

# Constraint-Handling Techniques used with Evolutionary Algorithms

Carlos A. Coello Coello  
CINVESTAV-IPN  
Mexico City, México  
*ccoello@cs.cinvestav.mx*

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

GECCO'19 Companion, Prague, Czech Republic.

© 2019 Copyright held by the owner/author(s).

978-1-4503-6748-6/19/07...\$15.00

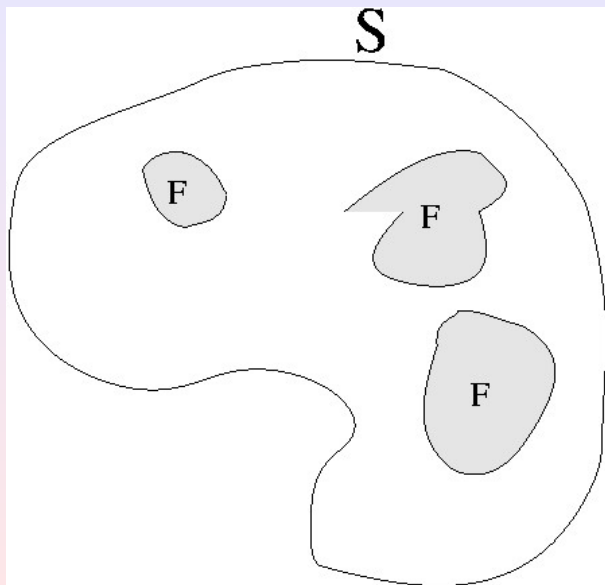
DOI: <https://doi.org/10.1145/3319619.3323366>

Evolutionary Algorithms (EAs) have been found successful in the solution of a wide variety of optimization problems. However, EAs are unconstrained search techniques. Thus, incorporating constraints into the fitness function of an EA is an open research area.

There is a considerable amount of research regarding mechanisms that allow EAs to deal with equality and inequality constraints; both type of constraints can be linear or nonlinear.

Such work is precisely the scope of this tutorial.

# Search Space



# A Taxonomy of Constraint-Handling Approaches

Constraint-handling techniques used with EAs can be classified in five main groups:

- Penalty Functions
- Special representations and operators
- Repair algorithms
- Separation of constraints and objectives
- Hybrid Methods

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, January 2002.

Efrén Mezura-Montes and Carlos A. Coello Coello, "**Constraint-Handling in Nature-Inspired Numerical Optimization: Past, Present and Future**", *Swarm and Evolutionary Computation*, Vol. 1, No. 4, pp. 173-194, December 2011.

# Penalty Functions



The most common approach in the EA community to handle constraints (particularly, inequality constraints) is to use penalties.

Penalty functions were originally proposed by Richard Courant in the 1940s and were later expanded by Carroll and Fiacco & McCormick.

R. Courant, "**Variational methods for the Solution of Problems of Equilibrium and Vibrations**", *Bulletin of the American Mathematical Society*, Vol. 49, pp. 1–23, 1943.

# Penalty Functions

In mathematical programming, two kinds of penalty functions are considered: exterior and interior. In the case of **exterior** methods, we start with an infeasible solution and from there we move towards the feasible region.

In the case of **interior methods**, the penalty term is chosen such that its value will be small at points away from the constraint boundaries and will tend to infinity as the constraint boundaries are approached. Then, if we start from a feasible point, the subsequent points generated will always lie within the feasible region since the constraint boundaries act as barriers during the optimization process.

G.V. Rekliatis, A. Ravindran and K.M. Ragsdell, **Engineering Optimization. Methods and Applications**, John Wiley and Sons, New York, USA, 1983, ISBN 0-471-05579-4.

# Penalty Functions

EAs normally adopt exterior penalty functions of the form:

$$\phi(\vec{X}) = f(\vec{X}) \pm \left[ \sum_{i=1}^n r_i \times G_i + \sum_{j=1}^p c_j \times L_j \right] \quad (1)$$

where  $\phi(\vec{X})$  is the new (expanded) objective function to be optimized,  $G_i$  and  $L_j$  are functions of the constraints  $g_i(\vec{X})$  and  $h_j(\vec{X})$ , respectively, and  $r_i$  and  $c_j$  are positive constants normally called “penalty factors”.

The most common form of  $G_i$  and  $L_j$  is:

$$G_i = \max[0, g_i(\vec{X})]^\beta \quad (2)$$

$$L_j = |h_j(\vec{X})|^\gamma \quad (3)$$

where  $\beta$  and  $\gamma$  are normally 1 or 2.

# Penalty Functions

Penalty functions can deal both with equality and inequality constraints, and the normal approach is to transform an equality to an inequality of the form:

$$|h_j(\vec{x})| - \epsilon \leq 0 \quad (4)$$

where  $\epsilon$  is the tolerance allowed (a very small value).

Alice E. Smith and David W. Coit, “**Constraint Handling Techniques–Penalty Functions**”, in Thomas Bäck, David B. Fogel and Zbigniew Michalewicz (editors), *Handbook of Evolutionary Computation*, Chapter C 5.2. Oxford University Press and Institute of Physics Publishing, 1997.



# Types of Penalty Functions used with EAs

- Death Penalty
- Static Penalty
- Dynamic Penalty
- Adaptive Penalty
- Other Approaches
  - Self-Adaptive Fitness Formulation
  - ASCHEA
  - Stochastic Ranking

Efrén Mezura-Montes and Carlos A. Coello Coello,  
“**Constraint-Handling in Nature-Inspired Numerical Optimization: Past, Present and Future**”, *Swarm and Evolutionary Computation*, Vol. 1, No. 4, pp. 173-194,  
December 2011.

# Death Penalty

The rejection of infeasible individuals (also called “death penalty”) is probably the easiest way to handle constraints and it is also computationally efficient, because when a certain solution violates a constraint, it is rejected and generated again. Thus, no further calculations are necessary to estimate the degree of infeasibility of such a solution.

This sort of approach was commonly used in the early days of evolution strategies.

H.-P. Schwefel, “**Evolution and Optimum Seeking**”, Wiley Interscience, New York, New York, USA, 1995.

Oliver Kramer, “**A Review of Constraint-Handling Techniques for Evolution Strategies**”, *Applied Computational Intelligence and Soft Computing*, Vol. 2010, Article ID 185063, No. 1, pp. 1–11, January, 2010.

# Criticism to Death Penalty

- Not advisable, except in the case of problems in which the feasible region is fairly large.
- No use of information from infeasible points.
- Search may “stagnate” in the presence of very small feasible regions.
- A variation that assigns a zero fitness to infeasible solutions may work surprisingly well in practice.

Kaisa Miettinen, Marko M. Mäkelä and Jari Toivanen,  
“**Numerical comparison of some penalty-based constraint handling techniques in genetic algorithms**”, *Journal of Global Optimization*, Vol. 27, No. 4, pp. 427–446, December 2003.

# Static Penalty

Under this category, we consider approaches in which the penalty factors do not depend on the current generation number in any way, and therefore, remain constant during the entire evolutionary process.

An example of this sort of scheme is the approach proposed by Homaifar, Lai and Qi [1994] in which they define levels of violation of the constraints (and penalty factors associated to them):

$$\text{fitness}(\vec{x}) = f(\vec{x}) + \sum_{i=1}^m \left( R_{k,i} \times \max [0, g_i(\vec{x})]^2 \right) \quad (5)$$

where  $R_{k,i}$  are the penalty coefficients used,  $m$  is total the number of constraints,  $f(\vec{x})$  is the unpenalized objective function, and  $k = 1, 2, \dots, l$ , where  $l$  is the number of levels of violation defined by the user.

A. Homaifar, S. H. Y. Lai and X. Qi, “**Constrained Optimization via Genetic Algorithms**”, *Simulation*, Vol. 62, No. 4, pp. 242–254, 1994.

# Criticism to Static Penalty

- It may not be a good idea to keep the same penalty factors along the entire evolutionary process.
- Penalty factors are, in general, problem-dependent.
- Approach is simple, although in some cases (e.g., in the approach by Homaifar, Lai and Qi [1994]), the user may need to set up a high number of penalty factors.

Alice E. Smith and David W. Coit, “**Constraint Handling Techniques–Penalty Functions**”, in Thomas Bäck, David B. Fogel and Zbigniew Michalewicz (editors), *Handbook of Evolutionary Computation*, Chapter C 5.2. Oxford University Press and Institute of Physics Publishing, 1997.

# Dynamic Penalty

Within this category, we will consider any penalty function in which the current generation number is involved in the computation of the corresponding penalty factors (normally the penalty factors are defined in such a way that they increase over time—i.e., generations).

An example is the approach from Joines and Houck [1994] in which individuals are evaluated (at generation  $t$ ) using:

$$\text{fitness}(\vec{x}) = f(\vec{x}) + (C \times t)^\alpha \times \text{SVC}(\beta, \vec{x}) \quad (6)$$

where  $C$ ,  $\alpha$  and  $\beta$  are constants defined by the user (the authors used  $C = 0.5$ ,  $\alpha = 1$  or  $2$ , and  $\beta = 1$  or  $2$ ).

J. Joines and C. Houck, “**On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GAs**”, in David Fogel (editor), *Proceedings of the first IEEE Conference on Evolutionary Computation*, pp. 579–584, IEEE Press, Orlando, Florida, 1994.

$SVC(\beta, \vec{x})$  is defined as:

$$SVC(\beta, \vec{x}) = \sum_{i=1}^n D_i^{\beta}(\vec{x}) + \sum_{j=1}^p D_j(\vec{x}) \quad (7)$$

and

$$D_i(\vec{x}) = \begin{cases} 0 & g_i(\vec{x}) \leq 0 \\ |g_i(\vec{x})| & \text{otherwise} \end{cases} \quad 1 \leq i \leq n \quad (8)$$

$$D_j(\vec{x}) = \begin{cases} 0 & -\epsilon \leq h_j(\vec{x}) \leq \epsilon \\ |h_j(\vec{x})| & \text{otherwise} \end{cases} \quad 1 \leq j \leq p \quad (9)$$

This dynamic function increases the penalty as we progress through generations.

# Criticism to Dynamic Penalty

- Some researchers have argued that dynamic penalties work better than static penalties.
- In fact, many EC researchers consider dynamic penalty as a good choice when dealing with an arbitrary constrained optimization problem.
- Note however, that it is difficult to derive good dynamic penalty functions in practice as it is difficult to produce good penalty factors for static functions.

S. Kazarlis and V. Petridis, **“Varying Fitness Functions in Genetic Algorithms: Studying the Rate of Increase of the Dynamic Penalty Terms”**, in A. E. Eiben, T. Bäck, M. Schoenauer and H.-P. Schwefel (editors), *Proceedings of the 5th Parallel Problem Solving from Nature (PPSN V)*, pages 211–220, Heidelberg, Germany, Springer-Verlag. Lecture Notes in Computer Science Vol. 1498, September 1998.



# Adaptive Penalty

Bean and Hadj-Alouane [1992] developed a method that uses a penalty function which takes a feedback from the search process. Each individual is evaluated by the formula:

$$\text{fitness}(\vec{x}) = f(\vec{x}) + \lambda(t) \left[ \sum_{i=1}^n g_i^2(\vec{x}) + \sum_{j=1}^p |h_j(\vec{x})| \right] \quad (10)$$

where  $\lambda(t)$  is updated at every generation  $t$ .

James C. Bean and Atidel Ben Hadj-Alouane, “**A Dual Genetic Algorithm for Bounded Integer Programs**”, Technical Report TR 92-53, Department of Industrial and Operations Engineering, The University of Michigan, USA, 1992

# Adaptive Penalty

$\lambda(t)$  is updated in the following way:

$$\lambda(t+1) = \begin{cases} (1/\beta_1) \cdot \lambda(t), & \text{if case \#1} \\ \beta_2 \cdot \lambda(t), & \text{if case \#2} \\ \lambda(t), & \text{otherwise,} \end{cases} \quad (11)$$

where cases #1 and #2 denote situations where the best individual in the last  $k$  generations was always (case #1) or was never (case #2) feasible,  $\beta_1, \beta_2 > 1$ ,  $\beta_1 > \beta_2$ , and  $\beta_1 \neq \beta_2$  (to avoid cycling).

In other words, the penalty component  $\lambda(t+1)$  for the generation  $t+1$  is decreased if all the best individuals in the last  $k$  generations were feasible or is increased if they were all infeasible. If there are some feasible and infeasible individuals tied as best in the population, then the penalty does not change.

# Criticism to Adaptive Penalty

- Setting the parameters of this type of approach is not trivial (e.g., what generational gap ( $k$ ) is appropriate?).
- This sort of approach regulates in a more “intelligent” way the penalty factors.
- An interesting aspect of this approach is that it tries to avoid having either an all-feasible or an all-infeasible population. Other constraint-handling approaches pay a lot of attention to this issue.
- A very similar approach was proposed by Rasheed [1998] for continuous optimization.

Khaled Rasheed, “**An Adaptive Penalty Approach for Constrained Genetic-Algorithm Optimization**”, in John R. Koza et al. (editors), *Proceedings of the Third Annual Genetic Programming Conference*, pp. 584–590, Morgan Kaufmann Publishers, San Francisco, California, USA, 1998.

# Penalty Functions: Central Issues

The main problem with penalty functions is that the “ideal” penalty factor to be adopted in a penalty function cannot be known *a priori* for an arbitrary problem.

If the penalty is too high and the optimum lies at the boundary of the feasible region, the EA will be pushed inside the feasible region very quickly, and will not be able to move back towards the boundary with the infeasible region.

On the other hand, if the penalty is too low, a lot of the search time will be spent exploring the infeasible region because the penalty will be negligible with respect to the objective function.

Jon T. Richardson, Mark R. Palmer, Gunar Liepins and Mike Hilliard, “**Some Guidelines for Genetic Algorithms with Penalty Functions**”, in J. David Schaffer (editor), *Proceedings of the Third International Conference on Genetic Algorithms (ICGA-89)*, pp. 191–197, Morgan Kaufmann Publishers, San Mateo, California, USA, June 1989.

# Other Approaches

Evidently, many other penalty-based techniques have been proposed in the literature (see for example Yeniyay [2005] and Lemonge et al. [2012]).

However, three modern constraint-handling approaches that use penalty functions deserve special consideration, since they are highly competitive:

- Self-Adaptive Fitness Formulation
- ASCHEA
- Stochastic Ranking

Özgür Yeniyay, “**Penalty Function Methods for Constrained Optimization with Genetic Algorithms**”, *Mathematical and Computational Applications*, Vol. 10, No. 1, pp. 45–56, 2005.

Afonso C.C. Lemonge, Helio J.C. Barbosa and Heder S. Bernardino, “**A Family of Adaptive Penalty Schemes for Steady-state Genetic Algorithms**”, in *2012 IEEE Congress on Evolutionary Computation (CEC'2012)*, pp. 505–512, IEEE Press, Brisbane, Australia, June 10-15, 2012.

# Self-Adaptive Fitness Formulation

- Proposed by Farmani and Wright [2003].
- The approach uses an adaptive penalty that is applied in 3 steps:
  - 1 The sum of constraint violation is computed for each individual.
  - 2 The best and worst solutions are identified in the current population.
  - 3 A penalty is applied in two parts:
    - It is applied only if one or more feasible solutions have a better objective function value than the best solution found so far. The idea is to increase the fitness of the infeasible solutions.
    - Increase the fitness of the infeasible solutions as to favor those solutions which are nearly feasible and also have a good objective function value.

Raziye Farmani and Jonathan A. Wright, “**Self-Adaptive Fitness Formulation for Constrained Optimization**,” *IEEE Transactions on Evolutionary Computation*, Vol. 7, No. 5, pp. 445–455, October 2003.

# Self-Adaptive Fitness Formulation

- The penalty factor is defined in terms of both the best and the worst solutions.
- The authors used a genetic algorithm with binary representation (with Gray codes) and roulette-wheel selection.
- Good results, but not better than the state-of-the-art techniques (e.g., Stochastic Ranking).
- The number of fitness function evaluations required by the approach is high (1,400,000).
- Its main selling point is that the approach does not require of any extra user-defined parameters. Also, the implementation seems relatively simple.
- Other self-adaptive penalty functions have also been proposed (see for example [Tessema & Yen, 2009]).

Biruk Tessema and Gary G. Yen, "**An Adaptive Penalty Formulation for Constrained Evolutionary Optimization**", *IEEE Transactions on Systems, Man and Cybernetics Part A—Systems and Humans*, Vol. 39, No. 3, pp. 565-578, May 2009.

The Adaptive Segregational Constraint Handling Evolutionary Algorithm (ASCHEA) was proposed by Hamida and Schoenauer [2000]. It uses an evolution strategy and it is based on three main components:

- An adaptive penalty function.
- A recombination guided by the constraints.
- A so-called “segregational” selection operator.

S. Ben Hamida and Marc Schoenauer, “**An Adaptive Algorithm for Constrained Optimization Problems**”, in M. Schoenauer et al. (editors), *Proceedings of the 6th Parallel Problem Solving From Nature (PPSN VI)*, pp. 529–538, Springer-Verlag. Lecture Notes in Computer Science Vol. 1917, September 2000.



The **adaptive penalty** adopted is the following:

$$fitness(\vec{x}) = \begin{cases} f(\vec{x}) & \text{if the solution is feasible} \\ f(\vec{x}) - penal(\vec{x}) & \text{otherwise} \end{cases} \quad (12)$$

where

$$penal(\vec{x}) = \alpha \sum_{j=1}^q g_j^+(\vec{x}) + \alpha \sum_{j=q+1}^m |h_j(\vec{x})| \quad (13)$$

where  $g_j^+(\vec{x})$  is the positive part of  $g_j(\vec{x})$  and  $\alpha$  is the penalty factor adopted for all the constraints.

The penalty factor is adapted based on the desired ratio of feasible solutions (with respect to the entire population)  $\tau_{target}$  and the current ratio at generation  $t$   $\tau_t$ :

$$\begin{aligned} \text{if } (\tau_t > \tau_{target}) \quad & \alpha(t+1) = \alpha(t) / fact \\ \text{else} \quad & \alpha(t+1) = \alpha(t) * fact \end{aligned}$$

where  $fact > 1$  and  $\tau_{target}$  are user-defined parameters and

$$\alpha(0) = \left| \frac{\sum_{i=1}^n f_i(\vec{x})}{\sum_{i=1}^n V_i(\vec{x})} \right| * 1000 \quad (14)$$

where  $V_i(\vec{x})$  is the sum of the constraint violation of individual  $i$ .

The **Recombination guided by the constraints** combines an infeasible solution with a feasible one when there are few feasible solutions, based on  $\tau_{target}$ . If  $\tau_t > \tau_{target}$ , then the recombination is performed in the traditional way (i.e., disregarding feasibility).

The **Segregational Selection** operator aims to define a ratio  $\tau_{select}$  of feasible solutions such that they become part of the next generation. The remaining individuals are selected in the traditional way based on their penalized fitness.  $\tau_{select}$  is another user-defined parameter.

- In its most recent version [2002], it uses a penalty factor for each constraint, as to allow more accurate penalties.
- This version also uses niching to maintain diversity (this, however, adds more user-defined parameters).
- The approach requires a high number of fitness function evaluations (1,500,000).

Sana Ben Hamida and Marc Schoenauer, “**ASCEHA: New Results Using Adaptive Segregational Constraint Handling**”, in *Proceedings of the Congress on Evolutionary Computation 2002 (CEC'2002)*, Vol. 1, pp. 884–889, IEEE Press, Piscataway, New Jersey, USA, May 2002.

# Stochastic Ranking

This approach was proposed by Runarsson and Yao [2000], and it consists of a multimembered evolution strategy that uses a penalty function and a selection based on a ranking process.

The idea of the approach is try to balance the influence of the objective function and the penalty function when assigning fitness to an individual.

An interesting aspect of the approach is that it doesn't require the definition of a penalty factor. Instead, the approach requires a user-defined parameter called  $P_f$ , which determines the balance between the objective function and the penalty function.

Thomas P. Runarsson and Xin Yao, “**Stochastic Ranking for Constrained Evolutionary Optimization**, *IEEE Transactions on Evolutionary Computation*, Vol. 4, No. 3, pp. 284–294, September 2000.

## Algorithm

```
Begin
  For i=1 to N
    For j=1 to P-1
      u=random(0,1)
      If ( $\phi(l_j) = \phi(l_{j+1}) = 0$ ) or ( $u < P_f$ )
        If ( $f(l_j) > f(l_{j+1})$ )
          swap( $l_j, l_{j+1}$ )
        Else
          If ( $\phi(l_j) > \phi(l_{j+1})$ )
            swap( $l_j, l_{j+1}$ )
        End For
      If no swap is performed
        break
    End For
  End For
```

# Stochastic Ranking

The population is sorted using an algorithm similar to bubble-sort (which sorts a list based on pairwise comparisons).

Depending on the value of  $P_f$ , the comparison of two adjacent individuals is performed either based only on the objective function values or based on the sum of constraint violation. As it turns out, about half of the time, comparisons are done based only on the objective function values and the rest of the time, the comparisons are based on the sum of constraint violation.

Thus,  $P_f$  introduces the “stochastic” component to the ranking process, so that some solutions may get a good rank even if they are infeasible.

# Stochastic Ranking

- The value of  $P_f$  certainly impacts the performance of the approach. The authors empirically found that  $0.4 < P_f < 0.5$  produces the best results.
- The authors reported the best results found so far for the benchmark adopted with only 350,000 fitness function evaluations.
- The approach is easy to implement.
- Several improvements and variations of Stochastic Ranking have been proposed. For example, Ho and Shimizu [2007] proposed the use of a relaxation of the equality constraints and Garcia et al. [2017] proposed the use of a multiple constraint ranking technique which aims to properly assess constraints with different orders of magnitude and/or different units.

Pei Yee Ho and Kazuyuki Shimizu, “**Evolutionary constrained optimization using an addition of ranking method and a percentage-based tolerance value adjustment scheme**”, *Information Sciences*, Vol. 177, No. 14, pp. 2985–3004, July 15, 2007.

Rafael de Paula Garcia, Beatriz Sousa Leite Pires de Lima, Afonso Celso de Castro Lemonge and Breno Pinheiro Jacob, “**A rank-based constraint handling technique for engineering design optimization problems solved by genetic algorithms**”, *Computers and Structures*, Vol. 187, pp. 77–87, 2017.



# Miscellaneous Approaches

There are two other approaches that we will also briefly discuss because they are very competitive:

- A Simple Multimembered Evolution Strategy (SMES)
- The  $\alpha$  Constrained Method

Efrén Mezura-Montes and Carlos A. Coello Coello,  
“**Constraint-Handling in Nature-Inspired Numerical Optimization: Past, Present and Future**”, *Swarm and Evolutionary Computation*, Vol. 1, No. 4, pp. 173-194, December 2011.

# A Simple Multimembered Evolution Strategy

Mezura-Montes and Coello [2005] proposed an approach based on a  $(\mu + \lambda)$  evolution strategy. Individuals are compared using the following criteria (originally proposed by [Deb, 2000]):

- 1 Between two feasible solutions, the one with the highest fitness value wins.
- 2 If one solution is feasible and the other one is infeasible, the feasible solution wins.
- 3 If both solutions are infeasible, the one with the lowest sum of constraint violation is preferred.

Efrén Mezura-Montes and Carlos A. Coello Coello, “**A Simple Multimembered Evolution Strategy to Solve Constrained Optimization Problems**, *IEEE Transactions on Evolutionary Computation*, Vol. 9, No. 1, pp. 1–17, February 2005.

Kalyanmoy Deb, “**An Efficient Constraint Handling Method for Genetic Algorithms**, *Computer Methods in Applied Mechanics and Engineering*, Vol. 186, Nos. 2–4, pp. 311–338, 2000.

# A Simple Multimembered Evolution Strategy

Additionally, the approach has 3 main mechanisms:

- 1 **Diversity Mechanism:** The infeasible solution which is closest to become feasible is retained in the population, so that it is recombined with feasible solutions.
- 2 **Combined Recombination:** Panmictic recombination is adopted, but with a combination of the discrete and intermediate recombination operators.
- 3 **Stepsize:** The initial stepsize of the evolution strategy is reduced so that finer movements in the search space are favored.

# A Simple Multimembered Evolution Strategy

This approach provided highly competitive results with respect to stochastic ranking, the homomorphous maps and ASCHEA while performing only 240,000 fitness function evaluations.

In a further paper, similar mechanisms were incorporated into a differential evolution algorithm, obtaining even better results.

The approach is easy to implement and robust.

Efrén Mezura-Montes and Jesús Velázquez-Reyes and Carlos A. Coello Coello, “**Modified Differential Evolution for Constrained Optimization**”, in *2006 IEEE Congress on Evolutionary Computation (CEC'2006)*, pp. 332–339, IEEE Press, Vancouver, BC, Canada, July 2006.

# The $\alpha$ Constrained Method

This is a transformation method for constrained optimization introduced by Takahama and Sakai [1999].

Its main idea is to define a satisfaction level for the constraints of a problem. The approach basically adopts a lexicographic order with relaxation of the constraints.

Equality constraints can be easily handled through the relaxation of the constraints.

Tetsuyuki Takahama and Setsuko Sakai, **“Tuning fuzzy control rules by the  $\alpha$  constrained method which solves constrained nonlinear optimization problems”**, *Electronics and Communications in Japan (Part III: Fundamental Electronic Science)*, Vol. 83, No. 9, pp. 1–12, 2000.

# The $\alpha$ Constrained Method

In [Takahama and Sakai, 2005], this approach is coupled to a modified version of Nelder and Mead's method.

The authors argue that Nelder and Mead's method can be seen as an evolutionary algorithm in which the variation operators are: reflection, contraction and expansion.

The authors also extend this method with a boundary mutation operator, the use of multiple simplexes, and a modification to the traditional operators of the method, as to avoid that the method gets easily trapped in a local optimum.

Tetsuyuki Takahama and Setsuko Sakai, "**Constrained Optimization by Applying the  $\alpha$  Constrained Method to the Nonlinear Simplex Method with Mutations**", *IEEE Transactions on Evolutionary Computation*, Vol. 9, No. 5, pp. 437–451, October 2005.

# The $\alpha$ Constrained Method

The approach was validated using a well-known benchmark of 13 test functions.

Results were compared with respect to stochastic ranking. The number of evaluations performed was variable and ranged from 290,000 to 330,000 evaluations in most cases.

The results found were very competitive, although the approach had certain sensitivity to the variation of some of its parameters.

In a further paper [Takahama and Sakai, 2006] the authors introduced another technique for relaxing the constraints, but using a parameter called  $\epsilon$ . In this case, differential evolution is the search engine, and a gradient-based mutation operator is adopted.

Tetsuyuki Takahama and Setsuko Sakai, “**Constrained Optimization by the  $\epsilon$  Constrained Differential Evolution with Gradient-Based Mutation and Feasible Elites**”, in *2006 IEEE Congress on Evolutionary Computation (CEC'2006)*, pp. 308–315, IEEE, Vancouver, BC, Canada, July 2006.

# Special representations and operators

Some researchers have decided to develop special representation schemes to tackle a certain (particularly difficult) problem for which a generic representation scheme (e.g., the binary representation used in the traditional genetic algorithm) might not be appropriate (see for example [Davidor, 1991]).

Due to the change of representation, it is necessary to design special genetic operators that work in a similar way than the traditional operators used with a binary representation. A good example are the Random Keys [Bean, 1994] which allow the use of real numbers to encode permutations of integers.

Yuval Davidor, “**A Genetic Algorithm Applied To Robot Trajectory Generation**”, in Lawrence Davis (editor), *Handbook of Genetic Algorithms*, chapter 12, pp. 144–165. Van Nostrand Reinhold, New York, USA, 1991.

James C. Bean, “**Genetics and random keys for sequencing and optimization**”, *ORSA Journal on Computing*, Vol. 6, No. 2, pp. 154–160, 1994.



# Special representations and operators

A more interesting type of approaches within this group are the so-called “Decoders”.

The emphasis of these approaches is to map chromosomes from the infeasible region into the feasible region of the problem to solve.

In some cases, special operators have also been designed in order to produce offspring that lie on the boundary between the feasible and the infeasible region.

G.R. Raidl and J. Gottlieb, **“On the Importance of Phenotypic Duplicate Elimination in Decoder-Based Evolutionary Algorithms.”**, in S. Brave and A.S. Wu (editors), *Late Breaking Papers at the Genetic and Evolutionary Computation Conference*, pp. 204–211, 1999.

# Special representations and operators

A more intriguing idea is to transform the whole feasible region into a different shape that is easier to explore.

The most important approach designed along these lines are the “homomorphous maps” [Koziel & Michalewicz, 1999].

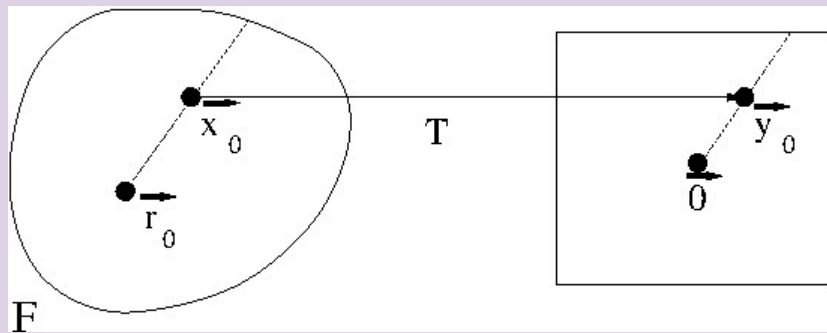
This approach performs a homomorphous mapping between an  $n$ -dimensional cube and a feasible search space (either convex or non-convex).

The main idea of this approach is to transform the original problem into another (topologically equivalent) function that is easier to optimize by an evolutionary algorithm.

Slawomir Koziel and Zbigniew Michalewicz, “**Evolutionary Algorithms, Homomorphous Mappings and Constrained Parameter Optimization**”, *Evolutionary Computation*, Vol. 7, No. 1, pp. 19–44, 1999.

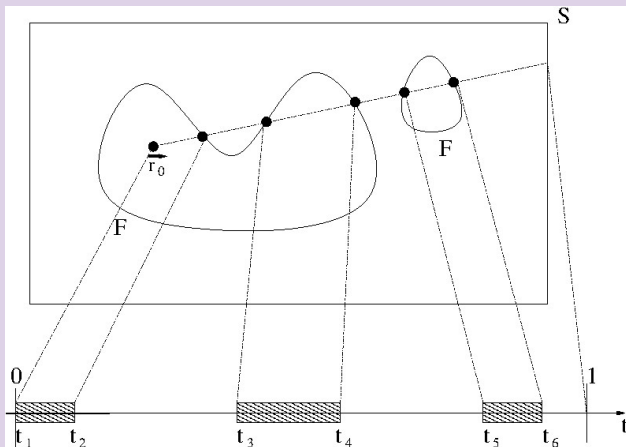
# Special representations and operators

## Convex Case



# Special representations and operators

## Non-Convex Case



# Special representations and operators

The Homomorphous Maps (HM) was for some time, the most competitive constraint-handling approach available (until the publication of Stochastic Ranking).

However, the implementation of the algorithm is more complex, and the experiments reported required a high number of fitness function evaluations (1,400,000).

The version of HM for convex feasible regions is very efficient.

However, the version for non-convex feasible regions requires a parameter  $\nu$  and a binary search procedure to find the intersection of a line with the boundary of the feasible region.

# Repair Algorithms

This sort of approach consists in devising a procedure (or mechanism) that allows to transform an infeasible solution into a feasible one (i.e., we “repair” the infeasible individual).

Such a repaired version can be used either for evaluation only, or it can also replace (with some probability) the original individual in the population.

Liepins and co-workers [1990] showed, through an empirical test of EA performance on a diverse set of constrained-combinatorial optimization problems, that a repair algorithm is able to surpass other approaches in both speed and performance.

G. E. Liepins and Michael D. Vose, “**Representational Issues in Genetic Optimization**”, *Journal of Experimental and Theoretical Computer Science*, Vol. 2, No. 2, pp. 4–30, 1990.

# Repair Algorithms

GENOCOP III [Michalewicz & Nazhiyath, 1995] also uses repair algorithms. The idea is to incorporate the original GENOCOP system [Michalewicz & Janikow, 1991] (which handles only linear constraints) and extend it by maintaining two separate populations, where results in one population influence evaluations of individuals in the other population.

The first population consists of the so-called search points which satisfy linear constraints of the problem; the feasibility (in the sense of linear constraints) of these points is maintained by specialized operators. The second population consists of feasible reference points. Since these reference points are already feasible, they are evaluated directly by the objective function, whereas search points are “repaired” for evaluation.

Zbigniew Michalewicz and G. Nazhiyath, “**Genocop III: A co-evolutionary algorithm for numerical optimization with nonlinear constraints**”, in David B. Fogel (editor), *Proceedings of the Second IEEE International Conference on Evolutionary Computation*, pp. 647–651, IEEE Press, Piscataway, New Jersey, USA, 1995.

Zbigniew Michalewicz and Cezary Z. Janikow, “**Handling Constraints in Genetic Algorithms**”, in R. K. Belew and L. B. Booker (editors), *Proceedings of the Fourth International Conference on Genetic Algorithms (ICGA-91)*, pp. 151–157, Morgan Kaufmann Publishers, San Mateo, California, USA 1991.

# Repair Algorithms

Xiao and co-workers [1995] used a repair algorithm to transform an infeasible path of a robot trying to move between two points in the presence of obstacles, so that the path would become feasible (i.e., collision-free).

The repair algorithm was implemented through a set of carefully designed genetic operators that used knowledge about the domain to bring infeasible solutions into the feasible region in an efficient way.

Other authors that have used repair algorithms are Orvosh and Davis [1994] and Mühlenbein [1992].

Jing Xiao, Zbigniew Michalewicz and Krzysztof Trojanowski, "**Adaptive Evolutionary Planner/Navigator for Mobile Robots**", *IEEE Transactions on Evolutionary Computation*, Vol. 1, No. 1, pp. 18–28, 1997.

David Orvosh and Lawrence Davis, "**Using a Genetic Algorithm to Optimize Problems with Feasibility Constraints**", in *Proceedings of the First IEEE Conference on Evolutionary Computation*, pp. 548–553, IEEE Press, 1994.

Heinz Mühlenbein, "**Parallel Genetic Algorithms in Combinatorial Optimization**", in O. Balci, R. Sharda and S. Zenios (editors), *Computer Science and Operations Research*, pp. 441–456. Pergamon Press, New York, 1992.





# Repair Algorithms

There are no standard heuristics for the design of repair algorithms: normally, it is possible to use a greedy algorithm (i.e., an optimization algorithm that proceeds through a series of alternatives by making the best decision, as computed locally, at each point in the series), a random algorithm or any other heuristic which would guide the repair process.

However, the success of this approach relies mainly on the ability of the user to come up with such a heuristic.

Dirk V. Arnold, “**Analysis of a Repair Mechanism for the  $(1, \lambda)$ -ES Applied to a Simple Constrained Problem**”, in *2011 Genetic and Evolutionary Computation Conference (GECCO'2011)*, pp. 853–860, ACM Press, Dublin, Ireland, July 12-16, 2011.

# Repair Algorithms

Another interesting aspect of this technique is that normally an infeasible solution that is repaired is only used to compute its fitness, but the repaired version is returned to the population only in certain cases (using a certain probability).

The question of replacing repaired individuals is related to the so-called Lamarckian evolution, which assumes that an individual improves during its lifetime and that the resulting improvements are coded back into the chromosome.

Patrick Koch, Samineh Bagheri, Wolfgang Konen, Christophe Foussette, Peter Krause and Thomas Bäck, “**A New Repair Method for Constrained Optimization**”, in *2015 Genetic and Evolutionary Computation Conference (GECCO 2015)*, pp. 273–280, ACM Press, Madrid, Spain, July 11-15, 2015, ISBN 978-1-4503-3472-3.

# Repair Algorithms

Some researchers like Liepins and Potter [1991] have taken the **never** replacing approach (that is, the repaired version is never returned to the population), while other authors such as Nakano [1991] have taken the *always* replacing approach.

Orvosh and Davis [1994] reported a so-called 5% rule for combinatorial optimization problems, which means that EAs (applied to combinatorial optimization problems) with a repairing procedure provide the best result when 5% of the repaired chromosomes replace their infeasible originals.

Gunar E. Liepins and W. D. Potter, “**A Genetic Algorithm Approach to Multiple-Fault Diagnosis**”, in Lawrence Davis (editor), *Handbook of Genetic Algorithms*, chapter 17, pp. 237–250, Van Nostrand Reinhold, New York, New York, 1991.

Ryohei Nakano, “**Conventional Genetic Algorithm for Job Shop**”, in R. K. Belew and L. B. Booker (editors), *Proceedings of the Fourth International Conference on Genetic Algorithms (ICGA-91)*, pp. 474–479, Morgan Kaufmann Publishers, San Mateo, California, USA 1991.

# Repair Algorithms

Michalewicz [1996] has reported, however, that a 15% replacement rule seems to be the best choice for numerical optimization problems with nonlinear constraints.

When an infeasible solution can be easily (or at least at a low computational cost) transformed into a feasible solution, repair algorithms are a good choice.

However this is not always possible and in some cases repair operators may introduce a strong bias in the search, harming the evolutionary process itself [Smith & Tate, 1993].

Zbigniew Michalewicz, **Genetic Algorithms + Data Structures = Evolution Programs**, Springer-Verlag, Third Edition, 1996.

Alice E. Smith and David M. Tate, “**Genetic Optimization Using a Penalty Function**”, in Stephanie Forrest (editor), *Proceedings of the Fifth International Conference on Genetic Algorithms (ICGA-93)*, pp. 499–503, Morgan Kaufmann Publishers, San Mateo, California, USA, July 1993.

# Repair Algorithms

Furthermore, this approach is problem-dependent, since a specific repair algorithm has to be designed for each particular problem.

Also, in its early days, this sort of approach was mostly used in combinatorial optimization problems.

However, in recent years, this has become a relatively active research area (see for example [Salcedo-Sanz, 2009]).

Sancho Salcedo-Sanz, “**A Survey of Repair Methods Used as Constraint Handling Techniques in Evolutionary Algorithms**”, *Computer Science Review*, Vol. 3, No. 3, pp. 175–192, 2009.

# Separation of constraints and objectives

These approaches handle constraints and objectives separately instead of combining them (as done with penalty functions). Some examples are:

- **Coevolution:** Use two populations that interact with each other and have encounters [Paredis, 1994].
- **Superiority of feasible points:** The idea is to assign always a higher fitness to feasible solutions [Powell & Skolnick, 1993].
- **Behavioral memory:** Schoenauer and Xanthakis [1993] proposed to satisfy, sequentially, the constraints of a problem.

Jan Paredis, "**Co-evolutionary Constraint Satisfaction**", in Yuval Davidor, Hans-Paul Schwefel and Reinhard Männer (editors), *Proceedings of the 3rd International Conference on Parallel Problem Solving from Nature (PPSN III)*, pp. 46–55, Jerusalem, Israel, 1994. Springer Verlag.

David Powell and Michael M. Skolnick, "**Using genetic algorithms in engineering design optimization with non-linear constraints**", in Stephanie Forrest (editor), *Proceedings of the Fifth International Conference on Genetic Algorithms (ICGA-93)*, pp. 424–431, Morgan Kaufmann Publishers, San Mateo, California, USA, July 1993.

Marc Schoenauer and Spyros Xanthakis, "**Constrained GA Optimization**", in Stephanie Forrest (editor), *Proceedings of the Fifth International Conference on Genetic Algorithms (ICGA-93)*, pp. 573–580, Morgan Kaufmann Publishers, San Mateo, California, USA, July 1993.



# Separation of constraints and objectives

- **Use of multiobjective optimization concepts:** The main idea in this case is to redefine the single-objective optimization of  $f(\vec{x})$  as a multiobjective optimization problem in which we will have  $m + 1$  objectives, where  $m$  is the total number of constraints. Then, any multi-objective evolutionary algorithm can be adopted [Coello et al., 2007]. Note however, that the use of multiobjective optimization is not straightforward, and several issues have to be taken into consideration.

Carlos A. Coello Coello, Gary B. Lamont and David A. Van Veldhuizen, **Evolutionary Algorithms for Solving Multi-Objective Problems**, Second Edition, Springer, New York, ISBN 978-0-387-33254-3, September 2007.

# Separation of constraints and objectives

This last type of approach has been very popular in the last few years as we will see next.

Surry & Radcliffe [1997] proposed COMOGA (Constrained Optimization by Multiobjective Optimization Genetic Algorithms) where individuals are Pareto-ranked based on the sum of constraint violation. Then, solutions can be selected using binary tournament selection based either on their rank or their objective function value.

Patrick D. Surry and Nicholas J. Radcliffe, “**The COMOGA Method: Constrained Optimisation by Multiobjective Genetic Algorithms**”, *Control and Cybernetics*, Vol. 26, No. 3, pp. 391–412, 1997.



# Separation of constraints and objectives

Zhou et al. [2003] proposed a ranking procedure based on the Pareto Strength concept (introduced in SPEA) for the bi-objective problem, i.e. to count the number of individuals which are dominated for a given solution. Ties are solved by the sum of constraint violation (second objective in the problem). The Simplex crossover (SPX) operator is used to generate a set of offspring where the individual with the highest Pareto strength and also the solution with the lowest sum of constraint violation are both selected to take part in the population for the next generation.

Yuren Zhou, Yuanxiang Li, Jun He and Lishan Kang,  
“**Multi-objective and MGG Evolutionary Algorithm for Constrained Optimization**”, in *Proceedings of the Congress on Evolutionary Computation 2003 (CEC'2003)*, Vol. 1, pp. 1–5, IEEE Press, Piscataway, New Jersey, USA, December 2003.

# Separation of constraints and objectives

Venkatraman and Yen [2005] proposed a generic framework divided in two phases: The first one treats the NLP as a constraint satisfaction problem i.e. the goal is to find at least one feasible solution. To achieve that, the population is ranked based only on the sum of constraint violation. The second phase starts when the first feasible solution was found. At this point, both objectives (original objective function and the sum of constraint violation) are taken into account and nondominated sorting [Deb, 2002] is used to rank the population.

Sangameswar Venkatraman and Gary G. Yen, **“A Generic Framework for Constrained Optimization Using Genetic Algorithms”**, *IEEE Transactions on Evolutionary Computation*, Vol. 9, No. 4, pp. 424–435, August 2005.

# Separation of constraints and objectives

Hernández et al. [2004] proposed an approach named IS-PAES which is based on the Pareto Archive Evolution Strategy (PAES) originally proposed by Knowles and Corne [2000].

IS-PAES uses an external memory to store the best set of solutions found. Furthermore, it adopts a shrinking mechanism to reduce the search space. The multiobjective concept is used in this case as a secondary criterion (Pareto dominance is used only to decide whether or not a new solution is inserted in the external memory).

Arturo Hernández-Aguirre, Salvador Botello-Rionda, Carlos A. Coello Coello, Giovanni Lizárraga-Lizárraga and Efrén Mezura-Montes, **“Handling Constraints using Multiobjective Optimization Concepts”**, *International Journal for Numerical Methods in Engineering*, Vol. 59, No. 15, pp. 1989–2017, April 2004.

# Separation of constraints and objectives

## Possible problems of the use of MO concepts

Runarsson and Yao [2005] presented a comparison of two versions of Pareto ranking in constraint space: (1) considering the objective function value in the ranking process and (2) without considering it. These versions were compared against a typical over-penalized penalty function approach. The authors found that the use of Pareto Ranking leads to bias-free search, then, they concluded that it causes the search to spend most of the time searching in the infeasible region; therefore, the approach is unable to find feasible solutions (or finds feasible solutions with a poor value of the objective function).

Note however, that “pure” Pareto ranking is rarely used as a mechanism to handle constraints, since some bias is normally introduced to the selection mechanism (when a constraint is satisfied, it makes no sense to keep using it in the Pareto dominance relationship).

Thomas Philip Runarsson and Xin Yao, “**Search biases in constrained evolutionary optimization**”, *IEEE Transactions on Systems, Man and Cybernetics Part C—Applications and Reviews*, Vol. 35, No. 2, pp. 233–243, May 2005.

# Hybrid Methods

Within this category, we consider methods that are coupled with another technique (either another heuristic or a mathematical programming approach).

Adeli and Cheng [1994] proposed a hybrid EA that integrates the penalty function method with the primal-dual method. This approach is based on sequential minimization of the Lagrangian method.

Hojjat Adeli and Nai-Tsang Cheng, “**Augmented Lagrangian Genetic Algorithm for Structural Optimization**”, *Journal of Aerospace Engineering*, Vol. 7, No. 1, pp. 104–118, January 1994.

Kim and Myung [1997] proposed the use of an evolutionary optimization method combined with an augmented Lagrangian function that guarantees the generation of feasible solutions during the search process.

## Constrained optimization by random evolution (CORE)

This is an approach proposed by Belur [1997] which combines a random evolution search with Nelder and Mead's method [1965].

J.-H. Kim and H. Myung, “**Evolutionary programming techniques for constrained optimization problems**”, *IEEE Transactions on Evolutionary Computation*, Vol. 1, pp. 129–140, July 1997.

Sheela V. Belur, “**CORE: Constrained Optimization by Random Evolution**”, in John R. Koza (editor), *Late Breaking Papers at the Genetic Programming 1997 Conference*, pp. 280–286, Stanford University, California, July 1997. Stanford Bookstore.

## Ant System (AS)

The main AS algorithm is a multi-agent system where low level interactions between single agents (i.e., artificial ants) result in a complex behavior of the whole ant colony. Although mainly used for combinatorial optimization, AS has also been successfully applied to numerical optimization [Bilchev & Parmee, 1995; Leguizamón, 2004].

Some of the recent research in this area focuses on the exploration of the boundary between the feasible and infeasible regions [Leguizamón & Coello, 2009].

Guillermo Leguizamón and Carlos A. Coello Coello, “**Boundary Search for Constrained Numerical Optimization Problems with an Algorithm Inspired on the Ant Colony Metaphor**”, *IEEE Transactions on Evolutionary Computation*, Vol. 13, No. 2, pp. 350-368, April 2009.

## Simulated Annealing (SA)

Wah & Chen [2001] proposed a hybrid of SA and a genetic algorithm (GA).

The first part of the search is guided by SA. After that, the best solution is refined using a GA. To deal with constraints, Wah & Chen use Lagrangian Multipliers.

Benjamin W. Wah and Yixin Chen, “**Hybrid Constrained Simulated Annealing and Genetic Algorithms for Nonlinear Constrained Optimization**”, in *Proceedings of the 2001 IEEE Congress on Evolutionary Computation CEC'2001*, Vol. 2, pp. 925–932, IEEE Press, Piscataway, New Jersey, USA, May 2001.



## Artificial Immune System (AIS)

Hajela and Lee [1996] proposed a GA hybridized with an AIS (based on the negative selection approach). The idea is to adopt as antigens some feasible solutions and evolve (in an inner GA) the antibodies (i.e., the infeasible solutions) so that they are “similar” (at a genotypic level) to the antigens.

An improved version of this approach was proposed by Coello and Cruz Cortés [2001].

P. Hajela and J. Lee, “**Constrained Genetic Search via Schema Adaptation. An Immune Network Solution**”, *Structural Optimization*, Vol. 12, pp. 11–15, 1996.

Carlos A. Coello Coello and Nareli Cruz Cortés, “**Use of Emulations of the Immune System to Handle Constraints in Evolutionary Algorithms**”, in Cihan H. Dagli et al. (editors), *Intelligent Engineering Systems through Artificial Neural Networks (ANNIE'2001)*, Vol. 11, pp. 141–146, ASME Press, St. Louis Missouri, USA, November 2001.

## Cultural Algorithms

In this sort of approach, the main idea is to preserve beliefs that are socially accepted and discard (or prune) unacceptable beliefs.

The acceptable beliefs can be seen as constraints that direct the population at the micro-evolutionary level. Therefore, constraints can influence directly the search process, leading to an efficient optimization process.

In other words, when using cultural algorithms, some sort of knowledge is extracted during the search process and is used to influence the evolutionary operators as to allow a more efficient search.

## Cultural Algorithms

The first versions of cultural algorithms for constrained optimization had some memory handling problems [Chung & Reynolds, 1996], but later on, they were improved using spatial data structures that allowed to handle problems with any number of decision variables [Coello & Landa, 2002; Landa & Coello, 2006].

Chan-Jin Chung and Robert G. Reynolds, “**A Testbed for Solving Optimization Problems Using Cultural Algorithms**”, in Lawrence J. Fogel, Peter J. Angeline and Thomas Bäck (editors), *Evolutionary Programming V: Proceedings of the Fifth Annual Conference on Evolutionary Programming*, pp. 225–236, MIT Press, Cambridge, Massachusetts, March 1996.

Carlos A. Coello Coello and Ricardo Landa Becerra, “**Adding Knowledge and Efficient Data Structures to Evolutionary Programming: A Cultural Algorithm for Constrained Optimization**”, in W.B. Langdon et al. (editors), *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2002)*, pp. 201–209, Morgan Kaufmann Publishers, San Francisco, California, USA, July 2002.

Ricardo Landa Becerra and Carlos A. Coello Coello, “**Cultured differential evolution for constrained optimization**”, *Computer Methods in Applied Mechanics and Engineering*, Vol. 195, Nos. 33–36, pp. 4303–4322, July 1, 2006.

# Recent Ideas: An Ensemble of Constraint-Handling Techniques

Mallipeddi and Suganthan [2010] proposed the use of an ensemble of constraint-handling techniques.

The idea is to have several constraint-handling technique, each with its own population and parameters. Each population produces its offspring and evaluates them. However, the offspring compete not only against its own population, but also against the others. Thus, a certain offspring could be rejected by its population, while being accepted by another population.

This intends to automate the selection of the best constraint-handling technique for a certain problem, based on the fact that none of them will be best in all cases (remember the No-Free-Lunch theorem!).

Rammohan Mallipeddi and Ponnuthurai N. Suganthan, "**Ensemble of Constraint Handling Techniques**", *IEEE Transactions On Evolutionary Computation*, Vol. 14, No. 4, pp. 561–579, August 2010.

# Recent Ideas: Use of Gradient-Based Information

There have been a few proposals that incorporate gradient information, either from the fitness values or from the constraints.

Sun and Garibaldi [2010] proposed an Estimation of Distribution Algorithm combined with a gradient-based local optimizer.

Jianyong Sun and Jonathan M. Garibaldi, “**A Novel Memetic Algorithm for Constrained Optimization**”, in *2010 IEEE Congress on Evolutionary Computation (CEC'2010)*, pp. 549–556, IEEE Press, Barcelona, Spain, July 18–23, 2010.

# Recent Ideas: Use of Gradient-Based Information

Handoko et al. [2010] proposed a memetic algorithm that combines a genetic algorithm, sequential quadratic programming with second-order functional approximations, and a technique for modeling the feasibility region through the use of support vector machines.

Hamza et al. [2014] proposed a consensus-based variant of a genetic algorithm which is combined with sequential quadratic programming, but using gradient information from the constraints.

Stephanus Daniel Handoko, Chee Keong Kwoh and Yew-Soon Ong, **“Feasibility Structure Modeling: An Effective Chaperone for Constrained Memetic Algorithms”**, *IEEE Transactions on Evolutionary Computation*, Vol. 14, No. 5, pp. 740–758, October 2010.

Noha M. Hamza, Ruhul A. Sarker, Daryl L. Essam, Kalyanmoy Deb and Saber M. Elsayed, **“A Constraint Consensus Memetic Algorithm for Solving Constrained Optimization Problems”**, *Engineering Optimization*, Vol. 46, No. 11, pp. 1447–1464, November 2014.

# Recent Ideas: Memetic Viability

Maesani et al. [2016] proposed the use of a notion called *viability evolution*, which emphasizes the elimination of solutions that do not satisfy viability criteria (i.e., boundaries on objectives and constraints).

Such boundaries are adapted during the search in a way analogous to other approaches such as ASCHEA. However, in this case, this approach (which is based on the use of the covariance matrix adaptation evolution strategy (CMA-ES)) produces a population of local search engines which can be recombined using differential evolution.

An interesting aspect of this approach is that it uses an adaptive scheduler that toggles between exploration and exploitation by selecting to advance one of the local search units (or individuals) or to recombine them.

Andrea Maesani, Giovanni Iacca and Dario Floreano, "**Memetic Viability Evolution for Constrained Optimization**", *IEEE Transactions on Evolutionary Computation*, Vol. 20, No. 1, pp. 125–144, February 2016.

# Test Functions

Michalewicz and Schoenauer [1996] proposed a set of test functions, which was later expanded by Runarsson and Yao [2000].

The full set consists of 13 test functions which contain characteristics that are representative of what can be considered “difficult” global optimization problems for an evolutionary algorithm.

Zbigniew Michalewicz and Marc Schoenauer, “**Evolutionary Algorithms for Constrained Parameter Optimization Problems**”, *Evolutionary Computation*, Vol. 4, No. 1, pp. 1–32, 1996.

Thomas P. Runarsson and Xin Yao, “**Stochastic Ranking for Constrained Evolutionary Optimization**”, *IEEE Transactions on Evolutionary Computation*, Vol. 4, No. 3, pp. 284–294, September 2000.



Note however, that many other test functions exist. See for example:

- Mezura Montes, Efrén and Coello Coello, Carlos A., **What Makes a Constrained Problem Difficult to Solve by an Evolutionary Algorithm**, Technical Report EVOCINV-01-2004, Evolutionary Computation Group at CINVESTAV, Sección de Computación, Departamento de Ingeniería Eléctrica, CINVESTAV-IPN, México, February 2004.
- C. A. Floudas and P. M. Pardalos, **A Collection of Test Problems for Constrained Global Optimization Algorithms**, Number 455 in Lecture Notes in Computer Science. Springer-Verlag, 1990.
- Christodoulos A. Floudas et al. (editors), **Handbook of Test Problems in Local and Global Optimization**, Kluwer Academic Publishers, Dordrecht, 1999.
- Takehisa Kohira, Takehisa Kohira, Akira Oyama and Takehisa Kohira, **“Proposal of benchmark problem based on real-world car structure design optimization”**, *GECCO '18 Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pp. 183–184, ACM Press, Kyoto, Japan, July 15-19, 2018.

# Test Functions

## Classical Benchmark

Problem	n	Type of function	$\rho$	LI	NI	LE	NE
g01	13	quadratic	0.0003%	9	0	0	0
g02	20	nonlinear	99.9973%	1	1	0	0
g03	10	nonlinear	0.0026%	0	0	0	1
g04	5	quadratic	27.0079%	0	6	0	0
g05	4	nonlinear	0.0000%	2	0	0	3
g06	2	nonlinear	0.0057%	0	2	0	0
g07	10	quadratic	0.0000%	3	5	0	0
g08	2	nonlinear	0.8581%	0	2	0	0
g09	7	nonlinear	0.5199%	0	4	0	0
g10	8	linear	0.0020%	3	3	0	0
g11	2	quadratic	0.0973%	0	0	0	1
g12	3	quadratic	4.7697%	0	9 <sup>3</sup>	0	0
g13	5	nonlinear	0.0000%	0	0	1	2

# Test Functions

Additional test functions have been proposed, and a new benchmark, consisting of 24 test functions (which includes the 13 indicated in the previous slide) is now more popular.

J. J. Liang, T. P. Runarsson, E. Mezura-Montes, M. Clerc, P. N. Suganthan, C. A. Coello Coello, K. Deb, **Problem Definitions and Evaluation Criteria for the CEC 2006, Special Session on Constrained Real-Parameter Optimization**, Technical Report, Nanyang Technological University, Singapore, 2006.

There is also a set of test problems that were introduced at the *2010 World Congress on Computational Intelligence* (WCCI'2010). This set consists of 18 scalable test problems.

# Test Case Generators

Michalewicz et al. [2000] proposed a Test Case Generator for constrained parameter optimization techniques.

This generator allows to build test problems by varying several features such as: dimensionality, multimodality, number of constraints, connectedness of the feasible region, size of the feasible region with respect to the whole search space and ruggedness of the objective function.

Zbigniew Michalewicz, Kalyanmoy Deb, Martin Schmidt and Thomas Stidsen, **“Test-Case Generator for Nonlinear Continuous Parameter Optimization Techniques”**, *IEEE Transactions on Evolutionary Computation*, Vol. 4, No. 3, pp. 197–215, September 2000.

# Test Case Generators

The first version of this test problems generator had some problems because the functions produced were symmetric.

This motivated the development of a new version called TCG-2 [Schmidt et al., 2000].

Martin Schmidt and Zbigniew Michalewicz, “**Test-Case Generator TCG-2 for Nonlinear Parameter Optimization**”, in M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J.J. Merelo and H.-P. Schwefel (editors), *Proceedings of 6th Parallel Problem Solving From Nature (PPSN VI)*, pp. 539–548, Springer-Verlag. Lecture Notes in Computer Science Vol. 1917, Heidelberg, Germany, September 2000.

# Some Recommendations

- Study (and try first) traditional mathematical programming techniques (e.g., gradient-based methods, Nelder-Mead, Hooke-Jeeves, etc.).
- If interested in numerical optimization, try evolution strategies or differential evolution, instead of using genetic algorithms. Also, the combination of parents and offspring in the selection process tends to produce better performance.
- Pay attention to diversity. Keeping populations in which every individual is feasible is not always a good idea.
- Normalizing the constraints of the problem is normally a good idea.

# Current Research Topics

- New constraint-handling approaches (e.g., based on multiobjective optimization concepts).
- Large-scale constrained optimization [Aguilar-Justo & Mezura-Montes, 2016].
- Use of voting schemes as a way of incorporating preferences for allowing the exploration of a portion of the infeasible region [Yu et al., 2014].

Adan E. Aguilar-Justo and Efrén Mezura-Montes, **“Towards an Improvement of Variable Interaction Identification for Large-Scale Constrained Problems”**, in *2016 IEEE Congress on Evolutionary Computation (CEC'2016)*, pp. 4167–4174, IEEE Press, Vancouver, Canada, July 24-29, 2016, ISBN 978-1-5090-0623-9.

Erdong Yu, Qing Fei, Hongbin Ma and Qingbo Geng, **“Improving Constraint Handling for Multiobjective Particle Swarm Optimization”**, in *Proceedings of the 33rd Chinese Control Conference*, pp. 8622–8627, IEEE Press, Nanjing, China, July 28-30, 2014.

# Current Research Topics

- Old constraint-handling techniques with new search engines (e.g., differential evolution, particle swarm optimization, ant colony, etc.).
- Constraint-handling techniques for multi-objective evolutionary algorithms [Yen, 2009; Fukumoto & Oyama, 2018].

Gary G. Yen, “**An Adaptive Penalty Function for Handling Constraint in Multi-objective Evolutionary Optimization**”, in Efrén Mezura-Montes (editor), *Constraint-Handling in Evolutionary Computation*, Chapter 6, pp. 121–143, Springer. Studies in Computational Intelligence, Volume 198, Berlin, 2009. ISBN 978-3-642-00618-0.

Hiroaki Fukumoto and Akira Oyama, “**A Generic Framework for Incorporating Constraint Handling Techniques into Multi-Objective Evolutionary Algorithms**”, in Kevin Sim and Paul Kaufmann (editors), *Applications of Evolutionary Computation 21st International Conference, EvoApplications 2018*, pp. 634–649, Springer-Verlag, Lecture Notes in Computer Science Vol. 10784, Parma, Italy, April 4-6, 2018.



# Current Research Topics

- Self-adaptation mechanisms for constrained optimization [Brest, 2009].
- Equality constraint-handling in multi-objective optimization [Saha & Ray, 2012].
- Time complexity analysis of evolutionary algorithms used for solving constrained problems (particularly, the role of penalty factors in the time complexity of an EA) [Zhou & He, 2007].

Janez Brest and Viljem Zumer and Mirjam Sepesy Maucec, “**Self-Adaptive Differential Evolution Algorithm in Constrained Real-Parameter Optimization**”, in *2006 IEEE Congress on Evolutionary Computation (CEC'2006)*, pp. 919–926, IEEE, Vancouver, BC, Canada, July 2006.

Amit Saha and Tapabrata Ray, “**Equality Constrained Multi-objective Optimization**”, in *2012 IEEE Congress on Evolutionary Computation (CEC'2012)*, pp. 47–53, IEEE Press, Brisbane, Australia, June 10-15, 2012.

Yuren Zhou and Jun He, “**A Runtime Analysis of Evolutionary Algorithms for Constrained Optimization Problems**”, *IEEE Transactions on Evolutionary Computation*, Vol. 11, No. 5, pp. 608–619, October 2007

# Current Research Topics

- Hybrids of EAs with mathematical programming techniques (e.g., evolution strategy + simplex, use of Lagrange multipliers, etc.). See for example [Mehta and Dasgupta, 2012].
- Approaches that reduce the number of objective function evaluations performed (e.g., surrogate models, fitness inheritance, fitness approximation).

Dirk V. Arnold and Jeremy Porter, “**Towards an Augmented Lagrangian Constraint-Handling Approach for the (1+1)-ES**”, in *2015 Genetic and Evolutionary Computation Conference (GECCO 2015)*, pp. 249–256, ACM Press, Madrid, Spain, July 11-15, 2015, ISBN 978-1-4503-3472-3.

Samineh Bagheri, Wolfgang Konen and Thomas Bäck, “**Equality Constraint Handling for Surrogate-Assisted Constrained Optimization**”, in *2016 IEEE Congress on Evolutionary Computation (CEC’2016)*, pp. 1924–1931, IEEE Press, Vancouver, Canada, July 24-29, 2016, ISBN 978-1-5090-0623-9.

# Current Research Topics

- Constrained robust optimization [Tagawa and Miyanaga, 2017].
- Stopping criteria for constrained optimization using evolutionary algorithms [Zielinski & Laur, 2008].
- Special operators for exploring the boundary between the feasible and infeasible regions [Leguizamón & Coello, 2009].
- Dynamic constraints [Nguyen & Yao, 2012].

Kiyoharu Tagawa and Shun Miyanaga, "**Weighted empirical distribution based approach to Chance Constrained Optimization Problems using Differential Evolution**", in *2017 IEEE Congress on Evolutionary Computation (CEC'2017)*, pp. 97–104, IEEE Press, San Sebastián, Spain, June 5-8, 2017, ISBN 978-1-5090-4601-0.

Karin Zielinski and Rainer Laur, "**Stopping Criteria for Differential Evolution in Constrained Single-Objective Optimization**", in Uday K. Chakraborty (Editor), *Advances in Differential Evolution*, pp. 111–138, Springer, Berlin, 2008, ISBN 978-3-540-68827-3.

Guillermo Leguizamón and Carlos A. Coello Coello, "**Boundary Search for Constrained Numerical Optimization Problems with an Algorithm Inspired on the Ant Colony Metaphor**", *IEEE Transactions on Evolutionary Computation*, Vol. 13, No. 2, pp. 350-368, April 2009.

Trung Thanh Nguyen and Xin Yao, "**Continuous Dynamic Constrained Optimization—The Challenges**", *IEEE Transactions on Evolutionary Computation*, Vol. 16, No. 6, pp. 769–786, December 2012.

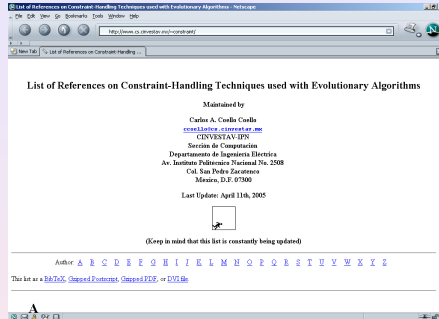
# Some Ideas

- We need new test problems and test problem generators. For example, test problems with equality constraints as well as constrained multi-objective optimization problems are required.
- It is important to have new metrics that allow us to assess the online performance of a constraint-handling technique and to characterize constrained search spaces (see for example the notion of *violation landscape* proposed in [Malan et al., 2015]).
- New constraint-handling techniques are still possible, but are rare these days. But what about understanding the limitations of existing constraint-handling techniques? See for example: [Arnold, 2011].

Katherine M. Malan, Johannes F. Oberholzer and Andries P. Engelbrecht, "**Characterising Constrained Continuous Optimisation Problems**", in *2015 IEEE Congress on Evolutionary Computation (CEC'2015)*, pp. 1351–1358, IEEE Press, Sendai, Japan, 25–28 May 2015, ISBN 978-1-4799-7492-4.

Dirk V. Arnold, "**Analysis of a Repair Mechanism for the  $(1, \lambda)$ -ES Applied to a Simple Constrained Problem**", in *2011 Genetic and Evolutionary Computation Conference (GECCO'2011)*, pp. 853–860, ACM Press, Dublin, Ireland, July 12–16, 2011.

# To know more about constraint-handling techniques used with EAs

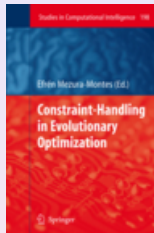


Please visit our constraint-handling repository located at:

**<http://www.cs.cinvestav.mx/~constraint>**

The repository currently (as of April 17th, 2019) has **1438** bibliographic references.

# Suggested Readings



Efrén Mezura-Montes (editor), **Constraint-Handling in Evolutionary Optimization**, Springer, 2009, ISBN: 978-3-642-00618-0.

# Suggested Readings



Rituparna Datta and Kalyanmoy Deb (editors), **Evolutionary Constrained Optimization**, Springer, 2015, ISBN: 978-81-322-2183-8.