A survey of Constraint Handling Techniques in Evolutionary Computation Methods

Author:
Zbigneiw Michalewicz

Presenter:
Masoud Mazloom

27th Oct. 2010

Outline

- Introduction
- Numerical Optimization and Unfeasible solutions:
  - Nonlinear programming problem
  - Constraint-handling methods:
    - The method of Homaifar, Lai & Qi
    - The method of Joines & Houck
    - The method of Michalewicz & Janikow
    - The method of Michalewicz & Attia
    - The method of Powell &skolnick
    - The method of Schoenauer
    - Death Penalty
    - Repair method
- Five test case
- Experiments, Results

Introduction

- Evaluation function:

  one of the major components of any Evolutionary system.
- Evolutionary computation techniques use efficient evaluation function for feasible individuals but these techniques have not developed any guidelines on how to deal with unfeasible solutions.
- Evolutionary programming and evolution strategy:

  (Back et al. 1991)

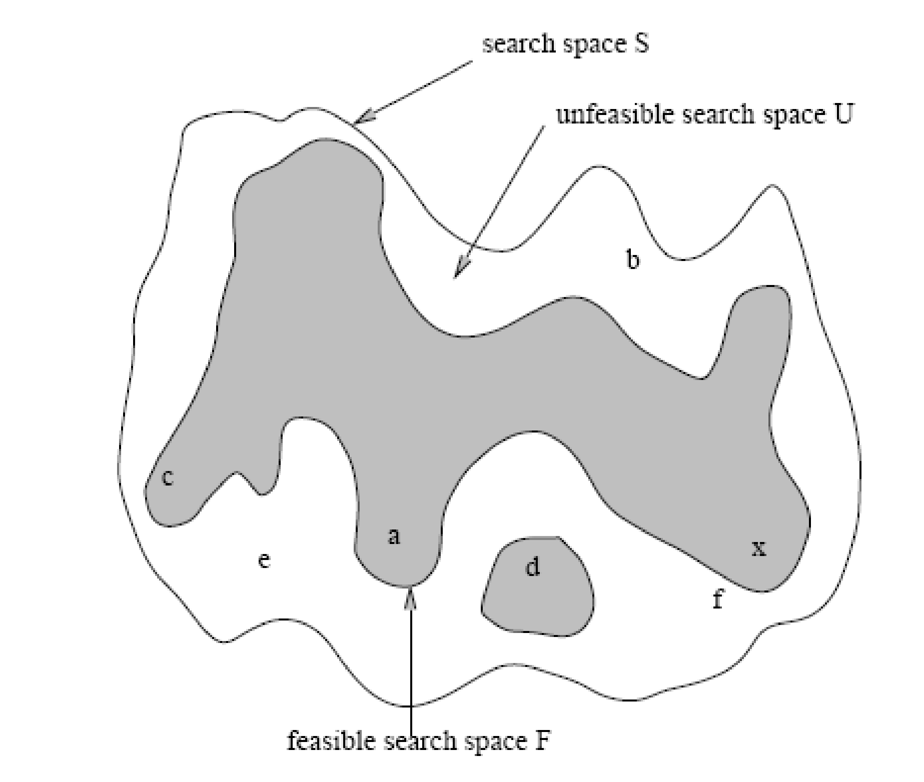
  (Fogel and Stayton 1994)

  Reject unfeasible individuals
- Genetic Algorithms:

  (Goldberg 1989)

  Penalize unfeasible individuals
- Is there any general rule for designing penalty functions?

Introduction Continue



A search space and its feasible part

We don’t make any assumptions about about these subspace:

- They need not be convex
- They not be connected

Introduction Continue.

- We have to deal with various feasible and unfeasible individuals.
- A population may contain some feasible (a, c, d) and unfeasible individuals (b, e, f), while the optimum solution is ‘x’
- Problem:

  How to deal with unfeasible individuals?
- In general we have to design two two evaluations functions:

  1. eval *f:* for feasible domain
  2. eval *u:* for unfeasible domain

Introduction Continue

1. How should two feasible individuals be compared?

2. How should two unfeasible individuals be compared?

3. Should we assume that $eval_f(s) \succ eval_u(p)$ for any $s \in \mathcal{F}$ and any $p \in \mathcal{U}$
   In particular, which individual is better: individual ‘c’ or unfeasible individual ‘f’ (note that the optimum is ‘x’)

4. Should we consider unfeasible individuals harmful and eliminate them from the population? (may be there are useful)

5. Should we ‘repair’ unfeasible solutions by moving them into the closest point of the feasible space ? (should we use a repair procedure for evaluation purpose only)

6. Should we chose to penalize unfeasible individuals?

$$eval_u(p) = eval_f(p) + penalty(p)$$

Numerical Optimization and Unfeasible solutions

Non linear programming problem:

The general nonlinear programming problem for continuous variables is to find $\overline{X}$ so as to

$$\text{optimize } f(\overline{X}),\ \overline{X} = (x_1, \ldots, x_n) \in R^n,$$

where $\overline{X} \in \mathcal{F} \subseteq \mathcal{S}$. The set $\mathcal{S} \subseteq R^n$ defines the search space and the set $\mathcal{F} \subseteq \mathcal{S}$ defines a *feasible* part of the search space. Usually, the search space $\mathcal{S}$ is defined as an n-dimensional rectangle in R^n (domains of variables defined as lower and upper bounds):

$$left(i) \leq x_i \leq right(i), \quad 1 \leq i \leq n,$$

whereas the feasible set $\mathcal{F}$ is defined by the search space $\mathcal{S}$ and an additional set of constraints:

$$g_j(\overline{X}) \leq 0, \text{ for } j = 1, \ldots, q, \text{ and}$$
$$h_j(\overline{X}) = 0, \text{ for } j = q+1, \ldots, m.$$

Condtrain Handling methods

1. Methods based on preserving feasibility of solutions:
   a) Use of specialized operations (The method of Michalewicz & Janikow)
   b) Searching the boundary of feasible region

2. Methods based on penalty functions:
   a) Static penalty (The method of Homaifar, Lai & Qi)
   b) Dynamic Penalty (The method of Joines & Houck)
   c) Annealing penalty (The method of Michalewicz & Attia)
   d) Death Penalty (The method of Michalewicz)

3. Methods based on a search for feasible soluitons:
   a) Superiority of feasible points (The method of Powell & Skolniks)
   b) Behavioral memory method (The method of Schoenauer & Xanthakis)

4. Muti-Objective optimization methods:
   a) the method of Paredis
   b) Cultural Algorithms

Static penalty (The method of Homaifar, Lai & Qi)

- For every constraint we establish a family of intervals that determine appropriate penalty value.
- The methods works as follows:

- for each constraint, create several (ℓ) levels of violation,
- for each level of violation and for each constraint, create a penalty coefficient R_{ij} ($i = 1, 2, \ldots, \ell$, $j = 1, 2, \ldots, m$); higher levels of violation require larger values of this coefficient.
- start with a random population of individuals (i.e., these individuals are feasible or unfeasible),
- evaluate individuals using the following formula

$$eval(\overline{X}) = f(\overline{X}) + \sum_{j=1}^{m} R_{ij} f_j^2(\overline{X}),$$

where R_{ij} is a penalty coefficient for the i-th level of violation and the j-th constraint.

Static penalty Continue

- Weakness:

  Number of parameters: m(2l+1)

  m=5, l=4: 45 parameters
- Quality of solutions heavily depends on the values of these parameters:

  if Rij are moderate: algorithm converge to an unfeasible solution

  if Rij are too large: algorithm reject unfeasible solution
- Finding an optimal set of parameters for reaching to a feasible solution near optimum is quite difficult.

Dynamic Penalty (The method of Joines & Houck)

- In this method individuals are evaluated (at the iteration t) by the following formula:

$$eval(\overline{X}) = f(\overline{X}) + (C \times t)^{\alpha} \sum_{j=1}^{m} f_j^{\beta}(\overline{X}),$$

• C,α, β are constant.
• The method is quite similar to Homaifar et al. but it require many fewer parameter .
• This method is independent of the total number of constraints.
•Penalty component changes with the generation number.
•The pressure on unfeasible solutions is increased due to the: $(C \times t)^{\alpha}$

Quality of the solution was very sensitive to $(C = 0.5, \alpha = \beta = 2)$ three parameters (reasonable parameters are

Annealing penalty (The method of Michalewicz & Attia)

- In this method linear and nonlinear constraints are processed separately.
- The method works as follow:

- divide all constraints into four subsets: linear equations, linear inequalities, nonlinear equations, and nonlinear inequalities,
- select a random single point as a starting point (the initial population consists of copies of this single individual). This initial point satisfies all linear constraints,
- create a set of active constraints A; include there all nonlinear equations and all violated nonlinear inequalities.
- set the initial temperature $\tau = \tau_0$,
- evolve the population using the following formula:

  $$eval(\overline{X}, \tau) = f(\overline{X}) + \frac{1}{2\tau} \sum_{j \in A} f_j^2(\overline{X}),$$

  (only active constraints are considered),
- if $\tau < \tau_f$, stop, otherwise
  - decrease temperature τ,
  - the best solution serves as a starting point of the next iteration,
  - update the set of active constraints A,
  - repeat the previous step of the main part.

- At every iteration the algorithm considers active constraints only.
- The pressure on unfeasible solution is increased due to the decreasing values of temperature ζ.
- This method is quite sensitive to values of its parameters:

  starting temperature ζ, freezing temperature ζ0 and cooling scheme ζf to decrease temperature ζ (standard value s are $\tau_0 = 1,\ \tau_{i+1} = 0.1 \cdot \tau_i$, with $\tau_f = 0.000001$)
- This algorithm may converge to a near-optimum solution just in one iteration

Death Penalty (The method of Michalewicz

- This method is popular option in many evolutionary techniques like evolutionary programming.
- Removing unfeasible solutions from population may work well when the feasible search space F is convex and it constitutes a reasonable part of the whole search space.
- For some problems where the ratio between the size of F and S is small and an initial population consist of unfeasible individuals only, we must to improve them.
- Experiment shows that when ratio between F and S was between 0% and 0.5% this method performed worse than other methods.
- Unfeasible solutions should provide information and not just be thrown away

Methods based on preserving feasibility of solutions:
Use of specialized operators
(The method of Michalewicz & Janikow)

- The idea behind this method is based on specialized operators which transform feasible individuala into feasible individuals.
- This method works only with linear constraint and a feasible starting point .
- A close set of operators maintains feasibility of solutions (the offspring solution vector is always feasible).
- Mutation (which values xi will give from its domain?)
- Crossover: $a\overline{X} + (1-a)\overline{Y}$ $(\text{for } 0 \le a \le 1)$
- It gave surprisingly good performance on many test functions.
- Weakness of the method lies in its inability to deal with non convex search space (to deal with nonlinear constraints in general)

Methods based on a search for feasible solutions: Superiority of feasible points (The method of Powell & Skolniks)

- The key concept behind this method is the assumption of superiority of feasible solutions over unfeasible ones.
- Incorporate a heuristic rule for processing unfeasible solutions:

" every feasible solution is better than every unfeasible solution"

- This rule is implemented in the following way:

evaluations of feasible solutions are mapped into the interval $(-\infty,1)$

evaluations of unfeasible solutions are mapped into the interval $(1, \infty)$

- Evaluation procedure:

$$eval_f(\overline{X}) = f(\overline{X}),$$
$$eval_u(\overline{X}) = f(\overline{X}) + r\sum_{j=1}^{m} f_j(\overline{X}),$$

where r is a constant, and

$$eval(\overline{X}) = \begin{cases} eval_f(\overline{X}), & if\ \overline{X} \in \mathcal{F} \\ eval_u(\overline{X}) + \rho(\overline{X}, t), & if\ \overline{X} \in \mathcal{S} - \mathcal{F}. \end{cases}$$

The function $\rho(\overline{X}, t)$ influences unfeasible solutions only; it is defined as

$$\rho(\overline{X}, t) = \max\{0, \max_{\overline{X} \in \mathcal{F}}\{eval_f(\overline{X})\} - \min_{\overline{X} \in \mathcal{S}-\mathcal{F}}\{eval_u(\overline{X})\}.$$

In other words, unfeasible individuals have increased penalties: they may not be better than the worst ($\max_{\overline{X} \in \mathcal{F}}\{eval_f(\overline{X})\}$) feasible individual.

- Experiments show that this method work very well however the topology of the feasible search space might be an important factor.
- For problems with a small ratio |F|/|S| the algorithm is often trapped into an unfeasible solution.
- This method should require at least one feasible individual for start.

Behavioral memory method
(The method of Schoenauer & Xanthakis)

- This method based on the idea of handling constraints in a particular order.

- start with a random population of individuals (i.e., these individuals are feasible or unfeasible),
- set $j = 1$ (j is constraint counter),
- evolve this population to minimize the violation of the j-th constraint, until a given percentage of the population (so-called flip threshold ϕ) is feasible for this constraint. In this case

$$eval(\overline{X}) = g_1(\overline{X}).$$

- set $j = j + 1$,
- the current population is the starting point for the next phase of the evolution, minimizing the violation of the j-th constraint:

$$eval(\overline{X}) = g_j(\overline{X}).^2$$

During this phase, points that do not satisfy at least one of the 1st, 2nd, ... ,$(j-1)$-th constraints are eliminated from the population. The halting criterion is again the satisfaction of the j-th constraint by the flip threshold percentage ϕ of the population.

- if $j < m$, repeat the last two steps, otherwise $(j = m)$ optimize the objective function f rejecting unfeasible individuals.

This method require 3 parameter:

sharing factor (to maintain diversity of the population)

flip threshold

particular permutation of constraint which determine their order

- In the final step of this algorithm the objective function *f* is optimized.
- But for large feasible spaces the method just provides additional computational overhead.
- For very small feasible search space it is essential to maintain a diversity in the population
- Experiments indicate that the method provides a reasonable performance except when the feasible search "too small"

Test case #1

The problem [4] is to minimize a function:

$$G1(\overline{X}) = 5x_1 + 5x_2 + 5x_3 + 5x_4 - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=5}^{13} x_i,$$

subject to

$$\begin{aligned}
&2x_1 + 2x_2 + x_{10} + x_{11} \le 10,\\
&2x_1 + 2x_3 + x_{10} + x_{12} \le 10,\\
&2x_2 + 2x_3 + x_{11} + x_{12} \le 10,\\
&-8x_1 + x_{10} \le 0, \quad -8x_2 + x_{11} \le 0,\\
&-8x_3 + x_{12} \le 0, \quad -2x_4 - x_5 + x_{10} \le 0,\\
&-2x_6 - x_7 + x_{11} \le 0, \quad -2x_8 - x_9 + x_{12} \le 0,\\
&0 \le x_i \le 1,\ i = 1, \ldots, 9, \quad 0 \le x_i \le 100,\\
&i = 10, 11, 12, \quad 0 \le x_{13} \le 1.
\end{aligned}$$

The problem has 9 linear constraints; the function $G1$ is quadratic with its global minimum at

$$\overline{X}^* = (1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 1),$$

where $G1(\overline{X}^*) = -15$. Six (out of nine) constraints are active at the global optimum (all except the following three: $-8x_1 + x_{10} \le 0$, $-8x_2 + x_{11} \le 0$, $-8x_3 + x_{12} \le 0$).

Test case #2

The problem [6] is to minimize a function:

$$G2(\overline{X}) = x_1 + x_2 + x_3,$$

where

$$\begin{aligned}
&1 - 0.0025(x_4 + x_6) \geq 0,\\
&1 - 0.0025(x_5 + x_7 - x_4) \geq 0,\\
&1 - 0.01(x_8 - x_5) \geq 0,\\
&x_1 x_6 - 833.33252 x_4 - 100 x_1 + 83333.333 \geq 0,\\
&x_2 x_7 - 1250 x_5 - x_2 x_4 + 1250 x_4 \geq 0,\\
&x_3 x_8 - 1250000 - x_3 x_5 + 2500 x_5 \geq 0,\\
&100 \leq x_1 \leq 10000, \quad 1000 \leq x_i \leq 10000,\\
&i = 2, 3, \quad 10 \leq x_i \leq 1000, \quad i = 4, \ldots, 8.
\end{aligned}$$

The problem has 3 linear and 3 nonlinear constraints; the function $G2$ is linear and has its global minimum at

$$\begin{aligned}
\overline{X}^* = (&579.3167, 1359.943, 5110.071, 182.0174,\\
&295.5985, 217.9799, 286.4162, 395.5979),
\end{aligned}$$

where $G2(\overline{X}^*) = 7049.330923$. All six constraints are active at the global optimum.

Test case #3

The problem [6] is to minimize a function:

$$G3(\overline{X}) = (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2 + 10x_5^6 + 7x_6^2 + x_7^4 - 4x_6x_7 - 10x_6 - 8x_7,$$

where

$$\begin{aligned}
&127 - 2x_1^2 - 3x_2^4 - x_3 - 4x_4^2 - 5x_5 \geq 0,\\
&282 - 7x_1 - 3x_2 - 10x_3^2 - x_4 + x_5 \geq 0,\\
&196 - 23x_1 - x_2^2 - 6x_6^2 + 8x_7 \geq 0,\\
&-4x_1^2 - x_2^2 + 3x_1x_2 - 2x_3^2 - 5x_6 + 11x_7 \geq 0\\
&-10.0 \leq x_i \leq 10.0,\ i = 1, \ldots, 7.
\end{aligned}$$

The problem has 4 nonlinear constraints; the function $G3$ is nonlinear and has its global minimum at

$$\overline{X}^* = (2.330499, 1.951372, -0.4775414, 4.365726, -0.6244870, 1.038131, 1.594227),$$

where $G3(\overline{X}^*) = 680.6300573$. Two (out of four) constraints are active at the global optimum (the first and the last one).

Test case #4

The problem [6] is to minimize a function:

$$G4(\overline{X}) = e^{x_1 x_2 x_3 x_4 x_5},$$

subject to

$$x_1^2+x_2^2+x_3^2+x_4^2+x_5^2 = 10, \quad x_2x_3-5x_4x_5 = 0,$$
$$x_1^3 + x_2^3 = -1, \quad -2.3 \le x_i \le 2.3,\ i = 1,2,$$
$$-3.2 \le x_i \le 3.2, \quad i = 3,4,5.$$

The problem has 3 nonlinear equations; nonlinear function $G4$ has its global minimum at

$$\overline{X}^* = (-1.717143, 1.595709, 1.827247, -0.7636413, -0.7636450),$$

where $G4(\overline{X}^*) = 0.0539498478.$

Test Case #5

The problem [6] is to minimize a function:

$$G5(\overline{X}) = x_1^2 + x_2^2 + x_1 x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 \\ +4(x_4 - 5)^2 + (x_5 - 3)^2 + 2(x_6 - 1)^2 + 5x_7^2 + \\ 7(x_8 - 11)^2 + 2(x_9 - 10)^2 + (x_{10} - 7)^2 + 45,$$

where

$$105 - 4x_1 - 5x_2 + 3x_7 - 9x_8 \geq 0,$$
$$-10x_1 + 8x_2 + 17x_7 - 2x_8 \geq 0,$$
$$8x_1 - 2x_2 - 5x_9 + 2x_{10} + 12 \geq 0,$$
$$-3(x_1 - 2)^2 - 4(x_2 - 3)^2 - 2x_3^2 + 7x_4 + 120 \geq 0,$$
$$-5x_1^2 - 8x_2 - (x_3 - 6)^2 + 2x_4 + 40 \geq 0,$$

$$-x_1^2 - 2(x_2 - 2)^2 + 2x_1 x_2 - 14x_5 + 6x_6 \geq 0,$$
$$-0.5(x_1 - 8)^2 - 2(x_2 - 4)^2 - 3x_5^2 + x_6 + 30 \geq 0,$$
$$3x_1 - 6x_2 - 12(x_9 - 8)^2 + 7x_{10} \geq 0,$$
$$-10.0 \leq x_i \leq 10.0, \quad i = 1, \ldots, 10.$$

The problem has 3 linear and 5 nonlinear constraints; the function $G5$ is quadratic and has its global minimum at

$$\overline{X}^* = (2.171996, 2.363683, 8.773926, 5.095984, \\ 0.9906548, 1.430574, 1.321644, 9.828726, \\ 8.280092, 8.375927),$$

where $G5(\overline{X}^*) = 24.3062091$. Six (out of eight) constraints are active at the global optimum (all except the last two).

Summary of test case

TC	n	Type of f	ρ	LI	NE	NI	a
#1	13	quadratic	0.0111%	9	0	0	6
#2	8	linear	0.0010%	3	0	3	6
#3	7	polynomial	0.5121%	0	0	4	2
#4	5	nonlinear	0.0000%	0	3	0	3
#5	10	quadratic	0.0003%	3	0	5	6

- In the table below :

Method #1 is: Homaifar’s method

Method #2 is: joines ‘s method

Method #3 is: Xanthakis’s mehtod

Method #4 is: Attia’ s method

Method #5 is: powell’s method

Method #6 is: Death penalty

Experimental, Result

TC	Exact opt.		Method #1	Method #2	Method #3	Method #4	Method #5	Method #6	Method #6(f)
#1	-15.000	b	-15.002	-15.000	-15.000	-15.000	-15.000		-15.000
		m	-15.002	-15.000	-15.000	-15.000	-15.000	—	-14.999
		w	-15.001	-14.999	-14.998	-15.000	-14.999		-13.616
		c	0, 0, 4	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0		0, 0, 0
#2	7049.331	b	2282.723	3117.242	7485.667	7377.976	2101.367		7872.948
		m	2449.798	4213.497	8271.292	8206.151	2101.411	—	8559.423
		w	2756.679	6056.211	8752.412	9652.901	2101.551		8668.648
		c	0, 3, 0	0, 3, 0	0, 0, 0	0, 0, 0	1, 2, 0		0, 0, 0
#3	680.630	b	680.771	680.787	680.836	680.642	680.805	680.934	680.847
		m	681.262	681.111	681.175	680.718	682.682	681.771	681.826
		w	689.660	682.798	685.640	680.955	685.738	689.442	689.417
		c	0, 0, 1	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0
#4	0.054	b	0.084	0.059		0.054	0.067		
		m	0.955	0.812	*	0.064	0.091	*	*
		w	1.000	2.542		0.557	0.512		
		c	0, 0, 0	0, 0, 0		0, 0, 0	0, 0, 0		
#5	24.306	b	24.690	25.486		18.917	17.388		25.653
		m	29.258	26.905	—	24.418	22.932	—	27.116
		w	36.060	42.358		44.302	48.866		32.477
		c	0, 1, 1	0, 0, 0		0, 1, 0	1, 0, 0		0, 0, 0

- Questions?