Feasible individuals (also called females) are the ones that are evolved inside (including the boundaries of the feasible space), while the infeasible individuals (also called males) are evolved outside the feasible space. These two populations are subject to mutation and asexual crossover. Female-male crossover ensures the search pressure on the boundaries of the feasible space. This crossover can be performed between two or more feasible and infeasible individuals, and with, or without, favoring the better-fit females or the better-ranked males.

When two populations, one of feasible and one of infeasible individuals, are distinctively evolved, these two populations interact systematically (rather than occasionally) for the purpose of exploring the boundaries of the feasible space. Since the criterion based on which individuals are assigned to the two populations does not change during evolution, the behavior of the two populations is easier to understand.

The two-population evolutionary algorithm described in this paper, was proposed by [31] and was also implemented with two other constrained optimization problems in [30]. This algorithm was initially tested for functions with two variables; this project works with the algorithm for functions with multiple variables and is tested. This algorithm is implemented using C++ and MPI on a small parallel computing cluster. The program was tested for two benchmark problems from literature:

the Rosen-Suzuki function and the Himmelblau function. Evaluation of the effectiveness of this technique is provided in sections 5 and 6.

2 Related Research

Genetic algorithms (GAs), developed by Holland (1975), have traditionally used a more domain independent representation, namely, bit-strings. However, many recent applications of GAs have focused on other representations, such as graphs (neural networks), Lisp expressions, ordered lists, and real-valued vectors.

GAs are highly abstract computational models inspired by natural selection [14]. Standard GAs apply an *a priori* defined fitness function (e.g. the function one wants to optimize) to an individual. They typically use an all-at-once calculation where individuals are evaluated immediately after their creation (i.e. birth). Fitness calculation in nature is substantially different [1]. It consists of a continuous series of tests during an individual's life originating from a complex environment. This environment is not only influenced by the animal's own actions but also by the other individuals as well as other processes occurring in the world (e.g. climatologically or geophysical changes). Summarizing one can say that - in contrast with GAs - nature uses a far more partial but continuous fitness evaluation in order to adapt to a complex world.

One of the key problems for using GAs in practical applications is to design the fitness function, particularly when we do not know where the global optimum is located [19] [22] [33]. A comparative estimate of how good as a solution turns out to be enough in most cases. However, for constrained problems, determining a way to estimate how close in an infeasible solution from the feasible region is difficult since most real-world problems have complex linear and non-linear constraints, and several approaches have been proposed in the past to handle them [14][33]. From those, the penalty function seems to be yet the most popular technique for engineering problems, but the intrinsic difficulties to define good penalty values makes harder the optimization process using a GA [6] [7] [19].

3 Two Population Parallelization Technique

Evolutionary algorithms are characterized by their repeated fitness evaluation of the individuals in the population. Therefore, it is natural to view them as parallel algorithms. In generational evolutionary algorithms, substantial savings in elapsed time can often be obtained by performing fitness evaluations in parallel. In the simplest form of parallelism, a manager process performs all the function of the evolutionary algorithm except evaluation of individuals, which are performed in parallel by worker processes operating on separate processors. The master process waits for all workers to return the evaluated individuals before varying on with the next generation.

The algorithm in this research is implemented using MPI in which there are *n* processes that work simultaneously using inter-process communication. This performs well when a complex function is being used and there is more number of iterations that are to be performed.

A "Manager-Worker" paradigm is used to implement the algorithm. In this paradigm, there is one Manager process and several workers processes and is depicted in Figure 1. The Manager generates uniform random points and sends these points to the workers; the workers process these points (check the feasibility and compute the value of the function at that point if it is a feasible point) and send back the processed results. The manager would then differentiate them into female and male based on their feasibility value (female if it is zero and male if not). Then the manager ranks the female individuals based on their fitness and the male individuals based on the number of constraints they violate. Generations of new individuals and mutating female and male individuals would also happen. The female-male pairs are formed with the females choosing their males in the rank decreasing order. Then the crossover is performed and all the steps are until the best female is constant for a given number of individuals.

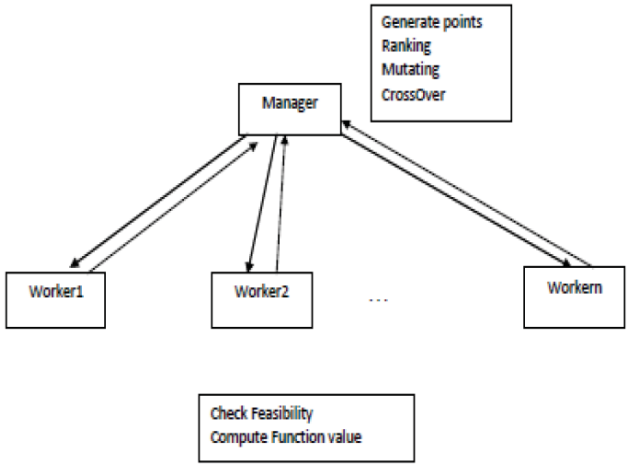

Figure 1: Manager-Worker paradigm for this research.

4 Test Functions

In this project, a technique based on the concept of co-evolution is used to create two populations that interact with each other in such a way that the objective function is minimized. The approach has been tested with two single-objective optimization problems with linear and non-linear inequality constraints and its results are

compared with those produced by other GA-based and mathematical programming approaches.

This research is tested using two benchmark problems from literature: the Rosen-Suzuki function and the Himmelblau function. This section provides the description of these two constrained optimization problems.

4.1 Rosen –Suzuki Fist Function (in four variables)

The Rosen-Suzuki Fist problem is a function in four variables with three non-linear constraints on the variables. Hock and Schittkowski introduced it in 1981. The object function is

$$f(x) = x_1^2 + x_2^2 + 2x_3^2 + x_4^2 - 5x_1 - 5x_2 - 21x_3 + 7x_4$$

The nonlinear constraints are:

$$0 \leq 8 - x_1^2 - x_2^2 - x_3^2 - x_4^2 - x_1 + x_2 - x_3 + x_4$$

$$0 \leq 10 - x_1^2 - 2x_2^2 - x_3^2 - 2x_4^2 + x_1 + x_4$$

$$0 \leq 5 - 2x_1^2 - x_2^2 - x_3^2 - 2x_1 + x_2 + x_4$$

The theoretical value for the optimum minimum for this function is -44 and is located at the point (0, 1, 2, -1). The best value that is obtained using this research is -43.9756 at the point (-0.00021, 0.99821, 2.00281, -0.98912). The best value reported in literature is (-0.0005463221, 1.000618, 2.000213, -0.9996195) at which the optimum value for the function is -44 using the GaNOP system[REF].

4.2 Himmelblau Function (in five variables)

Himmelblau originally proposed the Himmelblau nonlinear optimization problem in five variables in 1972. This function was selected since it was used as a benchmark for several other GA-based techniques. The Himmelblau function has five design variables (x1, x2, x3, x4, x5), six nonlinear inequality constraints, and ten boundary conditions. The problem is mathematically stated as follows:

Minimize:

$$f(X) = 5.3578547x_3^2 + 0.8356891x_1x_5 + 37.29329x_1 - 40792.141$$

Subject to:

$$g_1(X) = 85.334407 + 0.0056858x_2x_5 + 0.00026x_1x_4 - 0.0022053x_3x_5$$

$$g_1(X) = 80.51249 + 0,.0071317x_2x_5 + 0.0029955x_1x_2 + 0.0021813x_3^2$$

$$g_1(X) = 9.300961 + 0.0071317x_2x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4$$

with:

$$0 \leq g_1(X) \leq 92$$
$$90 \leq g_2(X) \leq 110$$
$$20 \leq g_3(X) \leq 25$$
$$78 \leq x_1 \leq 102$$
$$33 \leq x_2 \leq 45$$
$$27 \leq x_3 \leq 45$$
$$27 \leq x_4 \leq 45$$
$$27 \leq x_5 \leq 45$$

5 Summary of Results

The testing and analysis of the results for the Rosen-Suzuki Fist and HimmelBlau functions is presented in this section. The parallel program for each of these functions was run in several configurations varying the number of processes and varying the female and male population sizes.

5.1 Rosen-Suzuki Fist Function

The theoretical value for the optimum minimum for this function is -44 and is located at the point (0, 1, 2, -1). The best value that is obtained using this research is -43.9756 at the point (-0.00021, 0.99821, 2.00281, -0.98912).

Following is a table (table 1) in which the average value for 50 runs with a particular number of processes is performed. The table shows the number of processes (in other words, no. of workers), the female and male population, the number of iterations (for which the best female would be constant), the total time for the run (algorithm time plus function value computation time), the number of function calls and the optimum point for the function. The table shows it only for up to 8 processes, the experiment is performed for up to 20 processes, however, the graph below is the one that is drawn using the number of processes and the total required for the completion of the algorithm.

No.of processes	No. of female points	No. of male points	No. of iterations	Function Value	Total Time	No. of function calls	Optimum Point
1	20	15	1000	-43.0923	11.37833	11571	(-0.04239,0.89619, 1.9886,-0.955717)
2	20	15	1000	-43.9503	9.514633	11578	(-0.000138,0.97898, 2.006346,-0.986454)
3	20	15	1000	-43.9362	9.362131	11598	(-0.002435,0.97898, 2.002316,-0.996354)
4	20	15	1000	-43.7521	8.182058	11611	(-0.01134,0.87698, 2.02318,-0.97867)
5	20	15	1000	-43.2259	9.039228	11620	(-0.00054,0.984798, 1.95489,-0.973141)
6	20	15	1000	-42.6588	8.117775	11635	(-0.00187,0.96715, 1.98712,-0.98715)
7	20	15	1000	-43.5648	9.82167	11635	(-0.00583,0.984923, 2.0182,-0.88342)
8	20	15	1000	-43.872	10.11201	14321	(-0.00034,0.98672, 2.00178,-0.97821)

Table 1: Rosen-Suzuki Fist function results.

Figure 2 shows the graph of time vs. number of processes for algorithm computed at the manager process. Figure 2 shows the graph for the number of processes vs. the total time (time for algorithm plus the function value computation time at the worker processes). From the graph, it can be seen that as the number of processes increases, the time decreases. The graph is not a decreasing linearly, since the total time is dependent on the function computation time calculated at the worker processes (also depends on the number of function calls). Figure 3 clearly shows all the computation times in a single graph that shows that the time for algorithm is increased whenever there is an increase in the function value computation time (which depends on the number of function calls).

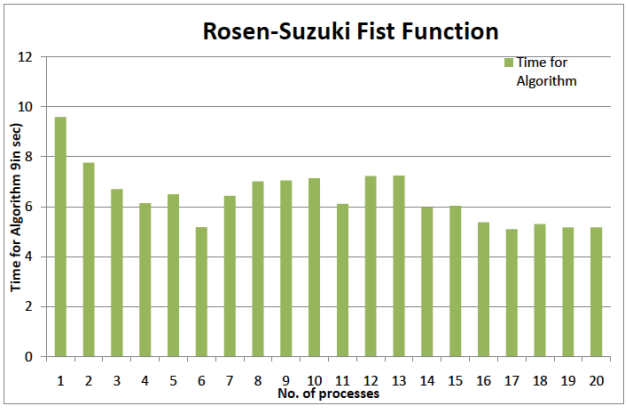

Figure 2: Time vs. number of processes for the Rosen-Suzuki Fist function.

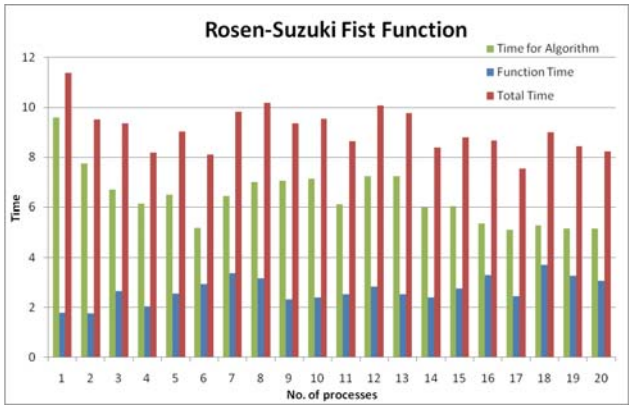

Figure 3: Algorithm, function, and total time vs. number of processes for the Rosen-Suzuki Fist function.

5.2 Himmelblau Function

This problem was originally proposed by Himmelblau and solved using the Generalized Reduced Gradient method (GRG). Gen and Cheng solved this problem using a genetic algorithm based on both local and global reference. The result shown in Table 4 is the best found with their approach.

The mean for the 100 runs performed is $f(X)$ = -30786.5. The worst solution found is $f(X)$ = -30692.9, which is better than the best solution previously reported. The best solution found with this algorithm is $f(X)$ = -30868.9 (corresponding to $x1$= 78.7032, $x2$ = 33.7124, $x3$ = 27.83889, $x4$ = 44.2504 and $x5$ = 43.1159).

There are 20 female and 15 male points generated every iteration. This was run using 8 processing units in which the best solution remained constant for 1000 iterations. The duration for the run is 0.419021 sec. The number of function calls is 59585. Table 2 summarizes the results of the two-population algorithm used in this research with other algorithms.

Design Variables	This algorithm	Gen[11]	Homaifar[14]	GRG[13]
x_1	78.7032	81.4900	78.0000	78.6200
x_2	33.7124	34.0900	33.00000	33.4400
x_3	27.8388	31.2400	29.9950	31.0700
x_4	44.2504	42.2000	45.0000	44.1800
x_5	43.1159	34.3700	36.7760	35.2200
$g_1(X)$	91.8574	90.522543	90.714681	90.520761
$g_2(X)$	100.517	99.318806	98.840511	98.892933
$g_3(X)$	20.0456	20.060410	19.999935	20.131578
$f(X)$	-30868.9	-30183.576	-30665.609	-30373.949

Table 2: Comparison of current algorithm with previous techniques for the Himmelblau function.

6 Conclusions and Future Work

Solving a constrained nonlinear programming problems using a co-evolutionary algorithm was

implemented. The two-population evolutionary algorithm proposed by [31] was used in which two distinct populations are evolved; feasible (females) and infeasible (males). The interaction between these populations concentrate on he boundaries of the feasible space. Mutating the female and male populations induced search capabilities inside the feasible space and parallel to the feasible-infeasible boundary. Figure 4 and Table 2 presents a table of a summary of the results in this research.

Optimization Problem	Theoretical Value	This Algorithm	Best Solution	Constraint Values
Rosen-Suzuki Fist	Value: -44 Point: (0, 1, 2, -1)	Average: -43.5589 (-0.01342, 0.93123, 1.98791, -0.99127)	Value: -42.9756 Point: (-0.00021, 0.99821, 2.00281, -98912)	G1 = -0.02046 G2 = -1.04986 G3 = -0.00185

Figure 4: Summary of results for this research.

There is no penalty factor that is involved in constraint handling; hence there is a clear distinction can be made between the feasible and infeasible individuals. Parallel Implementation of the Co-evolutionary Genetic Algorithm has been a long awaited development for the evolutionary algorithms as mentioned in [7] [19] [31]. In this project, the algorithm has been implemented using MPI, which uses multiple processes that run in parallel, by which the computation time is reduced. This implementation worked well with several test problems that were previously solved using GA-based and mathematical programming techniques, producing in most of the cases results better than those previously reported. The technique is able to achieve such good results with relatively small populations and using a relatively low number of generations.

Functions of multiple variables used in various disciplines of Engineering can be solved for the global optimum using this research. This research provides sound results in terms of the computation time and the number of generations that need to be evolved for a more accurate value. The project is tested using the two benchmark problems from the literature. A detailed comparison of the results obtained from the project with that of those from the literature is presented.

7 Bibliography

[1] Adenike A. Adewuya, "New Methods in Genetic Search with Real-Values Chromosomes", Thesis, Mississippi State University, 1993.

[2] T. Back, D. Fogel and Z. Michalewicz, Handbook of Evolutionary Computation, The Institute of Physics Publishing, 2000.

[3] Benacer R and Pham Dinh Tao, "Global Maximization of a Non-definite Quadratic Function over a Convex Polyhedron", Fermat Days 85: Mathematics for Optimization, Amsterdam, Holland, 1986, pp. 65-76.

[4] Carlos R. Garcia-Alonso, Leonor M. Pérez-Naranjo, Juan C. Fernandez-Caballero, "Multiobjective evolutionary algorithms to identify highly auto correlated areas: the case of spatial distribution in financially compromised farms" *Annals of Operations Research*, Jan 2011.

[5] Carlos A. Coello Coello, "Theoretical and Numerical Constraint-Handling Techniques used with Evolutionary Algorithms: A Survey of the State of the Art," Computer Methods in Applied Mechanics and Engineering, Vol. 191, No. 11-12, pp. 1245-1287, 2002.

[6] Carlos A. Coello Coello, "Constraint-handling using an evolutionary multiobjective optimization technique", *Civil Engineering and Environmental Systems*, 17:319-346, 2000.

[7] Carlos A. Coello Coello, "Use of a Self-Adaptive Penalty Approach for Engineering Optimization Problems", May 1999.

[8] Carlos A. Coello Coello, "An Empirical Study of Evolutionary Techniques for Multiobjective Optimization in Engineering Design", PhD thesis, Department of Computer Science, Tulane University, New Orleans, LA, Apr 1996.

[9] Pei-Chann Chang and Shih-Hshin Chen. The development of a sub-population genetic algorithm II (SPGA II) for multiobjective combinatorial problems, *Applied Soft Computing*, Vol. 9, No. 1, pp. 173-181, January 2009.

[10] C. Y. Cheong, K. C. Tan and B. Veeravalli. A multi-objective evolutionary algorithm for examination timetabling, *Journal of Scheduling*, Vol. 12, No. 3, pp. 121--145, April 2009 .

[11] David W. Coit, Alice E. Smith, David M. Tate, "Adaptive Penalty Methods for Genetic Optimization of Constrained Combinatorial Problems", *Informs Journal On Computing*, Vol. 8, No. 2, Spring 1996, pp. 173-182, DOI: 10.1287/ijoc.8.2.173.

[12] A.E. Eiben, "Multiparent Recombination in Evolutionary Computing," in A.Ghosh and S. Tsutsui, editors, Advances in Evolutionary

Computing, Natural Computing Series, Springer, pp. 175-192, 2002.

[13] Eshelman Larry J & Schaffer David J, "Real-coded Genetic Algorithms and Interval-Schemata", In Whitley, D. L. (Ed.), Foundations of Genetic Algorithms 2 (pp. 187-202). California: Morgan Kaufmann, 1993.

[14] Mistuo Gen and Runwei Cheng, "Genetic Algorithms and Engineering Design", John Weily & Sons, Inc, New York, 1997.

[15] Grefenstette J. J., "Predictive models using fitness distributions of genetic operators", In *Foundations of Genetic Algorithms 3*, D. Whitley (Ed.), San Mateo, CA: Morgan Kaufmann.

[16] David M Himmelblau, "Applied Nonlinear Programming", McGraw Hill, New York, 1972.

[17] Homaifar A, Lai S H Y and Qi X, "Constrained Optimization via Genetic Algorithms, Simulation", 62(4):242-254, 1994.

[18] Antony Iorio, Xiaodong Li, "Parameter Control within a Co-operative Co-evolutionary Genetic Algorithm", May 2002.

[19] Kenneth De Jong, William Spears, "On the State of Evolutionary Computation", September, 1993.

[20] [Kenneth Dee Jong A., "Are genetic algorithms function optimizers?" *Proceedings of the Second International Conference on Parallel Problem Solving from Nature*, 1992.

[21] Kenneth De Jong, William Spears, Thomas Back, David Fogel, Hugo de Garis, "An Overview of Evolutionary Computation", April, 1993.

[22] Kim J. H and Myung H, "Evolutionary programming techniques for constrained optimization problems," *IEEE Trans. Evolutionary Computation*, vol. 1, no. 2, pp. 129–140, 1997.

[23] Michalewicz Z and Attia N, "Evolutionary Optimization of Constrained Problems", In Proceedings of the 3rd Annual Conference on Evolutionary Programming, pages 98-108, World Scientific, 1994.

[24] Michalewicz Z and Schoenauer M, "Evolutionary Algorithms for Constrained Parameter Optimization Problems," Evolutionary Computation, Vol. 4, No. 1, pp. 1-32, 1996.

[25] Papalambros P and Li H L, "Notes on the Operational utility of Monotonicity in Optimization", ASME Journal of Mechanisms, Transmissions and Automation in Design, Vol.105, 1983, pp. 174-180.

[26] Papalambros P and Wilde D J, "Principles of Optimal Design – Modeling and Computation", Cambridge Univ. Press, New York, 2000.

[27] Paredis Jan, "Coevolutionary Algorithms", *Proceedings of the Seventh International Conference on Genetic Algorithms* ed TBaeck (San Mateo, CA: Morgan Kaufmann) pp 393-399, 1997.

[28] Ragsdell K M and Phillips D T, "Optimal Design of a class of Welded Structures Using Geometric Programming", ASME Journal of Engineering for Industry, Vol. 98, pp. 1021-1025, 1997.

[29] Rosenbrock, H. H., "An automatic method for finding the greatest or least value of a function", *The Computer Journal* 3: 175–184, doi:10.1093/comjnl/3.3.175, ISSN 0010-4620, MR0136042.

[30] Scherger M.C., Molupoju R. R., "The Parallelization of Two Constrianed Optimization Problems Using Evolutionary Algorithms", Genetic and Evolutionary Methods, WorldComp, July, 2014.

[31] Simionescu P.A., Dozier G.V. and Wainwright R.L. "A Two-Population Evolutionary Algorithm for Constrained Optimization Problems," IEEE World Congress on Computational Intelligence, Vancouver, Canada, July 16-21, 2006.

[32] Simionescu P.A., Beale D.G. and Dozier G.V. "Constrained Optimization Problem Solving Using Estimation of Distribution Algorithms," IEEE World Congress on Evolutionary Computation, Portland, OR, June 20-23, 2004.

[33] Min-Jea Tahk and Byung-Chan Sun, "Co evolutionary Augmented Lagrangian Methods for Constrained Optimization", IEEE Transactions On Evolutionary Computation, Vol. 4, No. 2, July 2000.

[34] Aimin Zhou, Bo-Yang Qu, Hui Li, Shi-Zheng Zhao, Ponnuthurai Nagaratnam Suganthan, Qingfu Zhang, " Multiobjective evolutionary algorithms: A survey of the state-of-the-art," *Swarm and and Evolutionary Computation*", March 2011.