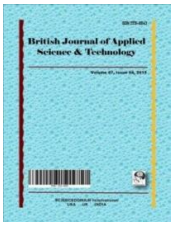

British Journal of Applied Science & Technology
8(3): 324-333, 2015, Article no.BJAST.2015.210
ISSN: 2231-0843

SCIENCEDOMAIN *international*
www.sciencedomain.org

A Combined Genetic Algorithms-local Search Engine (GAs–LCE) in Constrained Nonlinear Programming

Faiz A. El-Qorashy[1], Hossam A. Nabwey[2,3] and A. A. Mousa[1,3*]

[1]Department of Mathematics and Statistic, Faculty of Science, Taif University, Saudi Arabia.
[2]Department of Mathematics, Faculty of Science, Salman Bin Abdulaziz University, Al-Kharj, Saudi Arabia.
[3]Department of Basic Engineering Sciences, Faculty of Engineering, Menoufia University, Egypt.

Authors' contributions

This work was carried out in collaboration between all authors. Author FAEQ designed the study. Author HAN performed the statistical analysis. Author AAM wrote the protocol, and wrote the first draft of the manuscript and managed literature searches. All authors managed the analyses of the study and literature searches and approved the final manuscript.

Article Information

DOI: 10.9734/BJAST/2015/17059
Editor(s):
(1) Wei Wu, Applied Mathematics Department, Dalian University of Technology, China.
Reviewers:
(1) Anonymous, Malaysia.
(2) José Luis López-Bonilla, National Polytechnic Institute, Mexico.
(3) Anonymous, China.
(4) Varun Shukla, Electronics & Communication, Uttar Pradesh Technical University, India.
Complete Peer review History: http://www.sciencedomain.org/review-history.php?iid=1071&id=5&aid=8746

Original Research Article

Received 24th February 2015
Accepted 27th March 2015
Published 10th April 2015

ABSTRACT

Evolutionary optimization provides robust and efficient techniques for solving complex real-world problems. The aim of this paper is to present an enhanced evolutionary algorithm for solving constraint nonlinear programming problems NLPPs, which based on concept of co-evolution and repair algorithm for handling nonlinear constraints. Our proposed approach is made of two phases, firstly, phase I is a classical genetic algorithm, which based on the ideas of repair strategy and co-evolution. Secondly in phase II, Based on the k-means cluster algorithm, the search space is shrunk after phase I to the generated rectangular-atom with highly rate and concentrating the optimal solution region, so local search techniques will implemented in order to get more accurate optimal solution. Finally, the results of various experimental studies using a suite of benchmark functions have demonstrated the superiority of the proposed algorithm to finding the global optimal solution for constraint nonlinear programming problems.

**Corresponding author: E-mail: a_mousa15@yahoo.com;*

Keywords: Nonlinear programming; genetic algorithms; local search; k-means.

1. INTRODUCTION

Evolutionary algorithms have received a lot of attention regarding their potential as optimization approaches for complex numerical optimization problems [1,2]. However, they have not made a significant breakthrough in the area of constraint NLP [3] due to the fact that they have not addressed the issue of handling nonlinear constraints, also there is a fact that evolutionary algorithms may find only near-optimal solutions. On the other hand, many optimization problems involve inequality and/or equality constraints are thus posed as constrained optimization problems [4,5]. In trying to solve constrained problems using evolutionary algorithms or classical optimization techniques, penalty function methods have been the most popular approach [6], because of their simplicity and ease of implementation. However, since the penalty approaches are generic and applicable to any type of nonlinear constraint, their performance is not always satisfactory. Thus, several methods for handling unfeasible solutions have emerged recently [7].

Evolutionary approaches [8-12] are powerful computing systems to deal with large-scale problems. Michalewicz et al. [8] present an efficient evolutionary Algorithms for Constrained Parameter Optimization Problems, also they present modified version called “Genocop III” in [9]. AL-Oraby et al. [10] introduce Hybrid optimization technique coupling an evolutionary algorithm and chaotic local search. In [11] Al-Thobaiti et al. Integrate an optimization technique coupling an evolutionary algorithm and local search scheme. Osman et al. [12] Combine Genetic Algorithm with Fuzzy Logic Controller for nonlinear programming. However, they require time consuming, and they are very poor in terms of convergence performance. If the initial population has both good and worst individuals, there is a chance of selecting the worst individuals in processing, this may reduce the convergence of the algorithm. Hence it is necessary to provide good chromosomes in the search space [13].

On the other hand, local search strategy can converge quickly to local minima and get stuck in a local optimum solution, which is far away from the global optimal. The integration of global and local search procedures should offer the advantages of both optimization systems while offsetting their disadvantages. This paper presents a combined genetic algorithms-local search engine for constrained NLPPs.

This paper is organized as follows, in section 2, we formulate the nonlinear programming problems. Section 3 addresses the problem of optimization using genetic algorithms. In section 4, we present the combined (genetic algorithms_ local search engine) for nonlinear programming. Section 5 deals with the numerical simulation of the proposed algorithm to different benchmark problems and the discussion of the obtained results are followed in section 6.

2. NONLINEAR PROGRAMMING PROBLEMS (NLPPS)

The general NLP for continuous variables [14] is to find $\bar{x}$ so as to

$$\text{Min } f(\bar{x}),\ \bar{x}=(x_1,...,x_n)\in R^n, \tag{1}$$

Where $\bar{x}\in F\subseteq S$. The set $S\subseteq R^n$ defines the search space, and the set $F\subseteq S$ defines a feasible part of the search space. Usually, the search space S is defined as n-dimensional rectangular-atom in R^n (domains of variables defined as lower and upper bounds): $left(i)\le x_i\le right(i),\ 1\le i\le n$, whereas the feasible set F is defined by the search space S and an additional set of constraints:

$$g_j(\bar{x})\le 0,\ \text{for}\quad j=1,.....,m \tag{2}$$

Thus NLPP can be defined as follows:

$$\begin{aligned}&\text{NLPP}: \text{Max } f(x)\\ &\quad \text{S.t}\\ &\quad F=\{x\in R^n \mid g_i(x)\le 0,\ i=1,2,...,k \text{ and } h_j(x)=0\, j=k+1,...,m\}\\ &\quad S=\{x\in R^n \mid l(x_i)\le x_i\le u(x_i),\quad i=1,2,....,n\}\end{aligned} \tag{3}$$

At any point $\bar{x} \in F$, the constraint $g_k(\cdot)$ satisfies $g_k(\bar{x}) = 0$ are called "active constraints" at $\bar{x}$. By extension, equality constraints $h_j(\cdot)$ are called active at all feasible points. All nonlinear equations $h_j(x) = 0 \ \ (for\ j = k+1,...,m)$ are replaced by pair of inequalities: $-\psi \le h_j(x) \le \psi$ with additional parameter (ψ) to define the precision of the system [15], so we deal only with nonlinear inequalities.

$$\begin{aligned} &\text{NLPP}: \ \text{Max}\ f(x) \\ &\quad \text{S.t} \\ &\quad F = \{x \in R^n \mid g_i(x) \le 0,\ i = 1,2,...,m\} \\ &\quad S = \{x \in R^n \mid l(x_i) \le x_i \le u(x_i),\quad i = 1,2,....,n\} \end{aligned} \tag{4}$$

Any evolutionary algorithm applied to any particular problem should address the issue of handling unfeasible solutions. In general, a search space S consists of two disjoint subsets of feasible subspace F and unfeasible subspace, in general these two spaces need not be convex and they need not be connected (e.g., as in the case in Fig. 1 (Taken from [12]) where feasible part F of the search space consist of two disjoined subsets).

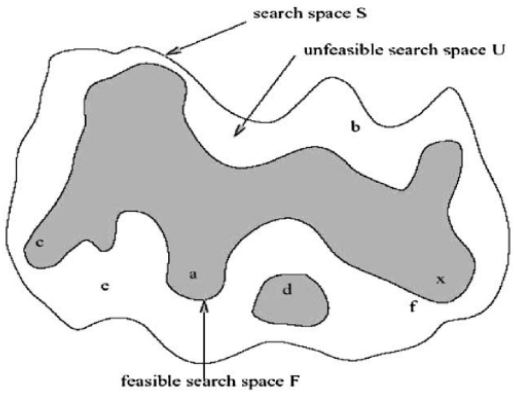

Fig. 1. A Search space and its feasible part

3. THE GENETIC ALGORITHMS GAS

Genetic algorithms, invented by John Holland [2] in the early 1970s, as a heuristic global search algorithm, that mimics the metaphor of natural biological evaluation. GAs operates on a population of candidate individual, which encoded to finite string called chromosome. In order to obtain optimality, the individual exchanges information by using some operators borrowed from natural biological to produce the better solution. GAs differs from other optimization algorithms and global search procedures in four ways [1].

(1) GAs work with a coding of the decision variables, not the decision variables themselves. Therefore GAs can easily handle the integer, mixed, discontinuous, and discrete systems.

(2) GAs search from a population of candidate solutions, not a single solutions. Therefore GAs can provide a globally optimal solution.

(3) GAs uses only the objective function information, not any auxiliary knowledge. Therefore GAs can deal with the discontinuous, non-smooth and non-differentiable systems which are actually existed in a practical engineering optimization problem.

(4) GAs use probabilistic rules, not deterministic rules. Fig. 2 illustrates flowchart of a simple EAs.

4. EVOLUTIONARY ALGORITHM BASED ON LOCAL SEARCH ENGINE

The main idea is based on the ideas of co-evolution and repair strategy (repair unfeasible individual). The proposed evolutionary algorithm combines concept of co-evolution, repairing procedure, and elitist scheme to produce an enhanced algorithm. Repairing procedure [15], repairs the unfeasible points to satisfy the constraints. Elitist strategy is used to produce a faster convergence to the optimal solution of the problem by ensuring that the individual of the most highly fit member of the population are passed on to the next generation. The working procedure of the proposed algorithm is described in the following manner:

4.1 Phase I: Genetic Algorithm

4.1.1 Solution representation

The proposed algorithm uses a floating point representation for potential solutions of the problem. Each generation contains both feasible individuals, and unfeasible individuals and we distinguish between them using flag pointer assigned to each individuals. Fig. 3 shows the structure of candidate individual, which contain a flag pointer assigned to each individuals, and it's fitness value.

4.1.2 Initialization stage

The populations are initialized randomly in the 1st generation, satisfying the constraint, i.e., each individual lie on the search space S, while elitist individual is initialized by zero.

4.1.3 Initial feasible point

The algorithm needs to allocate at least one feasible point (i.e., reference feasible point) to complete the algorithm procedure. If the proposed algorithm has difficulties in locating such an initial feasible point, the algorithm applies one of the following two ways. First, double the number of trials to obtain feasible point. Secondly, increase the value of the precision parameter ψ temporally (i.e., enlarge feasible space temporally). The interested reader is referred to [15] for further information.

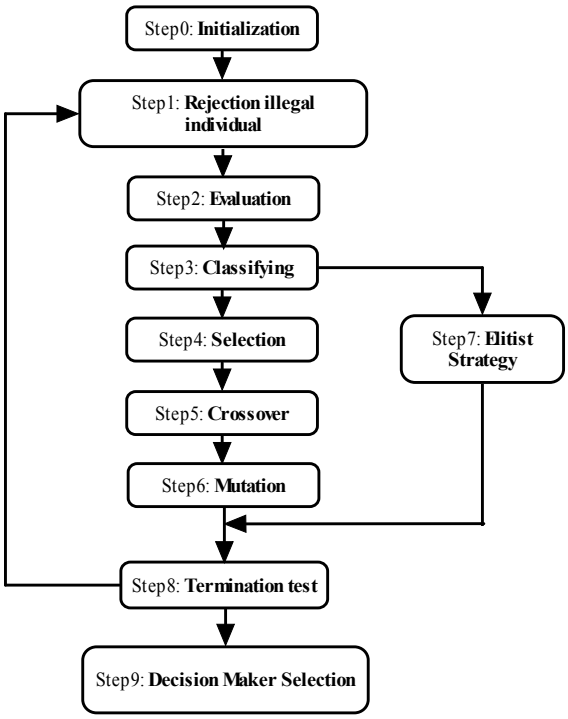

Fig. 2. Flowchart of EAs

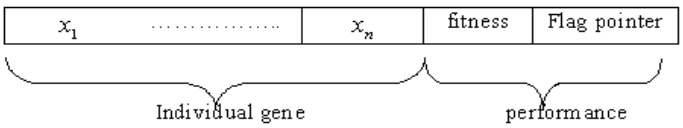

Fig. 3. Individual's structure

4.1.4 Repairing unfeasible individuals

The idea of this technique is to distinguish any feasible individual in a population from those that are unfeasible. The proposed approach coevolves unfeasible individuals until they become feasible individuals, in the way such that, feasible individuals (Ψ) are generated on a segment defined by two points, feasible point $\xi \in \mathrm{F}$ and any unfeasible point $\varsigma \notin \mathrm{F}$. The interested reader is referred to [15] for further information.

4.1.5 Elitist strategy

The elitist individual represents the fittest individual of the population. The use of elitist individual, guarantees that the best fitness individual never loses its fittest (Towards the end of the process).

4.1.6 Evolution process stage

The proposed algorithm uses the objective function to evaluate the fitness functions for each individual. The algorithm applies binary tournament selection procedure / roulette wheel selection to select the new population.

4.1.7 Stopping rule

The proposed algorithm is terminated for either one of the following conditions is satisfied:

- The maximum number of predetermined generations is achieved.
- When the genotypes of the population converge, convergences of the genotype structure occur when all bit positions in all strings are identical.

4.2 Phase 2: Local Search Engine

Upon termination of phase I, we have a set of points, which called the population, K-means cluster was implemented to reduce the size of population to a manageable size called initial set (IS), that guarantee uniform distribution of the data points. We start phase 2, with a set of points IS "initial set", then the following procedure are applied.

Step 1: For each point $x^e \in IS$, generate its own rectangular-atom as follows:

1- Range Initialization: for each decision variable x_i , we compute and sort from smallest to highest the different values it takes in the population set. Then, for each decision variable, we have a set of " $rang_i$ " values and combining all these sets we have a non-uniform grid in decision variable space.

2- Compute rectangular-atom boundaries: we compute "rectangular-atom" centered in each position of the initial set IS. To build a rectangular-atom associated to a point $x^e \in IS$ we compute the following upper and lower bounds for each decision variable x_i as follows :

- Lower Bound ^{i} : Middle point between x_i^e and the previous value in the set $rang_i$
- Upper Bound ^{i} : Middle point between x_i^e and the following value in the set $rang_i$
- If there are no pervious or subsequent values in $rang_i$, we consider the absolute lower or upper bound of variable ^{i} . This setting lets the method to explore close to the feasible set boundaries.

Fig. 4 shows the initial set of solutions and it's generated rectangular-atom.

3- Applying local search scheme inside each rectangular-atom to the off spring in order to Generate new Offspring: replacing the resultant offspring if the new offspring is better than the old one.

5. EXPERIMENTAL VERIFICATION

To validate our proposed algorithm, we present five nonlinear constraint problems [4,12,16-18] solved using four evolutionary algorithms [12,15,19] by Intel Core i5 processors and implemented in MATLAB 12.

P01 [12,17]

$$Min \quad f(x) = 5.3578547x_3^2 + 0.8356891x_1x_5$$
$$g_1(x) = 85.334407 + 0.0056858x_2x_5 + 0.00026x_1x_4 + 0.0022053x_3x_5$$
$$g_2(x) = 80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2 + 0.0021813x_3^2$$
$$g_3(x) = 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4$$
$$0 \le g_1(x) \le 92$$
$$90 \le g_2(x) \le 110$$
$$20 \le g_3(x) \le 25$$
$$78 \le x_1 \le 102$$
$$33 \le x_2 \le 45$$
$$27 \le x_3 \le 45$$
$$27 \le x_4 \le 45$$
$$27 \le x_5 \le 45$$

P02 [12,18]

$$Min \quad f(x) = x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 + 4(x_4 - 5)^2 + (x_5 - 3)^2 + 2(x_6 - 1)^2 + 5x_7^2 + 7(x_8 - 11)^2 + 2(x_9 - 11)^2 + (x_{10} - 7)^2 + 45,$$
$$g_1(x) = 105 - 4x_1 - 5x_2 + 3x_7 - 9x_8 \ge 0,$$
$$g_2(x) = -10x_1 + 8x_2 + 17x_7 - 2x_8 \ge 0,$$
$$g_3(x) = 8x_1 - 2x_2 + 5x_9 + 2x_{10} + 12 \ge 0,$$
$$g_4(x) = -3(x_1 - 2)^2 - 4(x_2 - 3)^2 - 2x_3^2 + 7x_4 + 120 \ge 0,$$
$$g_5(x) = -5x_1^2 - 8x_2 - (x_3 - 6)^2 + 2x_4 + 40 \ge 0,$$
$$g_6(x) = -x_1^2 - 2(x_2 - 2)^2 + 2x_1x_2 - 14x_5 + 6x_6 \ge 0,$$
$$g_7(x) = -0.5(x_1 - 8)^2 - 2(x_2 - 4)^2 - 3x_5^2 + x_6 + 30 \ge 0,$$
$$g_8(x) = 3x_1 - 6x_2 - 12(x_9 - 8)^2 + 7x_{10} \ge 0,$$
$$-10 \le x_i \le 10, i = 1,...10.$$

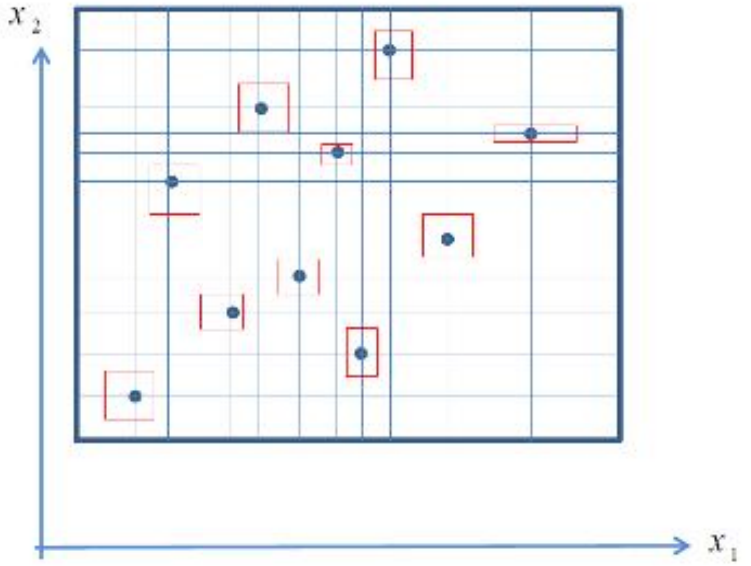

Fig. 4. The initial set of solutions and it's generated rectangular-atoms

P03 [16]

$$Min \quad f(x) = 5\sum_{i=1}^{4} x_i - 5\sum_{i=1}^{4} x^2{}_i - \sum_{i=5}^{13} x_i$$

$$g_1(x) = 2x_1 + 2x_2 + x_{10} + x_{11} - 10 \leq 0$$

$$g_2(x) = 2x_1 + 2x_2 + x_{10} + x_{12} - 10 \leq 0$$

$$g_3(x) = 2x_2 + 2x_3 + x_{11} + x_{12} - 10 \leq 0$$

$$g_4(x) = -8x_1 + x_{10} \leq 0$$

$$g_5(x) = -8x_2 + x_{11} \leq 0,$$

$$g_6(x) = -8x_3 + x_{12} \leq 0,$$

$$g_7(x) = -2x_4 - x_5 + x_{10} \leq 0,$$

$$g_8(x) = -2x_6 - x_7 + x_{10} \leq 0,$$

$$g_9(x) = -2x_8 - x_9 + x_{12} \leq 0,$$

$$0 \leq x_i \leq 1, i = 1,...9, 0 \leq x_i \leq 100, i = 10,11,12, 0 \leq x_{13} \leq 1.$$

P04 [16]

$$Max \; f(x) = \left| \frac{\sum_{i=1}^{n} \cos^4 x_i - \prod_{i=1}^{n} \cos^2 x_i}{\sqrt{\sum_{i=1}^{n} ix_i^2}} \right|$$

$$g_1(x) = 0.75 - \prod_{i=1}^{n} x_i \leq 0;$$

$$g_2(x) = \sum_{i=1}^{n} x_i - 7.5n \leq 0;$$

$$0 \leq x_i \leq 10, i = 1,...,20.$$

P05 [4,12]

$$Min \quad f_w(\vec{x}) = 1.10471h^2 l + .04811tb(14.0 + l)$$

$$g_1(\vec{x}) = 13600 - r(\vec{x}) \geq 0,$$

$$g_2(\vec{x}) = 30000 - s(\vec{x}) \geq 0,$$

$$g_3(\vec{x}) = b - h \geq 0,$$

$$g_4(\vec{x}) = p_c(\vec{x}) - 6000 \geq 0,$$

$$g_5(\vec{x}) = 0.25 - d(\vec{x}) \geq 0,$$

$$0.125 \leq h \leq 10$$

$$0.1 \leq l, t, b \leq 10.$$

where,

$$r(\vec{x}) = \sqrt{(r'(\vec{x}))^2 + (r''(\vec{x}))^2 + l(r'(\vec{x}) r''(\vec{x}))} / \sqrt{0.25 * (l^2 + (h + t)^2)}$$

$$s(\vec{x}) = \frac{504000}{t^2 b},$$

$$p_c(\vec{x}) = 64746.022(1 - 0.0282346t)tb^3,$$

$$d(\vec{x}) = \frac{2.1952}{t^3 b},$$

$$r'(\vec{x}) = \frac{6000}{\sqrt{2}hl},$$

$$r''(\vec{x}) = \frac{6000(14.0 + 0.5l)\sqrt{0.25 * (l^2 + (h + t)^2)}}{2\{0.707hl(l^2/12 + 0.25(h + t)^2)\}}.$$

Table 1 summarize the results for four evolutionary algorithms, where we list the best, worst , mean and standard deviations after 20 independent runs for each test problem, the results obtained by the proposed algorithm are better than the corresponding ones obtained from our proposed algorithm.

Table 1. Statistical analysis of the proposed algorithm versus other evolutionary algorithms

Function	Optimal	Status	Evolutionary algorithms			
			Proposed	**[12]**	**[15]**	**[19]**
P01	-30665.55	**Best**	-30665.55	-30665.51	-30665.35	-30665.53
		Mean	-30666.31	-30666.26	-30666.31	-30666.53
		Worst	-30665.53	-30665.33	-30665.23	-30665.53
		St. Dev.	5.1E-02	4.3E-04	4.2E-01	5.1E-09
P02	24.8641	**Best**	24.8641	24.9631	24.9641	24.3062
		Mean	24.6621	24.7691	24.8621	24.4312
		Worst	24.6521	24.6601	24.6532	24.6721
		St. Dev.	3.3E-02	4.3E-02	4.2E-03	5.1E-01
P03	-15	**Best**	-15	-15	-15	-15
		Mean	-15	-15	-15	-14.492
		Worst	-15	-15	-15	-14.354
		St. Dev.	0	0	0	9E-01
P04	-0.80325185	**Best**	-0.80325	-0.81325	-0.83215	-0.80315
		Mean	-0.81325	-0.81212	-0.84325	-0.81325
		Worst	-0.82325	-0.81532	-0.86312	-0.82335
		St. Dev.	5.3E-02	4.6E-04	4.7E-02	5.7E-03
P05	2.381021	Best	2.381021	2.38302	2.381054	2.38104
		Mean	2.392041	2.38405	2.381092	2.38710
		Worst	2.393061	2.38602	2.381063	2.3900
		St. Dev.	9.3E-06	5.3E-04	3.2E-03	3.2E-05

Figs. 5-9 illustrate the convergence analysis of the proposed algorithm for these five problems.

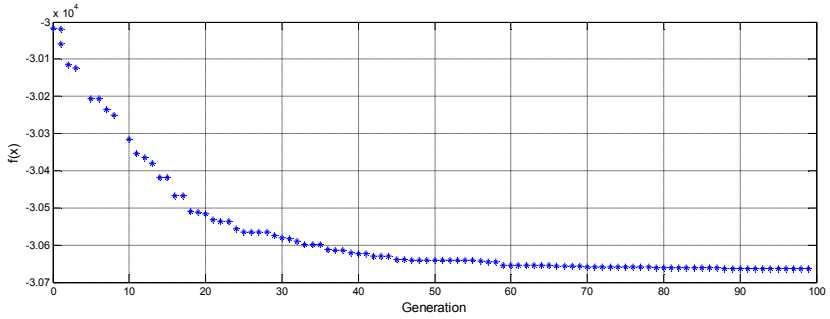

Fig. 5. Convergence analysis for problem P01

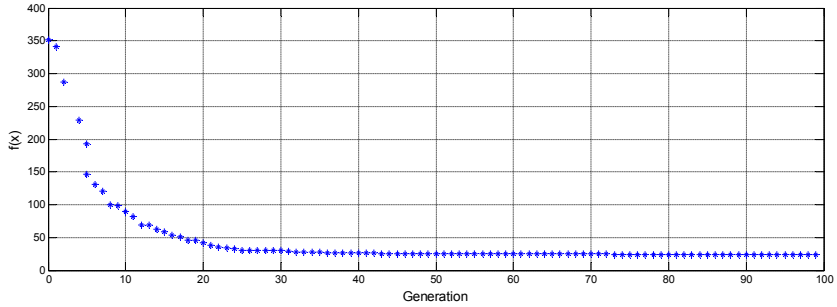

Fig. 6. Convergence analysis for problem P02

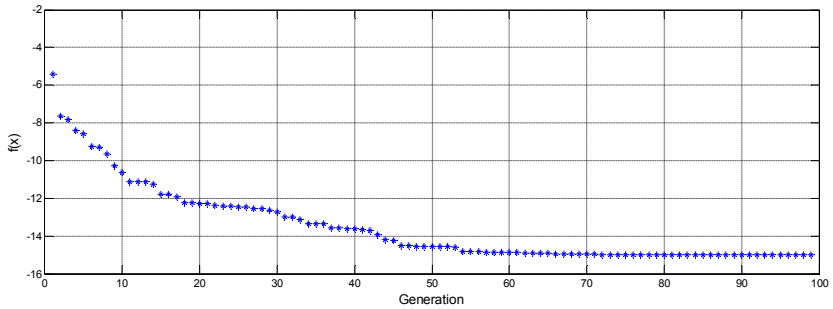

Fig. 7. Convergence analysis for problem P03

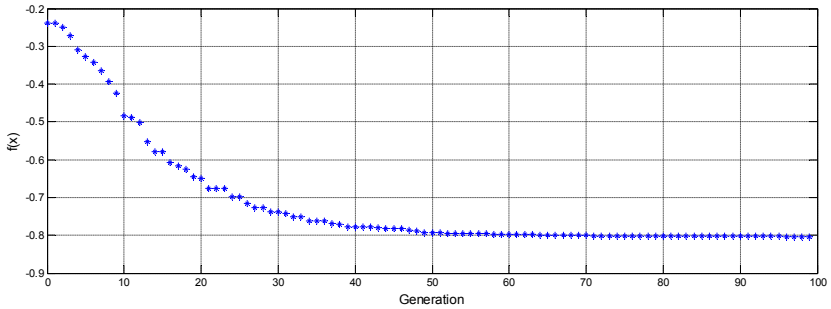

Fig. 8. Convergence analysis for problem P04

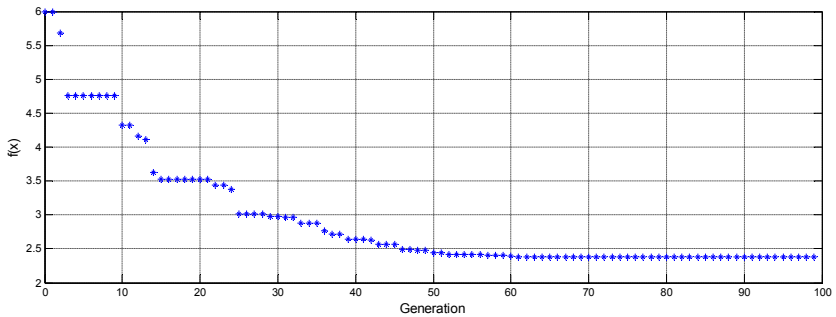

Fig. 9. Convergence analysis for problem P05

6. CONCLUSION

This paper presents a combined genetic algorithm-local search engine to solve constrained NLP. This algorithm is made of a classical genetic algorithm based on the ideas of repair strategy combined with a local search engine. Based on the k-means cluster algorithm, the search space is shrunked after phase I to the generated rectangular-atoms with highly rate and concentrating the optimal solution region. So the process converges rapidly to the final solution. Also, the proposed algorithm allowed us to get closer to the target than the previous evolutionary algorithm based techniques. In brief, this algorithm in phase I, has capability to adjust its starting population for the second phase, to avoid local minimum and to obtain more accuracies that classical evolutionary algorithms fail to obtain. Several examples allowed us to compare our results with those found in the literature. The numerical analysis shows that our combined system turns out to be very efficient in accuracy of the final solution.

COMPETING INTERESTS

Authors have declared that no competing interests exist.

REFERENCES

1. Michalewicz Z. Genetic Algorithms + Data Structures = Evolution Programs. Springer-Verlag, 3rd Edition; 1996.
2. Goldberg DE. Genetic algorithms in search, optimization and machine learning. Addison Wesley Publishing Company; 1989.
3. Michalewicz Z. A survey of constraint handling techniques in evolutionary computation methods. Proceeding of the 4th Annual Conference on Evolutionary Programming, pp135-155, MIT Press, Cambridge, MA; 1995.
4. Deb K. Optimal design of a welded beam structure via genetic algorithms. ALAA Journal. 1991;29(11):2013-2015.
5. Michalewicz Z. Genetic algorithms, numerical optimization, and constraints. In: L. Eshelman, ed., proceedings of the sixth international conference on genetic algorithms (Morgan Kauffman, San Mateo). 1995;1-158.
6. Atsushi Kato, Hiroshi Yabe, Hiroshi Yamashita. An interior point method with a primal–dual quadratic barrier penalty function for nonlinear semidefinite programming. Journal of Computational and Applied Mathematics. 2015;275:148–161.
7. Jinn-Tsong Tsai. Improved differential evolution algorithm for nonlinear programming and engineering design problems. Neurocomputing. 2015;148: 628–640.
8. Michalewicz Z, Schoenauer. Evolutionary Algorithms for Constrained Parameter Optimization Problems. Evolutionary Computation. 1996;4(1):1-32.
9. Michalewicz Z, Nazhiyath G. Genocop III: A Co-evolutionary Algorithm for Numerical Optimization Problems and with nonlinear constraints in D .B. Fogel (Ed), proceedings of the second IEEE International Conference on Evolutionary Computation. 19945;647-651.
10. AL-Oraby H, Kotb A. Kotb, Mousa AA. Hybrid optimization technique coupling an evolutionary algorithm and chaotic local search. Journal of Global Research in Mathematical Archives. 2014;2(1):1-10.
11. Al-Thobaiti M, Kotb A. Kotb, Mousa AA. Integrating optimization technique coupling an evolutionary algorithm and local search scheme. Journal of Global Research in Mathematical Archives. 2014;2(1):11-21.
12. Osman MMS, Abo-Sinna MA, Mousa AA. A combined genetic algorithm-fuzzy logic controller for nonlinear programming. Journal of Applied Mathematics & Computation (AMC). 2005;170:821-840.
13. Preuss M, Beielstein TB. Sequential parmeter optimization applied to self adaptation for binary coded evolutionary algorithms. Journal of Parameter Settings and Evolutionary Algorithms, Springer, Berlin, Germany. 2007;91-119.
14. Bazaraa MS. Nonlinear Programming: Theory and Algorithms, New York, John Willy & Sons; 1979.
15. Osman MS, Abo-Sinna MA, Mousa AA. A solution to the optimal power flow using genetic algorithm. Applied Mathematics and Computation. 2004;155:391-405.
16. Floundas C, Pardalos P. A collection of test problems for constrained global optimization, lecture Notes in computer sciences. Berlin: Sprringer-Verlag; 1987.
17. Himmelblau D. Applied nonlinear programming, New York, McGraw-hill, IEEE press; 1972.
18. Hock W, Schittkowski K. Test examples for nonlinear programming codes, berlin, springer-verlag, lecture notes on Econ. And Math. Syst; 1981.
19. Waiel F. Abd El-Wahed, Mousa AA, El-Shorbagy MA. Integrating particle swarm optimization with genetic algorithms for solving nonlinear optimization problems. Journal of Computational and Applied Mathematics. 2011;235:1446–1453.

Peer-review history:
The peer review history for this paper can be accessed here:
http://www.sciencedomain.org/review-history.php?iid=1071&id=5&aid=8746