A Novel Hybrid Constraint Handling Technique for Evolutionary Optimization

Ashish Mani and C. Patvardhan

***Abstract*— Evolutionary Algorithms are amongst the best known methods of solving difficult constraint optimization problems, for which traditional methods are not applicable. However, there are no inbuilt or organic mechanisms available in Evolutionary Algorithms for handling constraints in optimization problems. These problems are solved by converting or treating them as unconstrained optimization problems. Several constraint handling techniques have been developed and reported in literature, of which, the penalty factor and feasibility rules are the most promising and widely used for such purposes. However, each of these techniques has its own advantages and disadvantages and often require fine tuning of one or more parameters, which in itself becomes an optimization problem. This paper presents a hybrid constraint handling technique for a two population adaptive co-evolutionary algorithm, which uses a self determining and regulating penalty factor method as well as feasibility rules for handling constraints. Thus, the method overcomes the drawbacks in both the methods and utilizes their strengths to effectively and efficiently handle constraints. The simulation on ten benchmark problems demonstrates the efficacy of the approach.**

I. Introduction

EVOLUTIONARY Algorithms (EA) are population based stochastic search and optimization techniques inspired by nature's laws of evolution. They have been applied to solve complex constrained optimization problems, which cannot be solved by traditional methods such as calculus-based and enumerative strategies due to lack of proper domain information [1]. They are popular due to their simplicity and ease of implementation.

The general constrained optimization problem (COP) for continuous variable is formulated as follows:

Minimize or Maximize f($\vec{x}$),

Such that $x_{i\min} \leq x_i \leq x_{i\max}$; i - number of variables.

$g_j(\vec{x}) \leq 0$; j - number of inequality constraints

$h_k(\vec{x}) = 0$; k - number of equality constraints

where $\vec{x} = (x_1, x_2, \ldots, x_N) \in R^N$.

One of the known issues with the use of EA for solving COPs is handling of constraints, as no inherent mechanism is available in EA to incorporate the constraints seamlessly without consuming too much extra computational effort. Many approaches have been suggested in literature [2] for handling constraints like repair algorithms [3], [4], [5], penalty factor based methods [6], [7], [8], feasibility rules [12] and multi-objective optimization [9], [10] and Co-Evolution method [14] etc.

The penalty factor method and its derivatives like dynamic and adaptive penalty factor method [1], [8] and Deb's Feasibility Rules and its derivatives like stochastic ranking [11] are most commonly used for handling constraints. With the exception of Death Penalty Factor Method and Deb's Feasibility Rules, they require parameter tuning, which itself is an optimization problem solved by the developers mostly by using trial and error methods and is problem specific. If the penalty factor is small, the resulting solution may be infeasible and if it is too large, the solution found is usually far from the optimum. The death penalty factor method and Deb's feasibility rules do not require any parameter tuning and are straightforward in implementation. Further, their primary feature is preference of feasible solutions over infeasible ones. However, they impose death penalty and in complex cases fails to find near optimum solutions [2].

Researchers have tried to solve the above-mentioned problems by introducing modifications in generic penalty factor method and feasibility rules. The dynamic and adaptive penalty factor methods are examples of modification of penalty factor based method to solve the problems associated with the static penalty. However, they still require some parameters to be fine-tuned. The Stochastic ranking and other ranking based EAs are modifications of feasibility rule method to allow for better flexibility in search of high quality solutions by allowing some infeasible solutions to be treated as better than the feasible solutions, if their objective function values are better. They are relatively complicated in implementation and involve additional computation of ranking as well as parameter tuning [11].

This paper proposes an innovative approach to solve the problems in constraint handling by presenting a hybrid constraint handling scheme that utilizes the best features of both the Deb's Feasibility Rules and Adaptive Penalty Factor method. The hybrid scheme employs two separate populations and exchanges some solutions with each other. The simulations have been run on a suite of ten benchmark problems and have been compared with a set of well-known algorithms. The performance of the proposed technique

Manuscript received November 14, 2008.

Ashish Mani is Lecturer in USIC, Dayalbagh Educational Institute, Dayalbagh, Agra-5, India. (e-mail: ashish.mani@rediffmail.com).

C. Patvardhan is Professor in Dept. of Electrical Engg., Faculty of Engineering., Dayalbagh Educational Institute, Dayalbagh, Agra-5, India (e-mail: cpatvardhan@hotmail.com).

978-1-4244-2959-2/09/$25.00 © 2009 IEEE

competitive in comparison with other algorithms.

The rest of the paper is organized as follows. Section 2 analyses the current constraint handling techniques to arrive at design guidelines for improving constraint handling in EA. The proposed algorithm is presented in Section 3 and experimental results are discussed in Section 4. Section 5 concludes with a brief summary and throws some light on direction of future endeavor.

II. Analysis of Constraint handling Techniques

Repair algorithms are suited for specific problems where underlying structure can be utilized for providing computationally efficient algorithms. However, repair algorithms may in some cases, waste computational effort in converting infeasible solutions into the feasible ones [7], [2].

Penalty factor methods and their variants are popular approaches for handling constraints, which have been very useful in some specific cases. However, the drawback of such methods is their limited applicability and need for fine-tuning one or more parameters. The need for fine tuning parameter(s) can be overcome by using Death Penalty factor method or Feasibility Rules [2]. Though the death penalty method and feasibility rules are parameter-less and prefer feasible solutions, they limit the search. Thus cannot explore efficiently complex solution spaces where feasible regions are disjoint. The penalty factor methods with small penalty does have better ability to search complex disjoint solution spaces but they end up mostly with infeasible solution [2], [11].

Stochastic Ranking has been considered as one of the best known methods of handling constraints as it improves the searching ability by allowing infeasible solutions to win over feasible solutions stochastically. However, they are complicated in concept and implementation as well as require some parameter tuning [11].

The following design guidelines emerge from the study of various constraints handling techniques:

- It should be computationally efficient.
- It should prefer feasible solutions to infeasible solutions.
- It should explore entire solution space and avoid premature convergence.
- It should minimize fine-tuning of parameters and parametric values should be universal i.e. should not be problem specific.
- It should be simple in concept and implementation.

A comparison of some of the popularly known methods of constraint handling on the guidelines is presented in Table I. Though each constraint handling technique would meet these criteria to a certain degree depending on their specific implementation, they have been evaluated on their generic ability to meet the criterion.

III. Algorithm

It is clear from the above analysis that there is no single known technique, which meets all the criterion desired in a constraint handling technique. Therefore, there are two possible solutions to this problem. The first solution is to improve one of the existing techniques to meet the desired criteria and the other solution is to hybridize two existing but complimentary techniques. The paper advocates the selection of latter approach as it is simpler in concept and implementation also.

The paper proposes to hybridize Adaptive Penalty Factor method with Feasibility Rules method as they both are complementary and together they meet all the desired criteria. The hybridization is done using two independent populations evolving independently, each using one of the constraints handling technique. However, they exchange part of the population at the end of each cycle. Thus, each of them is guided by the other. This has the advantage of better exploration by Adaptive Penalty Factor method and, at the same time, feasibility of the optimum solution is guaranteed by Feasibility Rules Method.

The proposed method may appear similar to co-evolutionary technique [14] but the proposed method is significantly different from it. The co-evolutionary method in [14] is based on evolution of two populations with opposite objectives to solve saddle-point problems and uses augmented Lagrangian formulation. The limitation of such algorithms is well known in solving nonlinear COPs.

TABLE I
Comparison of Constraint Handling Techniques

Criterion	Repair	Penalty Factor Method (small Penalty)	Death Penalty / Feasibility Rules	Stochastic Ranking	Desired Technique
Computational Efficiency	Poor	Good	Good	Good	Good
Feasible soln. always wins	Yes	No	Yes	No	Yes
Exploration	Good	Good	Poor	Good	Good
Fine tuning	Yes	Yes	No	Yes	No
Conceptual and Implement-ation Simplicity	No	Yes	Yes	No	Yes

The proposed technique is implemented using a simple real coded EA as it is well known that the performance of any EA used for solving COPs is dependent on both evolutionary search part as well as the method used for constraint handling [11]. The evolutionary search part in the

proposed algorithm has been purposefully kept simple so that the efficiency of the proposed constraint handling technique can be measured and compared through experimental runs. The proposed algorithm is as given below:

A. Proposed Algorithm

1. Initialize two equal set of random Populations named A and B.
2. Compute Adaptive Penalty Factor.
3. Find fitness of Population A and B.
   (a) Compute fitness of Population A using Adaptive Penalty Factor method.
   (b) Compute fitness of Population B using Deb's Feasibility Rules.
4. Assign number of Children to each parent in both Population A and B on the basis of their fitness.
5. Crossover is performed independently for both Population A and B.
6. Local search is performed independently for both Population A and B.
7. Selection of next generation for both Population A and B.
8. Swap some members of Population A with Population B.
9. Terminate or Loop.

1) Initialization: Two Population A and B are chosen of same size and initialized randomly. The size of the population is dependent on the problem. The number of parents is based on the number of variables i.e. a factor of 10 is multiplied to the number of variables to arrive at the number of parents in the population as it has been recommended in [13]. The number of children is determined by multiplying the total number of parents again by a factor of 10.

2) Adaptive Penalty Factor: It is designed to make the objective function value of the fittest parents in the population A and B equal for each generation. This automatically chooses a new value of penalty factor for each generation to guide the search in entire solution domain. The equation for computing adaptive penalty factor 'S', is as follows:

If ($CV_{Ai} > 0$)
{ $S = (f_{BBi}(x) - f_{BAi}(x)) / CV_{Ai}$ for Minimization
$S = (f_{BAi}(x) - f_{BBi}(x)) / CV_{Ai}$ for Maximization }
If ($S <= 0$)
$S = -S$

Where $f_{BBi}(x)$ = Objective function value of Best Parent of Population B in the ith generation.

$f_{BAi}(x)$ = Objective function value of Best Parent of Population A in the ith generation.

CV_{Ai} = Constraint Violation of Best Parent of Population A in the ith generation.

Adaptive penalty factor methods have been known to work better provided the penalty factor is adaptively made smaller if the feasible solutions are being found and made larger otherwise [2]. This is done so that there is an indirect pressure for finding feasible solutions. However, these methods do not provide any assistance in actually finding the feasible solutions. In the two population method being proposed population B has a greater propensity towards feasible solutions because it utilizes the Feasibility Rules and population A ensures that the domain on search is not restricted.

3) Fitness Evaluation: The population A and B are evaluated differently for determining the fitness. Population A's fitness is evaluated by using modified objective function value, which is determined by using adaptive penalty factor.

Objective function = f(x).

Constraint Violation = Degree of infeasibility = $\{\Sigma(g_i^+(x)) + \Sigma(h_i(x))\}$.

Modified Objective function $\Phi = f(x) + S * \{\Sigma(g_i^+(x)) + \Sigma(h_i^*(x))\}$, where $g_i^+(x) = \max\{0, g_i(x)\}$ and $h_i^*(x) = 0$ if $|h_i(x)| - \delta < 0$ else $|h_i(x)|$, $\delta = 10^{-10}$.

Population B's Fitness is evaluated using following rule [12].

i. If the two parents being compared are both feasible, the one with better objective function value f(x) is considered fitter.
ii. If one parent is feasible and the other is unfeasible, the feasible one is fitter.
iii. If both parents are infeasible, the one with lower level of constraint violation or degree of infeasibility is fitter.

4) Assign Number of Children: The number of children to each parent is dependent on its fitness relative to rest of the population i.e. fitter parents will get more number of children. However, even the most unfit parent will at least get one children. This is achieved by ranking of the parents in each population in each generation.

5) Crossover: It is performed to spawn population for next generation by using BLX-α-β crossover scheme. It is applied on both the population A and B independently.

6) Local Search Heuristic: It is based on gradient descent search method and is applied on the parents of both the population.

7) Selection: The parents for next generation are selected by comparing individual parents with their best child and applying tournament selection i.e. the fitter one would make it to the next generation.

8) Swap: It exchanges parts of Population A and B so that search is guided towards global optima while maintaining diversity in either of the populations. The fittest member of

both Population A and B are exchanged with least fit members of either population. Further 5% of the population is randomly selected and exchanged. The 5% criterion ensures that at least one member is exchanged in all problems. The proposed algorithm is dependent on the number of variables in the problem, and, any problem would have at least two variables.

9) Terminate or Loop: If the condition for termination is satisfied, the program would be terminated otherwise it would loop back to second step in the algorithm.

IV. Simulation Results and Discussion

The proposed algorithm has been tested on a set of 10 benchmark problems, of which 9 are minimization problems and 1 is a maximization problem. The source of the ten benchmark problems has been given in Appendix. The number of variables, N, number of inequality constraints, IC, which have been further divided into linear L-IC and nonlinear NL-IC, number of equality constraints, EC and the number of active constraints, AC at the optimum value for each problem is listed in Table-II, which provides some indication of the complexity of each problem.

TABLE II
NATURE OF PROBLEMS

#	N	IC	L-IC	NL-IC	EC	AC	Value	Nature of F
F-1	13	9	9	0	0	6	-15.000	Quadratic
F-2	5	6	0	6	0	2	-30665.539	Quadratic
F-3	4	2	2	0	3	3	5126.498	Cubic
F-4	2	2	0	2	0	2	-6961.814	Cubic
F-5	10	8	3	5	0	6	24.306	Quadratic
F-6	2	2	0	2	0	0	0.095825	Nonlinear
F-7	7	4	0	4	0	2	680.630	Polynomial
F-8	8	6	3	3	0	6	7049.331	Linear
F-9	2	0	0	0	1	1	0.750	Quadratic
F-10	5	0	0	0	3	3	0.053950	Exponential

Thirty independent runs have been performed for each problem using the proposed algorithm, which was implemented in 'C' programming language. Table-III presents results obtained from the experimental study of the implemented algorithm for a maximum of 5000 generations. The known optimal of all the problems along with Best, Median, Mean, Worst and Standard Deviation obtained in the thirty independent runs has been mentioned. The proposed algorithm was able to find feasible solutions with a violation degree of less than or equal to 10^{-10} for equality constraints. This is a much more stringent requirement than the constraint violation reported in [11]. The algorithm was able to find optimal solution consistently for four problems viz., F-1, F-2, F-4, F-6 and F-9. It was further able to find better than known optimum solution in case of F-6. The algorithm was able to find the optimum or near optimum solution in most of the runs for problems F-3, F7 and F-10 as is indicated by the statistical data in Table-II. It was not able to find the optimum solution in case of F-5 and F-8. However, in either of the cases, the best solution and worst solution were less than 0.1% and 1.0% respectively away from the optimal.

A comparative study has been performed with Stochastic Ranking (SR) method presented in [11] and the results are presented in Table IV. The proposed technique outperforms the SR method of handling constraints in three problems (F-6, F-8 and F-10), when compared with respect to the best solution. It matches the results of SR method in five problems (F-1, F-2, F-4, F-7 and F-9). However, when compared with respect to the mean and worst solution, the proposed technique is considerably better than the SR method. The comparison with other methods of handling constraint was performed in [11] and it was found that SR method is better than they are. Therefore, the proposed hybrid technique has better performance than the known techniques of handling constraints.

Further, a comparison with Co-Evolutionary (CE) method proposed in [14], is presented in Table V. The results for four problems (F-1, F-5, F-7 and F-8) have been reported in [14] for 20 independent runs. The proposed technique performed considerably better for problem F-8 as the worst-case result is better than the best of CE. It matched the results for problem F-1. In case of problem F-7, the proposed technique provided better worst-case solution. The proposed technique was marginally worse than CE model. However, it can be improved with modifications in Evolutionary Algorithm that was simple for the purpose of this study.

TABLE III
EXPERIMENTAL RESULTS

Fn #	Best	Median	Mean	St. Dev.	Worst
F-1	-15.000	-15.000	-15.000	0.0	-15.000
F-2	-30665.539	-30665.54	-30665.539	0.0	-30665.539
F-3	5126.498	5126.604	5127.235	1.741	5135.928
F-4	-6961.814	-6961.814	-6961.814	0.0	-6961.814
F-5	24.319	24.408	24.410	0.054	24.541
F-6	0.095979	0.095979	0.095979	0.0	0.095979
F-7	680.630	680.643	680.646	0.009	680.667
F-8	7049.424	7071.258	7075.022	19.556	7111.849
F-9	0.750	0.750	0.750	0. 0	0.750
F-10	0.05395	0.05578	0.05664	0.0031	0.07601

A comparative study was also performed to understand the working of SWAP function. It was achieved by analyzing the results obtained from independent runs of the proposed algorithm WITH (W) and WITHOUT (WO) use of SWAP (S) function. Table VI presents the results of the study. The comparison has been made not only on Best,

Median, Mean, Worst solutions and standard deviations but also on the percentage of feasible solutions found during the experimental run. The first column of Table VI also indicates the percentage of feasible solutions, ρ(%), obtained randomly from an earlier study [15]. The WOS version was able to find optimum solution for F-1, F-2, F-4 and F-6. However, it took almost thrice as many iterations as were required by WS version. The WOS version found poorer feasible solutions for F-5, F-7, F-8 and F-10. Further, it was not able to find even a single feasible solution for F-3 and F-9.

The percentage of feasible solutions in Feasibility Rules part (FR), ρ(%)-FR, is higher for WOS version in comparison to ρ(%)-FR WS version for all problems but F-3, F-6 and F-9 as the focus is on finding the feasible solutions. Even then, It was not able to find any feasible solution for F-3 and F-9 as their search space is highly constrained, which is shown by ρ(%).

The ρ(%)-FR is much worse in WS version in most of the problems than the WOS version. In spite of this the WS version is not just "wandering" through the search space and is able to find at least as good and many a times better solutions than the WOS version. This is because in the early part of the search, the solutions found by the Adaptive Penalty part (AP) could contain solutions with large infeasibility and this would promote exploration of the search space by the FR population, when the former are pushed into the latter. However, as the search progresses, the best solutions in the AP population would be closer to the boundary between feasible and infeasible solutions because the constraint violation would be smaller. This promote search near the boundary, which in fact is the goal of any EA because of the expectation that the optimal solution would be located near boundary.

The percentage of feasible solutions in Adaptive Penalty part ρ(%)-AP is better for WS version as swapping is help in directing the search intelligently towards feasible regions which can provide better solutions.

It was found that WOS version's overall performance was poor. This is because the EA's implementation is indeed rather simple. Therefore, the better performance of WS version is due to the strengths of the proposed constraint handling technique. This is because the two constraints handling techniques, which are hybridized, are 'complimentary' and the swapping improves their search power.

V. Conclusions and Future Work

Effective Constraints handling in Evolutionary algorithm is a challenge worth reckoning. A new constraint handling technique based on hybridization of feasibility rules and adaptive penalty factor that utilizes advantages of parallel Co-evolution is proposed. Authors are not aware of any other such attempt of hybridizing two different constraints handling techniques in the literature. The method is at once parameter free, simple in implementation and effective.

Preliminary results are encouraging and motivate further studies and analysis including performance of the proposed constraint handling technique when coupled with newer and more powerful evolutionary searchers rather than the rudimentary one used in this study. Development of an improved SWAP function and adaptive penalty factor part of the proposed constraint handling technique and application of the proposed technique to solve real life problems is being pursued.

Appendix A

Formulation of benchmark problems [11]

- Problem F-1 is same as g01 in [11].
- Problem F-2 is same as g04 in [11].
- Problem F-3 is same as g05 in [11].
- Problem F-4 is same as g06 in [11].
- Problem F-5 is same as g07 in [11].
- Problem F-6 is same as g08 in [11].
- Problem F-7 is same as g09 in [11].
- Problem F-8 is same as g10 in [11].
- Problem F-9 is same as g11 in [11].
- Problem F-10 is same as g13 in [11].

Acknowledgment

We are extremely grateful to Most Revered Dr. P. S. Satasangi Sahab, Chairman Advisory Committee on Education, Dayalbagh, for continued guidance and support in our every endeavor.

References

[1] C.H. Lee, K.H. Park and J.H. Kim, "Hybrid Parallel Evolutionary Algorithms for constrained optimization utilizing PC Clustering", *Proc. Congress on Evolutionary Computation (2001)*, vol.2, pp. 1436-1441.

[2] C.A.C. Coello, "Theoretical and numerical constraint handling techniques used with evolutionary algorithms: a survey of the state of art", *J. Comput. Methods Appl. Mech. Engrg.*, vol. 191, pp. 1245-1287, 2002.

[3] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, 3rd ed. Berlin: Springer-Verlag, 1996.

[4] Z. Michalewicz and G. Nazhiyath, "Genocop III: A co-evolutionary algorithm for numerical optimization problems with nonlinear constraints," in *Proc. IEEE Int. Conf. Evolutionary Computation*. Piscataway, NJ: IEEE Press, 1995, pp. 647–651.

[5] P. Chootinan, A. Chen, "Constraint handling in genetic algorithms using a gradient-based repair method", *Computers and OR*, vol. 33, pp. 2263-2281, 2006.

[6] Z. Michalewicz "Genetic algorithms, numerical optimization, and constraints," in *Proc. 6th Int. Conf. Genetic Algorithms* L. J. Eshelman, Ed. Los Altos, CA: Morgan Kaufmann, 1995, pp. 151–158.

[7] A.E. Smith, D.W. Coit, "Constraint handling techniques – penalty functions", in: T. B€ack, D.B. Fogel, Z. Michalewicz (Eds.),

Handbook of Evolutionary Computation, Oxford University Press and Institute of Physics Publishing, Oxford, 1997(Chapter C 5.2).
[8] J. H. Kim and H.Myung, "Evolutionary Programming Techniques for Constrained Optimization Problems", *IEEE Trans. Evol. Comput.,* vol. 1, no. 2, pp. 129-140, JULY 1997.
[9] C.A.C. Coello and E.F. Montes, "Constraint-handling in genetic algorithms through the use of dominance-based tournament selection", *J. Adv. Engrg. Informatics*, vol. 16, pp. 192-203, 2002.
[10] P.D. Surry, N.J. Radcliffe, "The COMOGA method: Constrained optimisation by multiobjective genetic algorithms", *J. Control Cybern.*, vol. 26, no. 3, 1997.
[11] T.P. Runarsson and X. Yao, "Stochastic Ranking for constrained evolutionary optimization" , *IEEE Transl.Evol. Comput.*, vol. 4, no. 4, pp. 284-294, Sep. 2000.
[12] K. Deb, "An efficient constraint handling method for genetic algorithms", *Comput. Methods Appl. Mech. Eng.*, vol. 186, no. 2/4, pp. 311-338, Jun. 2000.
[13] G. Harik, E. Cantu-Paz, D.E. Goldberg and B.L. Miller, "The gambler's ruin problem, genetic algorithms and the sizing of populations", *Proc. IEEE International Conf. Evol. Comp.*, pp. 7-12, 1997.
[14] M.J. Tahk and B.C. Sun, "Coevolutionary Augmented Lagrangian Methods for Constrained Optimization", *IEEE Transl.Evol. Comput.*, vol. 4, no. 2, pp. 114-124, Jul. 2000.
[15] S. Koziel and Z. Michalewicz, "Evolutionary algorithms, homomorphous mappings and constrained parameter optimization", *IEEE Transl.Evol. Comput.*, vol. 7, no. 1, pp. 19-44, 1999.

TABLE IV
COMPARATIVE STUDY WITH PROPOSED ALGORITHM AND STOCHASTIC RANKING

Fn #	Best Results		Mean Results		Worst Results	
	H2PCE	SR	H2PCE	SR	H2PCE	SR
F-1	-15.000	-15.000	-15.000	-15.000	-15.000	-15.000
F-2	-30665.539	-30665.539	-30665.539	-30665.539	-30665.539	-30655.539
F-3	5126.498	5126.497	5127.235	5128.881	5135.928	5142.472
F-4	-6961.814	-6961.814	-6961.814	-6875.940	-6961.814	-6350.262
F-5	24.319	24.307	24.410	24.374	24.591	24.642
F-6	0.095979	0.095825	0.095979	0.095825	0.095979	0.095825
F-7	680.630	680.630	680.646	680.656	680.667	680.763
F-8	7049.42	7054.316	7075.02	7559.192	7151.85	8835.655
F-9	0.750	0.750	0.750	0.750	0.750	0.750
F-10	0.053950	0.053957	0.05664	0.057006	0.07601	0.216915

TABLE V
COMPARATIVE STUDY WITH CO-EVOLUTIONARY ALGORITHM (20 RUNS)

Problems	F-1	F-5	F-7	F-8
Optimal	-15.000	24.306	680.630	7049.331
H2PCE				
Best	-15.000	24.319	680.630	7049.424
Median	-15.000	24.408	680.640	7071.258
Worst	-15.000	24.541	680.648	7111.849
CE				
Best	-15.000	24.312	680.630	7126.113
Median	-15.000	24.353	680.633	7513.907
Worst	-15.000	24.516	680.659	11561.223

TABLE VI
COMPARATIVE STUDY OF PROPOSED ALGORITHM WITH SWAP AND WITHOUT SWAP

Fn # □(%)[15]	Swap	Best	Median	Mean	Worst	St. Dev.	□(%)-FR	□(%)-AP
F-1	Without	-15.000	-15.000	-15.000	-15.000	0.0	61.930	0.0
(0.011)	With	-15.000	-15.000	-15.000	-15.000	0.0	22.833	0.137
F-2	Without	-30665.539	-30665.539	-30665.539	-30665.539	0.0	56.431	0.0
(52.12)	With	-30665.539	-30665.539	-30665.539	-30665.539	0.0	10.878	6.245
F-3	Without	NA	NA	NA	NA	NA	0.0	0.0
(0.000)	With	5126.498	5126.604	5127.235	5135.928	1.741	0.074	0.004
F-4	Without	-6961.814	-6961.814	-6961.814	-6961.814	0.0	21.532	0.243
(0.0066)	With	-6961.814	-6961.814	-6961.814	-6961.814	0.0	6.204	0.327
F-5	Without	24.358	24.478	24.531	24.900	0.193	41.949	0.003
(0.0003)	With	24.319	24.408	24.410	24.541	0.054	9.797	1.127
F-6	Without	0.095979	0.095979	0.095979	0.095979	0.0	51.625	28.852
(0.8560)	With	0.095979	0.095979	0.095979	0.095979	0.0	53.439	50.257
F-7	Without	680.633	680.648	680.652	680.683	0.014	39.949	0.874
(0.5121)	With	680.630	680.643	680.646	680.667	0.009	21.050	1.235
F-8	Without	7098.019	7109.383	7115.460	7177.873	31.255	31.372	0.0
(0.0010)	With	7049.424	7071.258	7075.022	7111.849	19.556	3.324	0.343
F-9	Without	NA	NA	NA	NA	NA	0.0	0.0
(0.000)	With	0.750	0.750	0.750	0.750	0. 0	0.259	0.083
F-10	Without	0.529	0.586	0.654	0.942	0.148	0.042	0.0
(0.000)	With	0.05395	0.05578	0.05664	0.07601	0.0031	0.014	0.0065