# A NEW FITNESS ASSIGNMENT AND PARENT SELECTION STRATEGY WITHIN AN EVOLUTIONARY ALGORITHM FOR CONSTRAINED OPTIMIZATION PROBLEMS

*Tapabrata Ray*

Senior Research Scientist,
Temasek Laboratories
National University of Singapore,
10 Kent Ridge Crescent,
Singapore 119260.
Email: tsltray@nus.edu.sg

*Poan Choy Ling*

Postgraduate Student,
Singapore MIT Alliance
National University of Singapore,
10 Kent Ridge Crescent,
Singapore 119260.

*Tai Kang*

Fellow, Singapore MIT Alliance
School of Mechanical and Production
Engineering, Nanyang Technological
University, 50 Nanyang Avenue
Singapore 639798.

## ABSTRACT

Evolutionary algorithms (EA) are search strategies that mimic the process of natural evolution. All EA's have two fundamental strategies; a selection and a recombination strategy both of which are known to largely influence the performance of the algorithm. The selection strategy ensures fitter individuals have a greater chance of survival and a greater participation in mating while the recombination strategy aims to inherit meaningful parent properties. In this paper a new fitness assignment scheme and a new parent selection strategy is proposed. The individuals are assigned separate fitness values in the objective and the constraint space unlike most EAs that use a single fitness measure for selection. The parent selection mechanism employed in the algorithm is both elitist and adaptive. The recombination strategy is based on a parent centric operator that explores the neighborhoods of good parents in search for better ones. In this paper we present the results obtained by our algorithm and compare it with the reported results on a suite of six single objective constrained test problems.

## 1. INTRODUCTION

Evolutionary algorithms are a popular choice in attempts to solve real life optimization problems. EAs are particularly attractive as they are applicable to a wide class of problems and can deal with models without simplifying assumptions of linearity or continuity. The success of any evolutionary algorithm can be attributed to two fundamental processes of selection and recombination. In order to perform selection, every individual needs to be assigned a fitness value. This measure of fitness is used to determine whether or not an individual survives and copies itself to the next generation and whether or not it participates in mating.

The fitness values can be assigned to individuals based on either its objective function value or its rank for unconstrained problems. However, the assignment of fitness values to individuals for constrained problems is non trivial as it requires a means to compare infeasible solutions. Existing evolutionary methods use sum of constraint violations, sum of scaled constraint violations, amount of largest constraint violation, weighted and scaled combinations of the above or adaptive weights and scaling factors to assign fitness values to infeasible individuals. These methods of fitness assessment are computationally simple but often require additional inputs or intrinsically define fitness measures that may or may not be desirable.

In this paper, a formal notion of comparison is proposed that forms the basis of the algorithm presented later in Section 2. The algorithm is built upon the following concepts, the details of which are outlined in Section 2.

1. A feasible solution is preferred over an infeasible solution.
2. Between two feasible solutions, one with a better objective function value if preferable over the other.
3. Between two infeasible solutions, one with a lower Pareto Rank based on the Constraint matrix is preferred over the other.
4. The algorithm drives a population towards feasibility first before trying to improve an individuals' objective function value.

## 2. MATHEMATICAL FORMULATION

A single objective constrained optimization problem can be presented as follows:

Minimize

$$f(\mathbf{x}) \quad (1)$$

Subject to

$$g_i(\mathbf{x}) \geq a_i \quad , i = 1,2,\ldots,q \quad (2)$$

$$h_j(\mathbf{x}) = b_j \quad , j = 1,2,\ldots, r \quad (3)$$

where there are $q$ inequality and $r$ equality constraints and $\mathrm{x} = [x_1 \quad x_2 \quad \ldots \quad x_n]$ is the vector of $n$ design variables. In order to handle equality constraints, each equality constraint is replaced by a pair of inequalities of the form $h_j(\boldsymbol{x}) \le b_j + \delta$ and $h_j(\boldsymbol{x}) \ge b_j - \delta$. Thus $r$ equality constraints lead to $2r$ inequalities, and the total number of inequalities for the problem is denoted by $s=q+2r$. For each individual, **c** denotes the constraint satisfaction vector $\mathbf{c} = [c_1 \quad c_2 \quad \ldots \quad c_s]$ where $c_i > 0$ indicates the violation of the $i^{th}$ constraint. The CONSTRAINT matrix for a population of $M$ individuals assumes the form

$$\text{Constraint} = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1s} \\ c_{21} & c_{22} & \cdots & c_{2s} \\ \vdots & \vdots & \ddots & \vdots \\ c_{M1} & c_{M2} & & c_{Ms} \end{bmatrix} \qquad (4)$$

The objective matrix of the population assumes the form

$$\text{Objective} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_M \end{bmatrix} \qquad (5)$$

One can use a Nondominated Sorting to rank the individuals based on the Constraint matrix while a simple sorting can be used rank the individuals based on the Objective matrix. A linear expression of fitness is used in this study where fitness of the $i^{th}$ individual is presented as follows:

$$fitness_i = 1 + MaxRank - Rank(i) \qquad (6)$$

It can be observed from the constraint matrix that when all the individuals in the population are infeasible, the Rank=1 solutions are the best in terms of minimal constraint violation. Whenever there is one or more feasible individuals in the population, the feasible solutions assume the rank of 1. The pseudo code of the algorithm is presented below.

---

*Initialize a Population (P) with M individuals using a Uniform Random Generator and the Variable Bounds;*
*Do {*
- *Compute Objective value the Constraint Satisfaction Vector for each Individual in the P;*
- *Sort and Rank the Individuals of P based on their Objective Function values;*
- *Sort and Rank the Individuals of P based on their Constraint Satisfaction Vector using Non Dominated Sorting;*
- *Compute the Average Objective and Average Constraint Rank;*
- *Count the Number of Feasible Individuals;*
- *Assign Individuals with Objective Rank < Average Objective Rank to Set A;*
- *Assign Individuals with Constraint Rank < Average Constraint Rank to Set B;*
- *Assign Feasible Individuals as Set C;*
- *Compute A∩B and A∩C;*
- *If (Size of Set C = 0): Assign Individuals with Rank Constraint = 1 to Set E;*
- *If (Size of Set C > M/2) and (Size of A∩C > 0): Assign Individuals A∩C to Set E;*
- *If (Size of Set C > M/2) and (Size of A∩C=0): Assign Individuals of C to Set E;*
- *If (Size of Set C <= M/2) and (Size of C>0): Assign Individuals of C to Set E;*
- *Copy Individuals of Set E to the Population for the Next Generation;*
- *Do {*
  - *Sort and Rank the Individuals of Set E based on their Objective Function values;*
  - *Sort and Rank the Individuals of Set E based on their Constraint Satisfaction Vector using Non Dominated Sorting;*
  - *If (Size of C=0): Select Dad using Roulette Wheel based on Constraint Rank of the Individuals of Set E.*
  - *If (Size of C>0) Select Dad using Roulette Wheel based on Objective Rank of the Individuals of Set E.*
  - *If (Size of C>M/2): Select Mum 1 and Mum 2 using Roulette Wheel based on Objective Rank of the Individuals of the Population (P).*
  - *If (Size of C<=M/2): Select Mum 1 and Mum 2 using Roulette Wheel based on Constraint Rank of the Individuals of the Population (P).*
  - *If Mum 1 is Feasible and Mum 2 is not: Partner is Mum 1.*
  - *If Mum 2 is Feasible and Mum 1 is not: Partner is Mum 2.*
  - *If both Mum 1 and Mum 2 are Feasible: Partner is the one with mimimum Objective Rank.*
  - *If both Mum 1 and Mum 2 are Infeasible: Partner is the one with mimimum Constraint Rank.*
  - *Mate Dad with the Partner to generate a Child using a Parent Centric Crossover Operator.*
  - *Copy the Child to the Population of the Next Generation.*
- *} while Population is not Full;*

*} while (termination condition = False)*

---

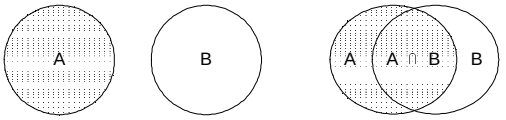
Pseudo code of the parent centric recombination operator.

---

*Select a Parent (P) randomly among parents P1 and P2;*
*Compute the Euclidian distance (D) between P1 and P2 in the parametric space;*
*For i =1 to Number of Variables*
*Generate a random number (R) using a Gaussian distribution with $\mu = 0$ and $\sigma = 1$;*
*C(i) = P(i) +R.D;*

---

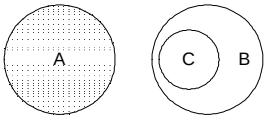
An instance of a population can assume any of the above-forms.



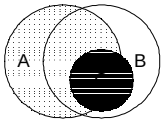

Form. I




Form. II




Form. III




Form. IV

**Form 1 and Form II**: A population of a highly constrained problem during the initial generations is likely to assume such forms where there are no feasible solutions. The elites refer to individuals that have a constraint rank =1. The parents are selected using a Roulette wheel based on the constraint rank of the individuals.

**Form III and Form IV:** A population will assume either of these forms when there are one or more feasible solutions. If the number of feasible solutions is less than half the size of the population, all the feasible individuals are assigned as elites. If the number of feasible solutions is more than half the size of the population, the set of solutions that are both feasible and good in objective performance are assigned as elites. As for mating, a parent P1 is selected from the set of elites using a Roulette wheel based on its objective rank. Two potential partners P2 and P3 are selected using Roulette wheel based on objective rank if the number of feasible solutions are more than half the population or selected using Roulette wheel based on constraint rank if the number of feasible solutions are less than half the population. A tournament selection between P2 and P3 is used to decide the winner that mates with P1.

## 3. RESULTS AND DISCUSSION

In this paper, a new evolutionary algorithm is proposed to solve constrained single objective optimization problems. The algorithm is attractive, as it does not require any additional inputs (scaling factors or weights) to handle constraints. The algorithm employs a parent centric recombination operator that is aimed to explore neighborhoods of good parents unlike some algorithms that use mean centric recombination.

We have reported our results for all the inequality constrained problems (without trigonometric functions) that appear in Koziel and Michalewicz[1]. The results of this study are based on 50 independent trials for each problem. The performance of the algorithm and the population histories are currently being studied to gain better insights on the process of evolution.

**References**


1. Koziel, S. and Michalewicz, Z. (1999). Evolutionary Algorithms, Homomorphous Mappings, and Constrained Parameter Optimization, *Evolutionary Computation*, Vol.7, No.1, pp.19-44.
2. Deb, K. (2000). An Efficient Constraint Handling Method for Genetic Algorithms, *Computer Methods in Applied Mechanics and Engineering*, 186, pp. 311-338.
3. Coello Coello, Carlos A. and Cruz Cortés, Nareli, (2001). Constraint-Handling in Genetic Algorithms through Emulations of the Immune System, *Tercer Encuentro Internacional de Ciencias de la Computación (ENC'01)*, Tomo I, pp. 115--124, Aguascalientes, Aguascalientes, Septiembre 2001.
4. Kowalczyk,R., (1997). Constraint Consistent Genetic Algorithms, Proceedings of The IEEE International Conference on Evolutionary Computation ICEC'97, Indianapolis, USA, p. 343-348.

**Table 1: Nature of the Test Functions**

| | N | Type of Function | % Feasible | LI | NI | Act. Const |
|---|---|---|---|---|---|---|
| G1 | 13 | Quadratic | 0.0111 | 9 | 0 | 6 |
| G4 | 5 | Quadratic | 52.1230 | 0 | 6 | 2 |
| G6 | 2 | Cubic | 0.0066 | 0 | 2 | 2 |
| G7 | 10 | Quadratic | 0.0003 | 3 | 5 | 6 |
| G9 | 7 | Polynomial | 0.5121 | 0 | 4 | 2 |
| G10 | 8 | Linear | 0.0010 | 3 | 3 | 6 |

**Table 2: Comparison of Results for G1**

| | | | | Best | Avg. /Med. | Worst | Fun. Evals. |
|---|---|---|---|---|---|---|---|
| G1 | Present | 50 | 20,000 | -15.0 | -14.7531<br>-15.0 | -13.0 | 718,854 |
| | Ref.1 | 70 | 5,000 | -14.7207 | -14.4609<br>---------- | -14.0566 | 350,000 |
| | Ref.1 | 70 | 20,000 | -14.7864 | -14.7082<br>---------- | -14.6154 | 1,400,000 |
| | Ref.2 | 130 | 500 | -15.0 | ----------<br>-15.0 | -13.0 | 65,130 |

The optimum solution to this problem is (1,1,1,1,1,1,1,1,3,3,3,1) with f =-15.0.

**Table 3: Comparison of Results for G4**

| | | | | Best | Avg. /Med. | Worst | Fun. Evals. |
|---|---|---|---|---|---|---|---|
| G4 | Present | 50 | 20,000 | -30603.0 | -30306.0<br>-30312.0 | -29849.0 | 11,523 |
| | Ref.1 | 70 | 5,000 | -30662.5 | -30643.8<br>---------- | -30617.0 | 350,000 |
| | Ref.1 | 70 | 20,000 | -30664.5 | -30655.3<br>---------- | -30645.9 | 1,400,000 |
| | Ref.2 | 50 | 1,000 | -30646.469 | ----------<br>-30279.744 | -29794.441 | 50,000 |
| | Ref.2 | 50 | 5,000 | -30665.537 | ----------<br>-30665.535 | -29846.654 | 250,000 |
| | Ref.3 | 90 | 10,000 | -30664.8 | -30632.4<br>---------- | -30493.7 | 900,000 |

The optimum solution is ( 78, 33, 29.995256,45,36.775812) with f = -30665.539.

**Table 4: Comparison of Results for G6**

| | | | | Best | Avg. /Med. | Worst | Fun. Evals. |
|---|---|---|---|---|---|---|---|
| G6 | Present | 50 | 20,000 | -6960.7 | -6885.3<br>-6911.1 | -6477.6 | 504,989 |
| | Ref.1 | 70 | 5,000 | -6901.5 | -6191.2<br>---------- | -4236.7 | 350,000 |
| | Ref.1 | 70 | 20,000 | -6952.1 | -6342.6<br>---------- | -5473.9 | 1,400,000 |
| | Ref.3 | 90 | 5,000 | -6961.7 | -6950.7<br>---------- | -6819.0 | 450,000 |

The optimum solution is (14.095, 0.84296) with f= -6961.81381.

**Table 5: Comparison of Results for G7**

| | | | | Best | Avg. /Med. | Worst | Fun. Evals. |
|---|---|---|---|---|---|---|---|
| G7 | Present | 50 | 20,000 | 24.5553 | 30.4769<br>29.2198 | 39.7454 | 517,887 |
| | Ref.1 | 70 | 5,000 | 25.132 | 26.619<br>---------- | 38.682 | 350,000 |
| | Ref.1 | 70 | 20,000 | 24.620 | 24.826<br>---------- | 25.069 | 1,400,000 |
| | Ref. 2 | 100 | 1,000 | 24.87747 | ----------<br>26.73401 | 50.40042 | 100,000 |
| | Ref. 2 | 100 | 3,500 | 24.37248 | ----------<br>24.40940 | 25.07530 | 350,100 |

The optimum solution is (2.171996, 2.363683, 8.773926, 5.095984, 0.9906548, 1.430574, 1.321644, 9.828726, 8.280092, 8.375927) with f= 24.3062091.

**Table 6: Comparison of Results for G9**

| | | | | Best | Avg. /Med. | Worst | Fun. Evals. |
|---|---|---|---|---|---|---|---|
| G9 | Present | 50 | 20,000 | 680.7160 | 683.6598<br>682.1545 | 714.4940 | 520,285 |
| | Ref.1 | 70 | 5,000 | 681.43 | 682.18<br>---------- | 682.88 | 350,000 |
| | Ref.1 | 70 | 20,000 | 680.91 | 681.16<br>---------- | 683.18 | 1,400,000 |
| | Ref.2 | 70 | 1,000 | 680.659424 | ----------<br>681.525635 | 687.188599 | 70,070 |
| | Ref.2 | 70 | 5,000 | 680.634460 | ----------<br>680.641724 | 680.650879 | 350,070 |

The optimal solution (2.330499, 1.951372, -0.4775414, 4.365726, -0.6244870, 1.038131, 1.594227) with f =680.6300573.

**Table 7: Comparison of Results for G10**

| | | | | Best | Avg. /Med. | Worst | Fun. Evals. |
|---|---|---|---|---|---|---|---|
| G10 | Present | 50 | 20,000 | 7323.4 | 8828.8<br>8565.0 | 12675.0 | 185,107 |
| | Ref.1 | 70 | 5,000 | 7215.8 | 9141.7<br>---------- | 11894.5 | 350,000 |
| | Ref.1 | 70 | 20,000 | 7147.9 | 8163.6<br>---------- | 9659.3 | 1,400,000 |
| | Ref. 2 | 80 | 1,000 | 7065.742 | ----------<br>8274.830 | 10925.165 | 80,080 |
| | Ref. 2 | 80 | 4,000 | 7060.221 | ----------<br>7220.026 | 10230.834 | 320,080 |
| | Ref. 4 | 70 | 2,000 | 7063.95605 | 7310.10449 | 7854.61475 | 140,000 |

The optimal solution (579.3167, 1359.943, 5110.071, 182.0174, 295.5985, 217.9799, 286.4162, 395.5979) with f=7049.330923.