Constraint-Handling in Nature-Inspired Optimization

Efrén Mezura-Montes

Artificial Intelligence Research Center
University of Veracruz, MEXICO
emezura@uv.mx
http://www.uv.mx/personal/emezura

2017 IEEE Congress on Evolutionary Computation
Donostia - San Sebastián, SPAIN, June 5th, 2017
Outline

1 Introduction
   The problem of interest
   Some important concepts
   Mathematical-programming methods
   Why alternative methods?

2 The early years
   Penalty functions
   Decoders
   Special operators
   Separation of objective function and constraints
   General comments

3 Current constraint-handling techniques
   Feasibility rules
   Stochastic ranking
   \(\varepsilon\)-constrained method
   Novel penalty functions
   Novel special operators
   Multi-objective concepts
   Ensemble of constraint-handling techniques

4 Summary and current trends
   A bird’s eye view
   Current trends
1 Introduction
   - The problem of interest
   - Some important concepts
   - Mathematical-programming methods
   - Why alternative methods?

2 The early years
   - Penalty functions
   - Decoders
   - Special operators
   - Separation of objective function and constraints
   - General comments

3 Current constraint-handling techniques
   - Feasibility rules
   - Stochastic ranking
   - $\varepsilon$-constrained method
   - Novel penalty functions
   - Novel special operators
   - Multi-objective concepts
   - Ensemble of constraint-handling techniques

4 Summary and current trends
   - A bird’s eye view
   - Current trends
Outline

1. Introduction
   - The problem of interest
   - Some important concepts
   - Mathematical-programming methods
   - Why alternative methods?

2. The early years
   - Penalty functions
   - Decoders
   - Special operators
   - Separation of objective function and constraints
   - General comments

3. Current constraint-handling techniques
   - Feasibility rules
   - Stochastic ranking
   - $\varepsilon$-constrained method
   - Novel penalty functions
   - Novel special operators
   - Multi-objective concepts
   - Ensemble of constraint-handling techniques

4. Summary and current trends
   - A bird’s eye view
   - Current trends
Problem statement

Constrained numerical optimization problem (CNOP)

Find $\vec{x}$ which minimizes $f(\vec{x})$

subject to:

$g_i(\vec{x}) \leq 0, \quad i = 1, \ldots, m$

$h_j(\vec{x}) = 0, \quad j = 1, \ldots, p$

- $\vec{x} \in \mathbb{R}^n$ is the vector of solutions $\vec{x} = [x_1, x_2, \ldots, x_n]^T$.
- Each $x_k, k = 1, \ldots, n$ is bounded by lower and upper limits $L_k \leq x_k \leq U_k$ which define the search space $S$.
- $F$ comprises the set of all solutions which satisfy the constraints of the problems and it is called the feasible region.
- To handle equality constraints they are transformed into inequality constraints as follows: $|h_j(\vec{x})| - \varepsilon \leq 0$. 
Problem statement

Constrained numerical optimization problem (CNOP)

Find $\vec{x}$ which minimizes $f(\vec{x})$

subject to:

$g_i(\vec{x}) \leq 0, \ i = 1, \ldots, m$

$h_j(\vec{x}) = 0, \ j = 1, \ldots, p$

- $\vec{x} \in \mathbb{R}^n$ is the vector of solutions $\vec{x} = [x_1, x_2, \ldots, x_n]^T$.
- Each $x_k, k = 1, \ldots, n$ is bounded by lower and upper limits $L_k \leq x_k \leq U_k$ which define the search space $S$.
- $\mathcal{F}$ comprises the set of all solutions which satisfy the constraints of the problems and it is called the feasible region.
- To handle equality constraints they are transformed into inequality constraints as follows: $|h_j(\vec{x})| - \varepsilon \leq 0$. 

Efrén Mezura-Montes
CEC 2017, SPAIN
Problem statement

Constrained numerical optimization problem (CNOP)

Find $\bar{x}$ which minimizes $f(\bar{x})$

subject to:

$g_i(\bar{x}) \leq 0, \quad i = 1, \ldots, m$

$h_j(\bar{x}) = 0, \quad j = 1, \ldots, p$

- $\bar{x} \in \mathbb{R}^n$ is the vector of solutions $\bar{x} = [x_1, x_2, \ldots, x_n]^T$.
- Each $x_k$, $k = 1, \ldots, n$ is bounded by lower and upper limits $L_k \leq x_k \leq U_k$ which define the search space $S$.
- $\mathcal{F}$ comprises the set of all solutions which satisfy the constraints of the problems and it is called the feasible region.
- To handle equality constraints they are transformed into inequality constraints as follows: $|h_j(\bar{x})| - \varepsilon \leq 0$. 

Problem statement

Constrained numerical optimization problem (CNOP)

Find \( \vec{x} \) which minimizes \( f(\vec{x}) \)

subject to:

\[ g_i(\vec{x}) \leq 0, \quad i = 1, \ldots, m \]
\[ h_j(\vec{x}) = 0, \quad j = 1, \ldots, p \]

- \( \vec{x} \in \mathbb{R}^n \) is the vector of solutions \( \vec{x} = [x_1, x_2, \ldots, x_n]^T \).
- Each \( x_k, \quad k = 1, \ldots, n \) is bounded by lower and upper limits \( L_k \leq x_k \leq U_k \) which define the search space \( S \).
- \( \mathcal{F} \) comprises the set of all solutions which satisfy the constraints of the problems and it is called the feasible region.
- To handle equality constraints they are transformed into inequality constraints as follows: \( |h_j(\vec{x})| - \varepsilon \leq 0 \).
Constrained numerical optimization problem

Constrained search space
Outline

1. Introduction
   - The problem of interest
   - Some important concepts
   - Mathematical-programming methods
   - Why alternative methods?

2. The early years
   - Penalty functions
   - Decoders
   - Special operators
   - Separation of objective function and constraints
   - General comments

3. Current constraint-handling techniques
   - Feasibility rules
   - Stochastic ranking
   - $\varepsilon$-constrained method
   - Novel penalty functions
   - Novel special operators
   - Multi-objective concepts
   - Ensemble of constraint-handling techniques

4. Summary and current trends
   - A bird’s eye view
   - Current trends
Feasible global optimum

In the following definitions we will assume minimization (without loss of generality). \( \vec{x}^* = [x_1^*, x_2^*, \ldots, x_n^*]^T \) refers to the feasible optimum point and its corresponding value of the objective function \( f(\vec{x}^*) \) is called the feasible optimum value. The pair \( \vec{x}^* \) and \( f(\vec{x}^*) \) is called feasible optimum solution.
A function \( f(\vec{x}) \) defined on a set \( S \) attains its feasible global minimum at a point \( \vec{x}^* \in \mathcal{F} \subseteq S \) if and only if: 
\[
 f(\vec{x}^*) \leq f(\vec{x}), \quad \forall \vec{x} \in \mathcal{F} \subseteq S.
\]
Kuhn and Tucker developed the necessary and sufficient optimality conditions for the CNOP assuming that the functions $f$, $g_i$, and $h_j$, are differentiable or twice-differentiable.

These optimality conditions, commonly known as the Kuhn-Tucker conditions (KTC) consist of finding a solution to a system of nonlinear equations.

However, it is quite difficult that KTC hold for real-world problems. Therefore, the CNOP is an open-problem.
Kuhn and Tucker developed the necessary and sufficient optimality conditions for the CNOP assuming that the functions $f$, $g_i$, and $h_j$, are differentiable or twice-differentiable.

These optimality conditions, commonly known as the Kuhn-Tucker conditions (KTC) consist of finding a solution to a system of nonlinear equations.

However, it is quite difficult that KTC hold for real-world problems. Therefore, the CNOP is an open-problem.
Kuhn and Tucker developed the necessary and sufficient optimality conditions for the CNOP assuming that the functions $f$, $g_i$, and $h_j$, are differentiable or twice-differentiable.

These optimality conditions, commonly known as the *Kuhn-Tucker conditions* (KTC) consist of finding a solution to a system of nonlinear equations.

However, it is quite difficult that KTC hold for real-world problems. Therefore, the CNOP is an open-problem.
Outline

1. Introduction
   - The problem of interest
   - Some important concepts
   - **Mathematical-programming methods**
     - Why alternative methods?

2. The early years
   - Penalty functions
   - Decoders
   - Special operators
   - Separation of objective function and constraints
   - General comments

3. Current constraint-handling techniques
   - Feasibility rules
   - Stochastic ranking
   - $\varepsilon$-constrained method
   - Novel penalty functions
   - Novel special operators
   - Multi-objective concepts
   - Ensemble of constraint-handling techniques

4. Summary and current trends
   - A bird’s eye view
   - Current trends
They usually require just one solution which is improved during the process.

Two categories:
- Direct Methods.
- Indirect Methods.
Two categories

- They usually require just one solution which is improved during the process.
- Two categories:
  - Direct Methods.
  - Indirect Methods.
These methods use only the information of the objective function to find search directions.
Indirect methods

These methods require that the objective function is differentiable or twice differentiable so as to use such information to guide the search.
Outline

1. Introduction
   - The problem of interest
   - Some important concepts
   - Mathematical-programming methods
   - Why alternative methods?

2. The early years
   - Penalty functions
   - Decoders
   - Special operators
   - Separation of objective function and constraints
   - General comments

3. Current constraint-handling techniques
   - Feasibility rules
   - Stochastic ranking
   - $\epsilon$-constrained method
   - Novel penalty functions
   - Novel special operators
   - Multi-objective concepts
   - Ensemble of constraint-handling techniques

4. Summary and current trends
   - A bird’s eye view
   - Current trends
Despite the large number of mathematical programming methods developed, several optimization problems present characteristics that make them difficult to solve using this kind of algorithms.
Difficulties found

- Problems with non-differentiable objective functions and/or non-differentiable constraints.
- Problems with disjoint feasible regions.
- Problems with objective function and/or constraints not available in algebraic form.
- Problems in which the Kuhn-Tucker conditions for optimality do not hold.
- Problems where no mathematical programming technique can guarantee convergence to the global optimum.
- Huge search spaces.
Difficulties found

- Problems with non-differentiable objective functions and/or non-differentiable constraints.
- Problems with disjoint feasible regions
  - Problems with objective function and/or constraints not available in algebraic form.
  - Problems in which the Kuhn-Tucker conditions for optimality do not hold.
  - Problems where no mathematical programming technique can guarantee convergence to the global optimum.
- Huge search spaces.
Difficulties found

- Problems with non-differentiable objective functions and/or non-differentiable constraints.
- Problems with disjoint feasible regions.
- Problems with objective function and/or constraints not available in algebraic form.
- Problems in which the Kuhn-Tucker conditions for optimality do not hold.
- Problems where no mathematical programming technique can guarantee convergence to the global optimum.
- Huge search spaces.
Difficulties found

- Problems with non-differentiable objective functions and/or non-differentiable constraints.
- Problems with disjoint feasible regions.
- Problems with objective function and/or constraints not available in algebraic form.
- Problems in which the Kuhn-Tucker conditions for optimality do not hold.
- Problems where no mathematical programming technique can guarantee convergence to the global optimum.
- Huge search spaces.
Problems with non-differentiable objective functions and/or non-differentiable constraints.

Problems with disjoint feasible regions

Problems with objective function and/or constraints not available in algebraic form.

Problems in which the Kuhn-Tucker conditions for optimality do not hold.

Problems where no mathematical programming technique can guarantee convergence to the global optimum.

Huge search spaces.
Difficulties found

- Problems with non-differentiable objective functions and/or non-differentiable constraints.
- Problems with disjoint feasible regions.
- Problems with objective function and/or constraints not available in algebraic form.
- Problems in which the Kuhn-Tucker conditions for optimality do not hold.
- Problems where no mathematical programming technique can guarantee convergence to the global optimum.
- Huge search spaces.
Nature-inspired algorithms (NIAs)

- Evolutionary algorithms (EAs) and swarm intelligence algorithms (SIAs) (grouped as NIAs) are popular meta-heuristics approaches used to solve complex optimization problems.
  - NIAs are designed to deal with unconstrained search spaces.
  - The design and addition of a constraint-handling techniques into a NIA to deal with a constrained search space is an open problem.
Nature-inspired algorithms (NIAs)

- Evolutionary algorithms (EAs) and swarm intelligence algorithms (SIAs) (grouped as NIAs) are popular meta-heuristics approaches used to solve complex optimization problems.
- NIAs are designed to deal with unconstrained search spaces.
- The design and addition of a constraint-handling techniques into a NIA to deal with a constrained search space is an open problem.
Nature-inspired algorithms (NIAs)

- Evolutionary algorithms (EAs) and swarm intelligence algorithms (SIAs) (grouped as NIAs) are popular meta-heuristics approaches used to solve complex optimization problems.
- NIAs are designed to deal with unconstrained search spaces.
- The design and addition of a constraint-handling techniques into a NIA to deal with a constrained search space is an open problem.
Main components of a nature-inspired algorithm

1. Solution encoding.
2. Fitness function.
3. Initial population.
5. Variation operators (crossover & mutation).
6. Replacement.
Why the search must change?

Unconstrained optimization problem

$$\text{Min: } f(\mathbf{x}) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2)^2$$

Why the search must change?

Unconstrained optimization problem

Why the search must change?

Constrained optimization problem

Min: $f(\bar{x}) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2)^2$
subject to:
$(x_1 - 5)^2 + x_2^2 - 26 \geq 0$

Why a constraint-handling technique?

- The initial population (usually generated at random) may contain several (if not all) infeasible solutions, and it may be difficult to generate only feasible solutions from the beginning.
- The information about feasibility must be incorporated into the fitness function to bias the search to the feasible region.
- The parent selection and/or replacement must distinguish between feasible and infeasible solutions.
- The variation operators are blind with respect to the constraints of the optimization problem.
The initial population (usually generated at random) may contain several (if not all) infeasible solutions, and it may be difficult to generate only feasible solutions from the beginning.

The information about feasibility must be incorporated into the fitness function to bias the search to the feasible region.

The parent selection and/or replacement must distinguish between feasible and infeasible solutions.

The variation operators are blind with respect to the constraints of the optimization problem.
Why a constraint-handling technique?

- The initial population (usually generated at random) may contain several (if not all) infeasible solutions, and it may be difficult to generate only feasible solutions from the beginning.
- The information about feasibility must be incorporated into the fitness function to bias the search to the feasible region.
- The parent selection and/or replacement must distinguish between feasible and infeasible solutions.
- The variation operators are blind with respect to the constraints of the optimization problem.
Why a constraint-handling technique?

- The initial population (usually generated at random) may contain several (if not all) infeasible solutions, and it may be difficult to generate only feasible solutions from the beginning.
- The information about feasibility must be incorporated into the fitness function to bias the search to the feasible region.
- The parent selection and/or replacement must distinguish between feasible and infeasible solutions.
- The variation operators are blind with respect to the constraints of the optimization problem.
Two classifications were proposed: one by Michalewicz and Schoenauer [96] and another one by Coello [18].

- Both taxonomies agreed on penalty functions as a particular class.
- This new classification for earlier methods is based on constraint-handling mechanisms, whereas the search algorithm employed is discussed as a separate issue.
Two classifications were proposed: one by Michalewicz and Schoenauer [96] and another one by Coello [18].

Both taxonomies agreed on penalty functions as a particular class.

This new classification for earlier methods is based on constraint-handling mechanisms, whereas the search algorithm employed is discussed as a separate issue.
Two classifications were proposed: one by Michalewicz and Schoenauer [96] and another one by Coello [18].

Both taxonomies agreed on penalty functions as a particular class.

This new classification for earlier methods is based on constraint-handling mechanisms, whereas the search algorithm employed is discussed as a separate issue.
Introduction
- The problem of interest
- Some important concepts
- Mathematical-programming methods
- Why alternative methods?

The early years
- Penalty functions
- Decoders
- Special operators
- Separation of objective function and constraints
- General comments

Current constraint-handling techniques
- Feasibility rules
- Stochastic ranking
- $\varepsilon$-constrained method
- Novel penalty functions
- Novel special operators
- Multi-objective concepts
- Ensemble of constraint-handling techniques

Summary and current trends
- A bird’s eye view
- Current trends
Penalty functions

Definition

Based on mathematical programming approaches, where a CNOP is transformed into an unconstrained numerical optimization problem, NIAs have adopted penalty functions, whose general formula is the following:

$$\phi(\vec{x}) = f(\vec{x}) + p(\vec{x})$$

where $$\phi(\vec{x})$$ is the expanded objective function to be optimized, and $$p(\vec{x})$$ is the penalty value that can be calculated as follows:

$$p(\vec{x}) = \sum_{i=1}^{m} r_i \cdot \max(0, g_i(\vec{x}))^2 + \sum_{j=1}^{p} c_j \cdot |h_j(\vec{x})|$$

where $$r_i$$ and $$c_j$$ are positive constants called “penalty factors”.
Penalty functions

Pros and cons

- The aim is to decrease the fitness of infeasible solutions.
- Unlike mathematical programming approaches, where interior and exterior penalty functions are employed, NIAs have mainly focused on the last ones.
- Their implementation is quite simple ... but,
- Penalty functions require a careful fine-tuning of their penalty factors.
- Such values usually are highly problem-dependent.
- Different approaches have been proposed to tackle this shortcoming.
Penalty functions

Pros and cons

- The aim is to decrease the fitness of infeasible solutions.
- Unlike mathematical programming approaches, where interior and exterior penalty functions are employed, NIAs have mainly focused on the last ones.
- Their implementation is quite simple ... but,
- Penalty functions require a careful fine-tuning of their penalty factors.
- Such values usually are highly problem-dependent.
- Different approaches have been proposed to tackle this shortcoming.
Penalty functions

Pros and cons

- The aim is to decrease the fitness of infeasible solutions.
- Unlike mathematical programming approaches, where interior and exterior penalty functions are employed, NIAs have mainly focused on the last ones.
- Their implementation is quite simple ... but,
  - Penalty functions require a careful fine-tuning of their penalty factors.
  - Such values usually are highly problem-dependent.
  - Different approaches have been proposed to tackle this shortcoming.
Penalty functions

Pros and cons

- The aim is to decrease the fitness of infeasible solutions.
- Unlike mathematical programming approaches, where interior and exterior penalty functions are employed, NIAs have mainly focused on the last ones.
- Their implementation is quite simple ... but,
- Penalty functions require a careful fine-tuning of their penalty factors.
- Such values usually are highly problem-dependent.
- Different approaches have been proposed to tackle this shortcoming.
Penalty functions

Pros and cons

- The aim is to decrease the fitness of infeasible solutions.
- Unlike mathematical programming approaches, where interior and exterior penalty functions are employed, NIAs have mainly focused on the last ones.
- Their implementation is quite simple ... but,
- Penalty functions require a careful fine-tuning of their penalty factors.
- Such values usually are highly problem-dependent.
- Different approaches have been proposed to tackle this shortcoming.
Penalty functions

Pros and cons

- The aim is to decrease the fitness of infeasible solutions.
- Unlike mathematical programming approaches, where interior and exterior penalty functions are employed, NIAs have mainly focused on the last ones.
- Their implementation is quite simple ... but,
- Penalty functions require a careful fine-tuning of their penalty factors.
- Such values usually are highly problem-dependent.
- Different approaches have been proposed to tackle this shortcoming.
Penalty functions

Death penalty

- The most simple penalty function.
- Infeasible solutions are assigned the worst possible fitness value or are simply eliminated from the optimization process.
- Keeps the search from using valuable information from infeasible solutions.
- Not suitable for very small feasible region with respect to the whole search space.
Penalty functions

Death penalty

- The most simple penalty function.
- Infeasible solutions are assigned the worst possible fitness value or are simply eliminated from the optimization process.
- Keeps the search from using valuable information from infeasible solutions.
- Not suitable for very small feasible region with respect to the whole search space.
Penalty functions

Death penalty

- The most simple penalty function.
- Infeasible solutions are assigned the worst possible fitness value or are simply eliminated from the optimization process.
- Keeps the search from using valuable information from infeasible solutions.

- Not suitable for very small feasible region with respect to the whole search space.
Penalty functions

Death penalty

- The most simple penalty function.
- Infeasible solutions are assigned the worst possible fitness value or are simply eliminated from the optimization process.
- Keeps the search from using valuable information from infeasible solutions.
- Not suitable for very small feasible region with respect to the whole search space.
Penalty functions

Static penalty functions

- Those whose penalty factor values \( r_i \) and \( c_j, i = 1, \ldots, m \) and \( j = 1, \ldots, m \) remain fixed during all the process.
  - Kuri and Villegas-Quezada [59].
  - Homaifar et al. [43]. Hoffmeister and Sprave [42].
  - Le Riche et al. [112]).

- The main drawback is the generalization of such type of approach, i.e., the values that may be suitable for one problem are normally unsuitable for another one.
Penalty functions

Static penalty functions

- Those whose penalty factor values \((r_i \text{ and } c_j, i = 1, \ldots, m \text{ and } j = 1 \ldots, m)\) remain fixed during all the process.
  - Kuri and Villegas-Quezada [59].
  - Homaifar et al. [43]. Hoffmeister and Sprave [42].
  - Le Riche et al. [112]).

- The main drawback is the generalization of such type of approach, i.e., the values that may be suitable for one problem are normally unsuitable for another one.
Penalty functions

Dynamic penalty functions

- Time (usually the generation counter in a NIA) is used to affect the penalty factors.
- Considering the usage of exterior penalty functions, soft penalties are expected first, while severe penalties are adopted in the last part of the search.
- Examples:
  - Joines and Houck [48].
  - Kazarlis and Petridis [51].
  - Crossley and Williams [21].
- The cooling factor of the simulated annealing algorithm has been employed to vary the penalty factors by Michalewicz and Attia [94].
- The main disadvantages of dynamic penalty functions are the parameters for their dynamic tuning and the difficulty to generalize them.
Penalty functions

Dynamic penalty functions

- Time (usually the generation counter in a NIA) is used to affect the penalty factors.

- Considering the usage of exterior penalty functions, soft penalties are expected first, while severe penalties are adopted in the last part of the search.

- Examples:
  - Joines and Houck [48].
  - Kazarlis and Petridis[51].
  - Crossley and Williams [21].

- The cooling factor of the simulated annealing algorithm has been employed to vary the penalty factors by Michalewicz and Attia [94].

- The main disadvantages of dynamic penalty functions are the parameters for their dynamic tuning and the difficulty to generalize them.
Penalty functions

Dynamic penalty functions

- Time (usually the generation counter in a NIA) is used to affect the penalty factors.
- Considering the usage of exterior penalty functions, soft penalties are expected first, while severe penalties are adopted in the last part of the search.
- Examples:
  - Joines and Houck [48].
  - Kazarlis and Petridis [51].
  - Crossley and Williams [21].
- The cooling factor of the simulated annealing algorithm has been employed to vary the penalty factors by Michalewicz and Attia [94].
- The main disadvantages of dynamic penalty functions are the parameters for their dynamic tuning and the difficulty to generalize them.
Penalty functions

Dynamic penalty functions

- Time (usually the generation counter in a NIA) is used to affect the penalty factors.
- Considering the usage of exterior penalty functions, soft penalties are expected first, while severe penalties are adopted in the last part of the search.
- Examples:
  - Joines and Houck [48].
  - Kazarlis and Petridis [51].
  - Crossley and Williams [21].
- The cooling factor of the simulated annealing algorithm has been employed to vary the penalty factors by Michalewicz and Attia [94].
- The main disadvantages of dynamic penalty functions are the parameters for their dynamic tuning and the difficulty to generalize them.
Penalty functions

Dynamic penalty functions

- Time (usually the generation counter in a NIA) is used to affect the penalty factors.
- Considering the usage of exterior penalty functions, soft penalties are expected first, while severe penalties are adopted in the last part of the search.
- Examples:
  - Joines and Houck [48].
  - Kazarlis and Petridis [51].
  - Crossley and Williams [21].
- The cooling factor of the simulated annealing algorithm has been employed to vary the penalty factors by Michalewicz and Attia [94].
- The main disadvantages of dynamic penalty functions are the parameters for their dynamic tuning and the difficulty to generalize them.
Penalty functions

Adaptive penalty functions

- The behavior of the NIA is used to update the penalty factors.
- Feasibility of the best solution in a number of generations by Hadj-Alouane and Bean [34].
- The fitness of the best feasible solution by Rasheed [105].
- The balance between feasible and infeasible solutions by Hamda and Schoenauer [35] and Hamida and Schoenauer [36].
- The average of the objective function and the level of violation of each constraint by Barbosa and Lemonge [11].
- Co-evolution by Coello [20].
- Fuzzy logic by Wu and Yu [150].
- Their main drawback lies in the following: there is no guarantee that the values defined based on the current behavior will be indeed useful later.
Penalty functions

Adaptive penalty functions

- The behavior of the NIA is used to update the penalty factors.
- Feasibility of the best solution in a number of generations by Hadj-Alouane and Bean [34].
- The fitness of the best feasible solution by Rasheed [105].
- The balance between feasible and infeasible solutions by Hamda and Schoenauer [35] and Hamida and Schoenauer [36].
- The average of the objective function and the level of violation of each constraint by Barbosa and Lemonge [11].
- Co-evolution by Coello [20].
- Fuzzy logic by Wu and Yu [150].
- Their main drawback lies in the following: there is no guarantee that the values defined based on the current behavior will be indeed useful later.
Penalty functions

Adaptive penalty functions

- The behavior of the NIA is used to update the penalty factors.
- Feasibility of the best solution in a number of generations by Hadj-Alouane and Bean [34].
- The fitness of the best feasible solution by Rasheed [105].
- The balance between feasible and infeasible solutions by Hamda and Schoenauer [35] and Hamida and Schoenauer [36].
- The average of the objective function and the level of violation of each constraint by Barbosa and Lemonge [11].
- Co-evolution by Coello [20].
- Fuzzy logic by Wu and Yu [150].
- Their main drawback lies in the following: there is no guarantee that the values defined based on the current behavior will be indeed useful later.
Penalty functions

Adaptive penalty functions

- The behavior of the NIA is used to update the penalty factors.
- Feasibility of the best solution in a number of generations by Hadj-Alouane and Bean [34].
- The fitness of the best feasible solution by Rasheed [105].
- The balance between feasible and infeasible solutions by Hamda and Schoenauer [35] and Hamida and Schoenauer [36].
- The average of the objective function and the level of violation of each constraint by Barbosa and Lemonge [11].
- Co-evolution by Coello [20].
- Fuzzy logic by Wu and Yu [150].
- Their main drawback lies in the following: there is no guarantee that the values defined based on the current behavior will be indeed useful later.
Penalty functions

Adaptive penalty functions

- The behavior of the NIA is used to update the penalty factors.
- Feasibility of the best solution in a number of generations by Hadj-Alouane and Bean [34].
- The fitness of the best feasible solution by Rasheed [105].
- The balance between feasible and infeasible solutions by Hamda and Schoenauer [35] and Hamida and Schoenauer [36].
- The average of the objective function and the level of violation of each constraint by Barbosa and Lemonge [11].
- Co-evolution by Coello [20].
- Fuzzy logic by Wu and Yu [150].
- Their main drawback lies in the following: there is no guarantee that the values defined based on the current behavior will be indeed useful later.
Adaptive penalty functions

- The behavior of the NIA is used to update the penalty factors.
- Feasibility of the best solution in a number of generations by Hadj-Alouane and Bean [34].
- The fitness of the best feasible solution by Rasheed [105].
- The balance between feasible and infeasible solutions by Hamda and Schoenauer [35] and Hamida and Schoenauer [36].
- The average of the objective function and the level of violation of each constraint by Barbosa and Lemonge [11].
- Co-evolution by Coello [20].
- Fuzzy logic by Wu and Yu [150].
- Their main drawback lies in the following: there is no guarantee that the values defined based on the current behavior will be indeed useful later.
Penalty functions

Adaptive penalty functions

- The behavior of the NIA is used to update the penalty factors.
- Feasibility of the best solution in a number of generations by Hadj-Alouane and Bean [34].
- The fitness of the best feasible solution by Rasheed [105].
- The balance between feasible and infeasible solutions by Hamda and Schoenauer [35] and Hamida and Schoenauer [36].
- The average of the objective function and the level of violation of each constraint by Barbosa and Lemonge [11].
- Co-evolution by Coello [20].
- Fuzzy logic by Wu and Yu [150].

Their main drawback lies in the following: there is no guarantee that the values defined based on the current behavior will be indeed useful later.
Penalty functions

Adaptive penalty functions

- The behavior of the NIA is used to update the penalty factors.
- Feasibility of the best solution in a number of generations by Hadj-Alouane and Bean [34].
- The fitness of the best feasible solution by Rasheed [105].
- The balance between feasible and infeasible solutions by Hamda and Schoenauer [35] and Hamida and Schoenauer [36].
- The average of the objective function and the level of violation of each constraint by Barbosa and Lemonge [11].
- Co-evolution by Coello [20].
- Fuzzy logic by Wu and Yu [150].
- Their main drawback lies in the following: there is no guarantee that the values defined based on the current behavior will be indeed useful later.
Penalty functions

Discussion

- Diverse ways to define penalty factors (static, dynamic, adaptive, co-evolved, fuzzy-adapted, etc.).
- Not clear which approach was more competitive.
- Most of the time, additional parameters were required.
**Penalty functions**

**Discussion**

- Diverse ways to define penalty factors (static, dynamic, adaptive, co-evolved, fuzzy-adapted, etc.).
- Not clear which approach was more competitive.
- Most of the time, additional parameters were required.
Penalty functions

Discussion

- Diverse ways to define penalty factors (static, dynamic, adaptive, co-evolved, fuzzy-adapted, etc.).
- Not clear which approach was more competitive.
- Most of the time, additional parameters were required.
Outline

1. Introduction
   - The problem of interest
   - Some important concepts
   - Mathematical-programming methods
   - Why alternative methods?

2. The early years
   - Penalty functions
   - Decoders
     - Special operators
     - Separation of objective function and constraints
   - General comments

3. Current constraint-handling techniques
   - Feasibility rules
   - Stochastic ranking
   - \( \varepsilon \)-constrained method
   - Novel penalty functions
   - Novel special operators
   - Multi-objective concepts
   - Ensemble of constraint-handling techniques

4. Summary and current trends
   - A bird’s eye view
   - Current trends
Decoders

- One of the most competitive constraint-handling techniques in the early years.
- They are based on the idea of mapping the feasible region $\mathcal{F}$ of the search space $S$ onto an easier-to-sample space where a NIA can provide a better performance [56].
Decoders

- One of the most competitive constraint-handling techniques in the early years.
- They are based on the idea of mapping the feasible region $\mathcal{F}$ of the search space $S$ onto an easier-to-sample space where a NIA can provide a better performance [56].
Decoders

- The mapping process must guarantee that each feasible solution in the search space is included in the decoded space and that a decoded solution corresponds to a feasible solution in the search space.
- The transformation process must be fast and it is highly desirable that small changes in the search space of the original problem cause small changes in the decoded space as well.
  - Homomorphous maps: the feasible region is mapped into an \( n \)-dimensional cube, by Koziol and Michalewicz [56, 57].
  - Riemann mappings by Kim and Husbands [52, 53, 54].
Decoders

- The mapping process must guarantee that each feasible solution in the search space is included in the decoded space and that a decoded solution corresponds to a feasible solution in the search space.
- The transformation process must be fast and it is highly desirable that small changes in the search space of the original problem cause small changes in the decoded space as well.
  - Homomorphous maps: the feasible region is mapped into an $n$-dimensional cube, by Koziel and Michalewicz [56, 57].
  - Riemann mappings by Kim and Husbands [52, 53, 54].
Decoders

Discussion

- Their actual implementation is far from trivial.
- They may involve a computational cost.
- Decoders are rarely used nowadays.
Decoders

Discussion

- Their actual implementation is far from trivial.
- They may involve a computational cost.
- Decoders are rarely used nowadays.
Decoders

Discussion

- Their actual implementation is far from trivial.
- They may involve a computational cost.
- Decoders are rarely used nowadays.
Outline

1. Introduction
   - The problem of interest
   - Some important concepts
   - Mathematical-programming methods
   - Why alternative methods?

2. The early years
   - Penalty functions
   - Decoders
   - Special operators
   - Separation of objective function and constraints
   - General comments

3. Current constraint-handling techniques
   - Feasibility rules
   - Stochastic ranking
   - \(\varepsilon\)-constrained method
   - Novel penalty functions
   - Novel special operators
   - Multi-objective concepts
   - Ensemble of constraint-handling techniques

4. Summary and current trends
   - A bird’s eye view
   - Current trends
Special operators

A special operator is conceived as a way of either preserving the feasibility of a solution or moving within a specific region of interest within the search space.

A variation operator which constructs linear combinations of feasible solutions to preserve their feasibility (GENOCOP) by Michalewicz [93].

Special operators designed to convert solutions which only satisfy linear constraints into fully feasible solutions (GENOCOP III) by Michalewicz and Nazhiyath [95].

Special operators to assign values to the decision variables aiming to keep the feasibility of the solution by Kowalczyk [55].

Special operators for two specific problems to sample the boundaries of their feasible regions by Schoenauer and Michalewicz [116, 117].
A special operator is conceived as a way of either preserving the feasibility of a solution or moving within a specific region of interest within the search space.

A variation operator which constructs linear combinations of feasible solutions to preserve their feasibility (GENOCOP) by Michalewicz [93].

Special operators designed to convert solutions which only satisfy linear constraints into fully feasible solutions (GENOCOP III) by Michalewicz and Nazhiyath [95].

Special operators to assign values to the decision variables aiming to keep the feasibility of the solution by Kowalczyk [55].

Special operators for two specific problems to sample the boundaries of their feasible regions by Schoenauer and Michalewicz [116, 117].
Special operators

- A special operator is conceived as a way of either preserving the feasibility of a solution or moving within a specific region of interest within the search space.

- A variation operator which constructs linear combinations of feasible solutions to preserve their feasibility (GENOCOP) by Michalewicz [93].

- Special operators designed to convert solutions which only satisfy linear constraints into fully feasible solutions (GENOCOP III) by Michalewicz and Nazhiyath [95].

- Special operators to assign values to the decision variables aiming to keep the feasibility of the solution by Kowalczyk [55].

- Special operators for two specific problems to sample the boundaries of their feasible regions by Schoenauer and Michalewicz [116, 117].
A special operator is conceived as a way of either preserving the feasibility of a solution or moving within a specific region of interest within the search space.

A variation operator which constructs linear combinations of feasible solutions to preserve their feasibility (GENOCOP) by Michalewicz [93].

Special operators designed to convert solutions which only satisfy linear constraints into fully feasible solutions (GENOCOP III) by Michalewicz and Nazhiyath [95].

Special operators to assign values to the decision variables aiming to keep the feasibility of the solution by Kowalczyk [55].

Special operators for two specific problems to sample the boundaries of their feasible regions by Schoenauer and Michalewicz [116, 117].
A special operator is conceived as a way of either preserving the feasibility of a solution or moving within a specific region of interest within the search space.

A variation operator which constructs linear combinations of feasible solutions to preserve their feasibility (GENOCOP) by Michalewicz [93].

Special operators designed to convert solutions which only satisfy linear constraints into fully feasible solutions (GENOCOP III) by Michalewicz and Nazhiyath [95].

Special operators to assign values to the decision variables aiming to keep the feasibility of the solution by Kowalczyk [55].

Special operators for two specific problems to sample the boundaries of their feasible regions by Schoenauer and Michalewicz [116, 117].
Special operators

Discussion

- Highly competitive results can be found when adopting special operators.
- Their main drawback is their limited applicability.
- Most of them require an ad-hoc initialization process or at least one feasible or partially-feasible solution in the initial population.
**Special operators**

**Discussion**

- Highly competitive results can be found when adopting special operators.
- Their main drawback is their limited applicability.
- Most of them require an ad-hoc initialization process or at least one feasible or partially-feasible solution in the initial population.
Special operators

Discussion

- Highly competitive results can be found when adopting special operators.
- Their main drawback is their limited applicability.
- Most of them require an ad-hoc initialization process or at least one feasible or partially-feasible solution in the initial population.
Outline

1. Introduction
   - The problem of interest
   - Some important concepts
   - Mathematical-programming methods
   - Why alternative methods?

2. The early years
   - Penalty functions
   - Decoders
   - Special operators
   - Separation of objective function and constraints
   - General comments

3. Current constraint-handling techniques
   - Feasibility rules
   - Stochastic ranking
   - $\varepsilon$-constrained method
   - Novel penalty functions
   - Novel special operators
   - Multi-objective concepts
   - Ensemble of constraint-handling techniques

4. Summary and current trends
   - A bird’s eye view
   - Current trends
Unlike combining the objective function and the values of the constraints into a single value (i.e. penalty function), there are constraint-handling techniques which work with the opposite idea.
Powell and Skolnick in [103] proposed an approach based on the following Equation.

\[
\text{fitness}(\vec{x}) = \begin{cases} 
  f(\vec{x}) & \text{if feasible} \\
  1 + r \left( \sum_{i=1}^{m} g_i(\vec{x}) + \sum_{j=1}^{p} h_j(\vec{x}) \right) & \text{otherwise}
\end{cases}
\]

where a feasible solution has always a better fitness value with respect to that of an infeasible solution, whose fitness is based only on their accumulated constraint violation.
Hinterding and Michalewicz in [41] proposed the idea of dividing the search in two phases: (1) finding feasible solutions, regardless of the objective function value, and (2) after a suitable number of feasible solutions has been found, optimizing the objective function.

Such idea was revisited by Venkatraman and Yen [138].

Schoenauer and Xanthakis in [118] proposed a lexicographic ordering (behavioral memory) to satisfy constraints, i.e., when a certain number of solutions in the population satisfy the first constraint, an attempt to satisfy the second one is made (but the first constraint must continue to be satisfied), and so on.
Hinterding and Michalewicz in [41] proposed the idea of dividing the search in two phases: (1) finding feasible solutions, regardless of the objective function value, and (2) after a suitable number of feasible solutions has been found, optimizing the objective function.

Such idea was revisited by Venkatraman and Yen [138].

Schoenauer and Xanthakis in [118] proposed a lexicographic ordering (behavioral memory) to satisfy constraints, i.e., when a certain number of solutions in the population satisfy the first constraint, an attempt to satisfy the second one is made (but the first constraint must continue to be satisfied), and so on.
Hinterding and Michalewicz in [41] proposed the idea of dividing the search in two phases: (1) finding feasible solutions, regardless of the objective function value, and (2) after a suitable number of feasible solutions has been found, optimizing the objective function.

Such idea was revisited by Venkatraman and Yen [138].

Schoenauer and Xanthakis in [118] proposed a lexicographic ordering (behavioral memory) to satisfy constraints, i.e., when a certain number of solutions in the population satisfy the first constraint, an attempt to satisfy the second one is made (but the first constraint must continue to be satisfied), and so on.
Deb [25] proposed a set of three feasibility criteria as follows:

1. When comparing two feasible solutions, the one with the best objective function is chosen.
2. When comparing a feasible and an infeasible solution, the feasible one is chosen.
3. When comparing two infeasible solutions, the one with the lowest sum of constraint violation is chosen.

The sum of constraint violation can be calculated as follows:

$$
\phi(\vec{x}) = \sum_{i=1}^{m} \max(0, g_i(\vec{x}))^2 + \sum_{j=1}^{p} |h_j(\vec{x})|
$$
Different multi-population schemes have been proposed. Coello [19] divided a GA-population into sub-populations and each sub-population tried to satisfy one constraint of a CNOP and another one optimized the objective function.

Liang and Suganathan proposed a dynamic assignment of sub-swarms to constraints in PSO [67].

The approach was further improved in [68], where only two sub-swarms, one of them with a tolerance for inequality constraints, were used. Each particle, and not a sub-swarm, was dynamically assigned the objective function or the constraint, in such a way that more difficult objectives to optimize (satisfy) were assigned more frequently.

Li et al. [66] adopted a similar approach but using DE as a search algorithm.
Separation of objective function and constraints

- Different multi-population schemes have been proposed.
- Coello [19] divided a GA-population into sub-populations and each sub-population tried to satisfy one constraint of a CNOP and another one optimized the objective function.
- Liang and Suganthan proposed a dynamic assignment of sub-swarms to constraints in PSO [67].
- The approach was further improved in [68], where only two sub-swarms, one of them with a tolerance for inequality constraints, were used. Each particle, and not a sub-swarm, was dynamically assigned the objective function or the constraint, in such a way that more difficult objectives to optimize (satisfy) were assigned more frequently.
- Li et al. [66] adopted a similar approach but using DE as a search algorithm.
Different multi-population schemes have been proposed.

Coello [19] divided a GA-population into sub-populations and each sub-population tried to satisfy one constraint of a CNOP and another one optimized the objective function.

Liang and Suganthan proposed a dynamic assignment of sub-swarms to constraints in PSO [67].

The approach was further improved in [68], where only two sub-swarms, one of them with a tolerance for inequality constraints, were used. Each particle, and not a sub-swarm, was dynamically assigned the objective function or the constraint, in such a way that more difficult objectives to optimize (satisfy) were assigned more frequently.

Li et al. [66] adopted a similar approach but using DE as a search algorithm.
Separation of objective function and constraints

- Different multi-population schemes have been proposed.
- Coello [19] divided a GA-population into sub-populations and each sub-population tried to satisfy one constraint of a CNOP and another one optimized the objective function.
- Liang and Suganthan proposed a dynamic assignment of sub-swarms to constraints in PSO [67].
- The approach was further improved in [68], where only two sub-swarms, one of them with a tolerance for inequality constraints, were used. Each particle, and not a sub-swarm, was dynamically assigned the objective function or the constraint, in such a way that more difficult objectives to optimize (satisfy) were assigned more frequently.
- Li et al. [66] adopted a similar approach but using DE as a search algorithm.
Different multi-population schemes have been proposed.

Coello [19] divided a GA-population into sub-populations and each sub-population tried to satisfy one constraint of a CNOP and another one optimized the objective function.

Liang and Suganthan proposed a dynamic assignment of sub-swarms to constraints in PSO [67].

The approach was further improved in [68], where only two sub-swarms, one of them with a tolerance for inequality constraints, were used. Each particle, and not a sub-swarm, was dynamically assigned the objective function or the constraint, in such a way that more difficult objectives to optimize (satisfy) were assigned more frequently.

Li et al. [66] adopted a similar approach but using DE as a search algorithm.
Liu et al. [69] proposed a separation scheme based on a co-evolutionary approach in which two populations are adopted. The first one optimized the objective function without considering the constraints, while the second population aimed to satisfy the constraints of the problem. Each population could migrate solutions to the other.
Multi-objective optimization concepts (Pareto dominance and Pareto ranking) have been quite popular to solve constrained optimization problems [84]. Two groups can be identified:

1. CNOP as a bi-objective problem (the original objective function and the sum of constraint violation).
2. CNOP as a multi-objective optimization problem (the original objective function and each constraint are handled as objectives).
The main shortcomings are related to the lack of bias provided by Pareto ranking when used in a straightforward manner [115], and the difficulties of these approaches to preserve diversity in the population [84].

Additional mechanisms have been adopted such as Pareto ranking in different search spaces [106, 107, 1, 4], the shrinking of the search space [40] and the use of non-dominated sorting and clustering techniques to generate collaboration among sub-populations [108].
The main shortcomings are related to the lack of bias provided by Pareto ranking when used in a straightforward manner [115], and the difficulties of these approaches to preserve diversity in the population [84].

Additional mechanisms have been adopted such as Pareto ranking in different search spaces [106, 107, 1, 4], the shrinking of the search space [40] and the use of non-dominated sorting and clustering techniques to generate collaboration among sub-populations [108].
Discussion

- This type of constraint-handling technique has been found to generate an important diversity loss.
- It is important to design appropriate diversity maintenance mechanisms.
- However, they are quite popular (usually no additional parameters required and easy to generalize).
Discussion

This type of constraint-handling technique has been found to generate an important diversity loss.

It is important to design appropriate diversity maintenance mechanisms.

However, they are quite popular (usually no additional parameters required and easy to generalize).
Separation of objective function and constraints

Discussion

- This type of constraint-handling technique has been found to generate an important diversity loss.
- It is important to design appropriate diversity maintenance mechanisms.
- However, they are quite popular (usually no additional parameters required and easy to generalize).
Outline

1. Introduction
   - The problem of interest
   - Some important concepts
   - Mathematical-programming methods
   - Why alternative methods?

2. The early years
   - Penalty functions
   - Decoders
   - Special operators
   - Separation of objective function and constraints
   - General comments

3. Current constraint-handling techniques
   - Feasibility rules
   - Stochastic ranking
   - \(\varepsilon\)-constrained method
   - Novel penalty functions
   - Novel special operators
   - Multi-objective concepts
   - Ensemble of constraint-handling techniques

4. Summary and current trends
   - A bird’s eye view
   - Current trends
General comments

The first attempts to generate constraint-handling techniques were similar to an exploration phase, in which a variety of approaches were proposed.

Main shortcomings:
- Unsuitable bias.
- Need of a careful fine-tuning of parameters.
- Difficult to generalize.
- High computational cost and difficult implementations.

The exploitation phase was about to begin.
The first attempts to generate constraint-handling techniques were similar to an exploration phase, in which a variety of approaches were proposed. 

Main shortcomings:
- Unsuitable bias.
- Need of a careful fine-tuning of parameters.
- Difficult to generalize.
- High computational cost and difficult implementations.

The exploitation phase was about to begin.
General comments

The first attempts to generate constraint-handling techniques were similar to an exploration phase, in which a variety of approaches were proposed.

Main shortcomings:
- Unsuitable bias.
- Need of a careful fine-tuning of parameters.
- Difficult to generalize.
- High computational cost and difficult implementations.

The exploitation phase was about to begin.
Outline

1. Introduction
   - The problem of interest
   - Some important concepts
   - Mathematical-programming methods
   - Why alternative methods?

2. The early years
   - Penalty functions
   - Decoders
   - Special operators
   - Separation of objective function and constraints
   - General comments

3. Current constraint-handling techniques
   - Feasibility rules
   - Stochastic ranking
   - \( \varepsilon \)-constrained method
   - Novel penalty functions
   - Novel special operators
   - Multi-objective concepts
   - Ensemble of constraint-handling techniques

4. Summary and current trends
   - A bird’s eye view
   - Current trends
The feasibility rules proposed by Deb [25, 100] constitute an example of a constraint-handling technique that was proposed several years ago, but whose impact is still present in the literature.

Its popularity lies on its ability to be coupled to a variety of algorithms, without introducing new parameters.
The feasibility rules proposed by Deb [25, 100] constitute an example of a constraint-handling technique that was proposed several years ago, but whose impact is still present in the literature.

Its popularity lies on its ability to be coupled to a variety of algorithms, without introducing new parameters.
Parameter control mechanisms in DE-based constrained numerical optimization by Palomeque and Mezura-Montes (DE self-adaptive parameters, including diversity parameters) [89] and by Zielinski et al. (DE adaptive parameters) [162].

Zielinski and Laur [160] explored different termination conditions (e.g., improvement-based criteria, movement-based criteria, distribution-based criteria) for DE in constrained optimization.

Zielinski and Laur [161] studied the effect of the tolerance utilized in the equality constraints, where values between $\epsilon = 1 \times 10^{-7}$ and $\epsilon = 1 \times 10^{-15}$ allowed the algorithm, coupled with the feasibility rules, to reach competitive results.

Mezura-Montes and Coello Coello [83] explored diversity mechanisms to improve the performance of evolution strategies when solving CNOPs.
Feasibility rules

Studies on other topics

- Parameter control mechanisms in DE-based constrained numerical optimization by Palomeque and Mezura-Montes (DE self-adaptive parameters, including diversity parameters) [89] and by Zielinski et al. (DE adaptive parameters) [162].

- Zielinski and Laur [160] explored different termination conditions (e.g., improvement-based criteria, movement-based criteria, distribution-based criteria) for DE in constrained optimization.

- Zielinski and Laur [161] studied the effect of the tolerance utilized in the equality constraints, where values between $\epsilon = 1 \times 10^{-7}$ and $\epsilon = 1 \times 10^{-15}$ allowed the algorithm, coupled with the feasibility rules, to reach competitive results.

- Mezura-Montes and Coello Coello [83] explored diversity mechanisms to improve the performance of evolution strategies when solving CNOPs.
Feasibility rules

Studies on other topics

- Parameter control mechanisms in DE-based constrained numerical optimization by Palomeque and Mezura-Montes (DE self-adaptive parameters, including diversity parameters) [89] and by Zielinski et al. (DE adaptive parameters) [162].

- Zielinski and Laur [160] explored different termination conditions (e.g., improvement-based criteria, movement-based criteria, distribution-based criteria) for DE in constrained optimization.

- Zielinski and Laur [161] studied the effect of the tolerance utilized in the equality constraints, where values between $\epsilon = 1 \times 10^{-7}$ and $\epsilon = 1 \times 10^{-15}$ allowed the algorithm, coupled with the feasibility rules, to reach competitive results.

- Mezura-Montes and Coello Coello [83] explored diversity mechanisms to improve the performance of evolution strategies when solving CNOPs.
Studies on other topics

- Parameter control mechanisms in DE-based constrained numerical optimization by Palomeque and Mezura-Montes (DE self-adaptive parameters, including diversity parameters) [89] and by Zielinski et al. (DE adaptive parameters) [162].
- Zielinski and Laur [160] explored different termination conditions (e.g., improvement-based criteria, movement-based criteria, distribution-based criteria) for DE in constrained optimization.
- Zielinski and Laur [161] studied the effect of the tolerance utilized in the equality constraints, where values between $\epsilon = 1 \times 10^{-7}$ and $\epsilon = 1 \times 10^{-15}$ allowed the algorithm, coupled with the feasibility rules, to reach competitive results.
- Mezura-Montes and Coello Coello [83] explored diversity mechanisms to improve the performance of evolution strategies when solving CNOPs.
Feasibility rules

Multi-operator mechanisms

- The use of feasibility rules has favored the development of approaches with self-adaptive variation operator selection mechanisms on DE:
  - jDE-2 by Brest [14], where different variants are combined with an injection of solutions generated at random
  - SaDE by Huang et al. [46], where, besides the combination of DE variants, SQP is adopted as a local search operator.
  - Four DE variants with four sub-populations with fixed-dynamic size and migration by Elsayed et al. [80].
Feasibility rules

Multi-operator mechanisms

- The use of feasibility rules has favored the development of approaches with self-adaptive variation operator selection mechanisms on DE:
- jDE-2 by Brest [14], where different variants are combined with an injection of solutions generated at random
- SaDE by Huang et al. [46], where, besides the combination of DE variants, SQP is adopted as a local search operator.
- Four DE variants with four sub-populations with fixed-dynamic size and migration by Elsayed et al. [80].
Feasibility rules

Multi-operator mechanisms

- The use of feasibility rules has favored the development of approaches with self-adaptive variation operator selection mechanisms on DE:
  - jDE-2 by Brest [14], where different variants are combined with an injection of solutions generated at random
  - SaDE by Huang et al. [46], where, besides the combination of DE variants, SQP is adopted as a local search operator.
  - Four DE variants with four sub-populations with fixed-dynamic size and migration by Elsayed et al. [80].
Feasibility rules

Multi-operator mechanisms

- The use of feasibility rules has favored the development of approaches with self-adaptive variation operator selection mechanisms on DE:
  - jDE-2 by Brest [14], where different variants are combined with an injection of solutions generated at random.
  - SaDE by Huang et al. [46], where, besides the combination of DE variants, SQP is adopted as a local search operator.
  - Four DE variants with four sub-populations with fixed-dynamic size and migration by Elsayed et al. [80].
Feasibility rules

Multi-operator mechanisms

... and studies on PSO:

A combination of global-local best PSO with dynamic mutation operator by Cagnina et al.\cite{17}. Due to stagnation in some test problems, a further version of this approach was proposed by the same authors in \cite{16}, where a bi-population scheme and a “shake” operator were added \cite{16}.
Feasibility rules

Multi-operator mechanisms

... and studies on PSO:

A combination of global-local best PSO with dynamic mutation operator by Cagnina et al.[17]. Due to stagnation in some test problems, a further version of this approach was proposed by the same authors in [16], where a bi-population scheme and a “shake” operator were added [16].
Feasibility rules

Multi-operator mechanisms

- and also on GAs.

- Elsayed et al. implemented their four sub-population scheme with GAs in [80].

- Elsayed et al. [28] proposed a modified GA where a novel crossover operator called multi-parent crossover and also a randomized operator were added to a real-coded GA.

- Elsayed et al. [27] compared ten different GA variants. The crossover operators employed were triangular crossover, Simulated binary crossover, parent-centric crossover, simplex crossover, and blend crossover. The mutation operators adopted were non-uniform mutation and polynomial crossover.
Feasibility rules

Multi-operator mechanisms

- and also on GAs.
- Elsayed et al. implemented their four sub-population scheme with GAs in [80].
- Elsayed et al. [28] proposed a modified GA where a novel crossover operator called multi-parent crossover and also a randomized operator were added to a real-coded GA.
- Elsayed et al. [27] compared ten different GA variants. The crossover operators employed were triangular crossover, Simulated binary crossover, parent-centric crossover, simplex crossover, and blend crossover. The mutation operators adopted were non-uniform mutation and polynomial crossover.
Feasibility rules

Multi-operator mechanisms

- and also on GAs.
- Elsayed et al. implemented their four sub-population scheme with GAs in [80].
- Elsayed et al. [28] proposed a modified GA where a novel crossover operator called multi-parent crossover and also a randomized operator were added to a real-coded GA.
- Elsayed et al. [27] compared ten different GA variants. The crossover operators employed were triangular crossover, Simulated binary crossover, parent-centric crossover, simplex crossover, and blend crossover. The mutation operators adopted were non-uniform mutation and polynomial crossover.

Efrén Mezura-Montes
CEC 2017, SPAIN
Feasibility rules

**Multi-operator mechanisms**

- and also on GAs.
- Elsayed et al. implemented their four sub-population scheme with GAs in [80].
- Elsayed et al. [28] proposed a modified GA where a novel crossover operator called multi-parent crossover and also a randomized operator were added to a real-coded GA.
- Elsayed et al. [27] compared ten different GA variants. The crossover operators employed were triangular crossover, Simulated binary crossover, parent-centric crossover, simplex crossover, and blend crossover. The mutation operators adopted were non-uniform mutation and polynomial crossover.
Feasibility rules

Combination with special operators

- Barkat Ullah [135] designed a mechanism to force infeasible individuals to move to the feasible region through the application of search space reduction and diversity checking mechanisms designed to avoid premature convergence.

- Mezura-Montes and Cetina-Domíngez [82] proposed a special operator designed to locate infeasible solutions close to the best feasible solution.

- A more recent version was proposed by the authors in [92], where two operators were improved and a direct-search local operator was added to the algorithm.
Combination with special operators

- Barkat Ullah [135] designed a mechanism to force infeasible individuals to move to the feasible region through the application of search space reduction and diversity checking mechanisms designed to avoid premature convergence.

- Mezura-Montes and Cetina-Domínguez [82] proposed a special operator designed to locate infeasible solutions close to the best feasible solution.

- A more recent version was proposed by the authors in [92], where two operators were improved and a direct-search local operator was added to the algorithm.
Feasibility rules

Combination with special operators

- Barkat Ullah [135] designed a mechanism to force infeasible individuals to move to the feasible region through the application of search space reduction and diversity checking mechanisms designed to avoid premature convergence.

- Mezura-Montes and Cetina-Domínguez [82] proposed a special operator designed to locate infeasible solutions close to the best feasible solution.

- A more recent version was proposed by the authors in [92], where two operators were improved and a direct-search local operator was added to the algorithm.
Adapted to DE

- Zielinski and Laur [159] coupled DE with the feasibility rules in a greedy selection scheme between target and trial vectors.
- Lampinen used a similar DE-based approach in [60]. However, the third criterion (originally based on the sum of constraint violation) was based on Pareto dominance in constraints space. Kukkonen and Lampinen proposed their Generalized Differential Evolution (GDE) [58] based on the aforementioned idea.
- Mezura-Montes et al. [85, 90, 91] proposed a new DE variant coupled with the ability to generate more than one offspring per parent.
Zielinski and Laur [159] coupled DE with the feasibility rules in a greedy selection scheme between target and trial vectors.

Lampinen used a similar DE-based approach in [60]. However, the third criterion (originally based on the sum of constraint violation) was based on Pareto dominance in constraints space. Kukkonen and Lampinen proposed their Generalized Differential Evolution (GDE) [58] based on the aforementioned idea.

Mezura-Montes et al. [85, 90, 91] proposed a new DE variant coupled with the ability to generate more than one offspring per parent.
Zielinski and Laur [159] coupled DE with the feasibility rules in a greedy selection scheme between target and trial vectors.

Lampinen used a similar DE-based approach in [60]. However, the third criterion (originally based on the sum of constraint violation) was based on Pareto dominance in constraints space. Kukkonen and Lampinen proposed their Generalized Differential Evolution (GDE) [58] based on the aforementioned idea.

Mezura-Montes et al. [85, 90, 91] proposed a new DE variant coupled with the ability to generate more than one offspring per parent.
Feasibility rules

Adapted to artificial immune systems

- Cruz et al. [22] and Aragón et al. [6], based on the clonal selection principle used the feasibility rules to rank antibodies based on affinity.

- Aragón et al. [7], based on a T-cell model in which three types of cells (solutions) are adopted, used the feasibility rules as the criteria in the replacement process.
Adapted to artificial immune systems

- Cruz et al. [22] and Aragón et al. [6], based on the clonal selection principle used the feasibility rules to rank antibodies based on affinity.

- Aragón et al. [7], based on a T-cell model in which three types of cells (solutions) are adopted, used the feasibility rules as the criteria in the replacement process.
Feasibility rules

Adapted to artificial bee colony

- Karaboga and Basturk [50] and Karaboga and Akay [49] changed a greedy selection based only on the objective function values by the use of the feasibility rules with the aim of adapting an artificial bee colony algorithm (ABC) to solve CNOPs.

- Mezura-Montes and Cetina-Domínguez [82] and Mezura-Montes and Velez-Koeppel [92] combined ABC with a smart-flight and a local-search operator, respectively, to improve its performance in constrained search spaces.
Adapted to artificial bee colony

Karaboga and Basturk [50] and Karaboga and Akay [49] changed a greedy selection based only on the objective function values by the use of the feasibility rules with the aim of adapting an artificial bee colony algorithm (ABC) to solve CNOPs.

Mezura-Montes and Cetina-Domínguez [82] and Mezura-Montes and Velez-Koeppel [92] combined ABC with a smart-flight and a local-search operator, respectively, to improve its performance in constrained search spaces.
Adapted to other NIAs

- Ma and Simon [76] proposed an improved version of the biogeography-based optimization (BBO) algorithm (inspired on the study of distributions of species over time and space) with the feasibility rules as criteria to choose solutions with the so-called “habitat suitability index”.

- Liu et al. [72] proposed the organizational evolutionary algorithm (OEA). A static penalty function and the feasibility rules were compared as constraint-handling techniques.

- Mezura-Montes and Hernández-Ocaña [88] used the feasibility rules with the Bacterial Foraging Optimization Algorithm (BFOA) in the greedy selection mechanism within the chemotactic loop.

- Landa and Coello [61] adopted the rules in an approach where a cultural DE-based mechanism was developed.
Ma and Simon [76] proposed an improved version of the biogeography-based optimization (BBO) algorithm (inspired on the study of distributions of species over time and space) with the feasibility rules as criteria to choose solutions with the so-called “habitat suitability index”.

Liu et al. [72] proposed the organizational evolutionary algorithm (OEA). A static penalty function and the feasibility rules were compared as constraint-handling techniques.

Mezura-Montes and Hernández-Ocaña [88] used the feasibility rules with the Bacterial Foraging Optimization Algorithm (BFOA) in the greedy selection mechanism within the chemotactic loop.

Landa and Coello [61] adopted the rules in an approach where a cultural DE-based mechanism was developed.
Ma and Simon [76] proposed an improved version of the biogeography-based optimization (BBO) algorithm (inspired on the study of distributions of species over time and space) with the feasibility rules as criteria to choose solutions with the so-called “habitat suitability index”.

Liu et al. [72] proposed the organizational evolutionary algorithm (OEA). A static penalty function and the feasibility rules were compared as constraint-handling techniques.

Mezura-Montes and Hernández-Ocaña [88] used the feasibility rules with the Bacterial Foraging Optimization Algorithm (BFOA) in the greedy selection mechanism within the chemotactic loop.

Landa and Coello [61] adopted the rules in an approach where a cultural DE-based mechanism was developed.
Feasibility rules

Adapted to other NIAs

- Ma and Simon [76] proposed an improved version of the biogeography-based optimization (BBO) algorithm (inspired on the study of distributions of species over time and space) with the feasibility rules as criteria to choose solutions with the so-called “habitat suitability index”.

- Liu et al. [72] proposed the organizational evolutionary algorithm (OEA). A static penalty function and the feasibility rules were compared as constraint-handling techniques.

- Mezura-Montes and Hernández-Ocaña [88] used the feasibility rules with the Bacterial Foraging Optimization Algorithm (BFOA) in the greedy selection mechanism within the chemotactic loop.

- Landa and Coello [61] adopted the rules in an approach where a cultural DE-based mechanism was developed.
Feasibility rules

Used in hybrid approaches

- Muñoz-Zavala et al. [98] used a DE mutation operator to update the local-best particle in PSO.
- Wang et al. [142] implicitly used feasibility rules to rank the particles in a hybrid multi-swarm PSO (HMPSO) where the DE mutation operator was also adopted.
- HMPSO was improved by Lui et al. in [70], where two additional mutation operators were used. The number of evaluations required by the improved approach, decreased in almost 50%.
- He and Wang [38] used simulated annealing (SA) as a local search operator and applied it to the gbest particle at each generation in PSO.
Feasibility rules

Used in hybrid approaches

- Muñoz-Zavala et al. [98] used a DE mutation operator to update the local-best particle in PSO.
- Wang et al. [142] implicitly used feasibility rules to rank the particles in a hybrid multi-swarm PSO (HMPSO) where the DE mutation operator was also adopted.
- HMPSO was improved by Lui et al. in [70], where two additional mutation operators were used. The number of evaluations required by the improved approach, decreased in almost 50%.
- He and Wang [38] used simulated annealing (SA) as a local search operator and applied it to the gbest particle at each generation in PSO.
Feasibility rules

Used in hybrid approaches

- Muñoz-Zavala et al. [98] used a DE mutation operator to update the local-best particle in PSO.

- Wang et al. [142] implicitly used feasibility rules to rank the particles in a hybrid multi-swarm PSO (HMPSO) where the DE mutation operator was also adopted.

- HMPSO was improved by Lui et al. in [70], where two additional mutation operators were used. The number of evaluations required by the improved approach, decreased in almost 50%.

- He and Wang [38] used simulated annealing (SA) as a local search operator and applied it to the gbest particle at each generation in PSO.
Muñoz-Zavala et al. [98] used a DE mutation operator to update the local-best particle in PSO.

Wang et al. [142] implicitly used feasibility rules to rank the particles in a hybrid multi-swarm PSO (HMPSO) where the DE mutation operator was also adopted.

HMPSO was improved by Lui et al. in [70], where two additional mutation operators were used. The number of evaluations required by the improved approach, decreased in almost 50%.

He and Wang [38] used simulated annealing (SA) as a local search operator and applied it to the gbest particle at each generation in PSO.
Feasibility rules

### Used in memetic algorithms

- Menchaca-Mendez and Coello Coello [81] proposed a DE-based algorithm with a variation of the Nelder-Mead algorithm as a local search operator.
- Sun and Garibaldi [122] proposed an estimation of distribution algorithm (EDA) with SQP as a local search operator.
- Ullah et al. [136, 137] presented an agent-based memetic algorithm where a learning process (a mutation operator chosen by the solution) is used to improve solutions.
- Hamza et al. [37] proposed a DE-based algorithm with a constraint-consensus operator applied to infeasible vectors so as to become them feasible.
Feasibility rules

Used in memetic algorithms

- Menchaca-Mendez and Coello Coello [81] proposed a DE-based algorithm with a variation of the Nelder-Mead algorithm as a local search operator.
- Sun and Garibaldi [122] proposed an estimation of distribution algorithm (EDA) with SQP as a local search operator.
- Ullah et al. [136, 137] presented an agent-based memetic algorithm where a learning process (a mutation operator chosen by the solution) is used to improve solutions.
- Hamza et al. [37] proposed a DE-based algorithm with a constraint-consensus operator applied to infeasible vectors so as to become them feasible.
Feasibility rules

**Used in memetic algorithms**

- Menchaca-Mendez and Coello Coello [81] proposed a DE-based algorithm with a variation of the Nelder-Mead algorithm as a local search operator.

- Sun and Garibaldi [122] proposed an estimation of distribution algorithm (EDA) with SQP as a local search operator.

- Ullah et al. [136, 137] presented an agent-based memetic algorithm where a learning process (a mutation operator chosen by the solution) is used to improve solutions.


- Hamza et al. [37] proposed a DE-based algorithm with a constraint-consensus operator applied to infeasible vectors so as to become them feasible.
Feasibility rules

Used in memetic algorithms

- Menchaca-Mendez and Coello Coello [81] proposed a DE-based algorithm with a variation of the Nelder-Mead algorithm as a local search operator.
- Sun and Garibaldi [122] proposed an estimation of distribution algorithm (EDA) with SQP as a local search operator.
- Ullah et al. [136, 137] presented an agent-based memetic algorithm where a learning process (a mutation operator chosen by the solution) is used to improve solutions.
- Hamza et al. [37] proposed a DE-based algorithm with a constraint-consensus operator applied to infeasible vectors so as to become them feasible.
Feasibility rules

Used in memetic algorithms

- Menchaca-Mendez and Coello Coello [81] proposed a DE-based algorithm with a variation of the Nelder-Mead algorithm as a local search operator.
- Sun and Garibaldi [122] proposed an estimation of distribution algorithm (EDA) with SQP as a local search operator.
- Ullah et al. [136, 137] presented an agent-based memetic algorithm where a learning process (a mutation operator chosen by the solution) is used to improve solutions.
- Hamza et al. [37] proposed a DE-based algorithm with a constraint-consensus operator applied to infeasible vectors so as to become them feasible.
Feasibility rules

Empirical studies

- Mezura-Montes and Flores-Mendoza compared PSO variants [87].
- Mezura-Montes et al. compared DE variants [86].
Feasibility rules

Empirical studies

- Mezura-Montes and Flores-Mendoza compared PSO variants [87].
- Mezura-Montes et al. compared DE variants [86].
1. Introduction
   - The problem of interest
   - Some important concepts
   - Mathematical-programming methods
   - Why alternative methods?

2. The early years
   - Penalty functions
   - Decoders
   - Special operators
   - Separation of objective function and constraints
   - General comments

3. Current constraint-handling techniques
   - Feasibility rules
   - Stochastic ranking
   - \( \varepsilon \)-constrained method
   - Novel penalty functions
   - Novel special operators
   - Multi-objective concepts
   - Ensemble of constraint-handling techniques

4. Summary and current trends
   - A bird’s eye view
   - Current trends
Stochastic ranking (SR)

- Proposed by Runarsson and Yao [114] to deal with the shortcomings of a penalty function (over and under penalization).
- A user-defined parameter called $P_f$ controls the criterion used for comparison of infeasible solutions:
  - Based on their sum of constraint violation
  - Based only on their objective function value.
- SR uses a bubble-sort-like process to rank the solutions in the population.
Stochastic ranking (SR)

- Proposed by Runarsson and Yao [114] to deal with the shortcomings of a penalty function (over and under penalization).
- A user-defined parameter called $P_f$ controls the criterion used for comparison of infeasible solutions:
  - Based on their sum of constraint violation
  - Based only on their objective function value.
- SR uses a bubble-sort-like process to rank the solutions in the population.
Stochastic ranking (SR)

- Proposed by Runarsson and Yao [114] to deal with the shortcomings of a penalty function (over and under penalization).
- A user-defined parameter called $P_f$ controls the criterion used for comparison of infeasible solutions:
  - Based on their sum of constraint violation
  - Based only on their objective function value.
- SR uses a bubble-sort-like process to rank the solutions in the population.
Stochastic Ranking (SR)

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 If
    End For
    If (not swap performed)
        break
    End For
End For
End
Stochastic ranking (SR)

Adapted to DE

- Despite being a ranking process, SR has been adopted by NIAs which do not rank solutions, such as DE.
- Zhang et al. [156] used SR in a DE variant based on [90]. $P_f$ was defined by a dynamic parameter control mechanism (high value at the beginning, low value at the end).
- Liu et al. [73, 71] also used SR in DE and proposed the concept of directional information related to the choice of the most convenient search direction based on the DE mutation operator.
- Fan et al. [30] ranked vectors with SR before the DE operators are applied. The population is split into two sets: (1) the vectors with the highest ranks, and (2) the remaining vectors. The base vector and the vector which determines the search direction are chosen at random from the first set. The other vector is chosen at random from the second set.
Stochastic ranking (SR)

Adapted to DE

- Despite being a ranking process, SR has been adopted by NIAs which do not rank solutions, such as DE.
- Zhang et al. [156] used SR in a DE variant based on [90]. $P_f$ was defined by a dynamic parameter control mechanism (high value at the beginning, low value at the end).
- Liu et al. [73, 71] also used SR in DE and proposed the concept of directional information related to the choice of the most convenient search direction based on the DE mutation operator.
- Fan et al. [30] ranked vectors with SR before the DE operators are applied. The population is split into two sets: (1) the vectors with the highest ranks, and (2) the remaining vectors. The base vector and the vector which determines the search direction are chosen at random from the first set. The other vector is chosen at random from the second set.
Stochastic ranking (SR)

Adapted to DE

Despite being a ranking process, SR has been adopted by NIAs which do not rank solutions, such as DE.

Zhang et al. [156] used SR in a DE variant based on [90]. \( P_f \) was defined by a dynamic parameter control mechanism (high value at the beginning, low value at the end).

Liu et al. [73, 71] also used SR in DE and proposed the concept of directional information related to the choice of the most convenient search direction based on the DE mutation operator.

Fan et al. [30] ranked vectors with SR before the DE operators are applied. The population is split into two sets: (1) the vectors with the highest ranks, and (2) the remaining vectors. The base vector and the vector which determines the search direction are chosen at random from the first set. The other vector is chosen at random from the second set.
Despite being a ranking process, SR has been adopted by NIAs which do not rank solutions, such as DE.

Zhang et al. [156] used SR in a DE variant based on [90]. $P_f$ was defined by a dynamic parameter control mechanism (high value at the beginning, low value at the end).

Liu et al. [73, 71] also used SR in DE and proposed the concept of directional information related to the choice of the most convenient search direction based on the DE mutation operator.

Fan et al. [30] ranked vectors with SR before the DE operators are applied. The population is split into two sets: (1) the vectors with the highest ranks, and (2) the remaining vectors. The base vector and the vector which determines the search direction are chosen at random from the first set. The other vector is chosen at random from the second set.
Stochastic ranking (SR)

Adapted to other NIAs

- Leguizamón and Coello Coello [63] added SR to an ACO version for dealing with CNOPs. A comparison against traditional penalty functions showed that SR provided better and more robust results.

- Fonseca et al. [32] used ACO with SR to solve discrete structural optimization problems.

- Mallipeddi et al. [77] proposed a two-population evolutionary programming (EP) approach with an external memory to store solutions based on an Euclidean distance measure that aimed to promote diversity. SR was compared against the feasibility rules.
Adapted to other NIAs

- Leguizamón and Coello Coello [63] added SR to an ACO version for dealing with CNOPs. A comparison against traditional penalty functions showed that SR provided better and more robust results.
- Fonseca et al. [32] used ACO with SR to solve discrete structural optimization problems.
- Mallipeddi et al. [77] proposed a two-population evolutionary programming (EP) approach with an external memory to store solutions based on an Euclidean distance measure that aimed to promote diversity. SR was compared against the feasibility rules.
Stochastic ranking (SR)

Adapted to other NIAs

- Leguizamón and Coello Coello [63] added SR to an ACO version for dealing with CNOPs. A comparison against traditional penalty functions showed that SR provided better and more robust results.

- Fonseca et al. [32] used ACO with SR to solve discrete structural optimization problems.

- Mallipeddi et al. [77] proposed a two-population evolutionary programming (EP) approach with an external memory to store solutions based on an Euclidean distance measure that aimed to promote diversity. SR was compared against the feasibility rules.
Huan-Tong et al. [44] used SR with its original search algorithm, an evolution strategy, (ES) for solving reactive power optimization problems.

Runarsson and Yao [115] improved their ES by adding a differential mutation similar to that used in DE. The authors concluded that a good constraint-handling mechanism needs to be coupled to an appropriate search engine.

SR has been further developed by Runarsson and Yao [113] in one of the earliest approaches focused on using fitness approximation for constrained numerical optimization ($k$-nearest-neighbors was adopted for such purpose).
Huan-Tong et al. [44] used SR with its original search algorithm, an evolution strategy, (ES) for solving reactive power optimization problems.

Runarsson and Yao [115] improved their ES by adding a differential mutation similar to that used in DE. The authors concluded that a good constraint-handling mechanism needs to be coupled to an appropriate search engine.

SR has been further developed by Runarsson and Yao [113] in one of the earliest approaches focused on using fitness approximation for constrained numerical optimization (k-nearest-neighbors was adopted for such purpose).
Stochastic ranking (SR)

Applications and other studies

- Huan-Tong et al. [44] used SR with its original search algorithm, an evolution strategy, (ES) for solving reactive power optimization problems.

- Runarsson and Yao [115] improved their ES by adding a differential mutation similar to that used in DE. The authors concluded that a good constraint-handling mechanism needs to be coupled to an appropriate search engine.

- SR has been further developed by Runarsson and Yao [113] in one of the earliest approaches focused on using fitness approximation for constrained numerical optimization ($k$-nearest-neighbors was adopted for such purpose).
Introduction
- The problem of interest
- Some important concepts
- Mathematical-programming methods
- Why alternative methods?

The early years
- Penalty functions
- Decoders
- Special operators
- Separation of objective function and constraints
- General comments

Current constraint-handling techniques
- Feasibility rules
- Stochastic ranking
- \( \epsilon \)-constrained method
- Novel penalty functions
- Novel special operators
- Multi-objective concepts
- Ensemble of constraint-handling techniques

Summary and current trends
- A bird’s eye view
- Current trends
Proposed by Takahama and Sakai [130]. It transforms a CNOP into an unconstrained numerical optimization problem.

Two main components:
- A relaxation of the limit to consider a solution as feasible.
- A lexicographical ordering mechanism in which the minimization of the sum of constraint violation precedes the objective function.

The value $\varepsilon > 0$, determines the so-called $\varepsilon$-level comparisons between a pair of solutions $\tilde{x}_1$ and $\tilde{x}_2$ with objective function values $f(\tilde{x}_1)$ and $f(\tilde{x}_2)$ and sums of constraint violation $\phi(\tilde{x}_1)$ and $\phi(\tilde{x}_2)$. 
\( \varepsilon \)-constrained method

Proposed by Takahama and Sakai [130]. It transforms a CNOP into an unconstrained numerical optimization problem.

Two main components:

- A relaxation of the limit to consider a solution as feasible.
- A lexicographical ordering mechanism in which the minimization of the sum of constraint violation precedes the objective function.

The value \( \varepsilon > 0 \), determines the so-called \( \varepsilon \)-level comparisons between a pair of solutions \( \tilde{x}_1 \) and \( \tilde{x}_2 \) with objective function values \( f(\tilde{x}_1) \) and \( f(\tilde{x}_2) \) and sums of constraint violation \( \phi(\tilde{x}_1) \) and \( \phi(\tilde{x}_2) \).
** Proposed by Takahama and Sakai [130]. It transforms a CNOP into an unconstrained numerical optimization problem. 

** Two main components:
  - A relaxation of the limit to consider a solution as feasible.
  - A lexicographical ordering mechanism in which the minimization of the sum of constraint violation precedes the objective function.

** The value \( \varepsilon > 0 \), determines the so-called \( \varepsilon \)-level comparisons between a pair of solutions \( \tilde{x}_1 \) and \( \tilde{x}_2 \) with objective function values \( f(\tilde{x}_1) \) and \( f(\tilde{x}_2) \) and sums of constraint violation \( \phi(\tilde{x}_1) \) and \( \phi(\tilde{x}_2) \).
$\varepsilon$-constrained method

\[(f(\vec{x}_1), \phi(\vec{x}_1)) \leq_{\varepsilon} (f(\vec{x}_2), \phi(\vec{x}_2)) \iff \begin{align*}
&f(\vec{x}_1) \leq f(\vec{x}_2), \quad \text{if } \phi(\vec{x}_1), \phi(\vec{x}_2) \leq \varepsilon \\
&f(\vec{x}_1) \leq f(\vec{x}_2), \quad \text{if } \phi(\vec{x}_1) = \phi(\vec{x}_2) \\
&\phi(\vec{x}_1) < \phi(\vec{x}_2), \quad \text{otherwise}
\end{align*}\]

\[(f(\vec{x}_1), \phi(\vec{x}_1)) <_{\varepsilon} (f(\vec{x}_2), \phi(\vec{x}_2)) \iff \begin{align*}
&f(\vec{x}_1) < f(\vec{x}_2), \quad \text{if } \phi(\vec{x}_1), \phi(\vec{x}_2) \leq \varepsilon \\
&f(\vec{x}_1) < f(\vec{x}_2), \quad \text{if } \phi(\vec{x}_1) = \phi(\vec{x}_2) \\
&\phi(\vec{x}_1) < \phi(\vec{x}_2), \quad \text{otherwise}
\end{align*}\]
If both solutions in the pairwise comparison are feasible, slightly infeasible (as determined by the $\varepsilon$ value) or even if they have the same sum of constraint violation, they are compared using their objective function values.

- If both solutions are infeasible, they are compared based on their sum of constraint violation.
- If $\varepsilon = \infty$, the $\varepsilon$-level comparison works by using only the objective function values as the comparison criteria.
- If $\varepsilon = 0$, then the $\varepsilon$-level comparisons $\prec_0$ and $\preceq_0$ are equivalent to a lexicographical ordering (i.e., $\phi(\bar{x})$ precedes $f(\bar{x})$).
If both solutions in the pairwise comparison are feasible, slightly infeasible (as determined by the $\varepsilon$ value) or even if they have the same sum of constraint violation, they are compared using their objective function values.

If both solutions are infeasible, they are compared based on their sum of constraint violation.

If $\varepsilon = \infty$, the $\varepsilon$-level comparison works by using only the objective function values as the comparison criteria.

If $\varepsilon = 0$, then the $\varepsilon$-level comparisons $\prec_0$ and $\preceq_0$ are equivalent to a lexicographical ordering (i.e., $\phi(\bar{x})$ precedes $f(\bar{x})$).
-constrained method

If both solutions in the pairwise comparison are feasible, slightly infeasible (as determined by the \( \epsilon \) value) or even if they have the same sum of constraint violation, they are compared using their objective function values.

If both solutions are infeasible, they are compared based on their sum of constraint violation.

If \( \epsilon = \infty \), the \( \epsilon \)-level comparison works by using only the objective function values as the comparison criteria.

If \( \epsilon = 0 \), then the \( \epsilon \)-level comparisons \( \prec_0 \) and \( \preceq_0 \) are equivalent to a lexicographical ordering (i.e., \( \phi(\bar{x}) \) precedes \( f(\bar{x}) \)).
\(\varepsilon\)-constrained method

- If both solutions in the pairwise comparison are feasible, slightly infeasible (as determined by the \(\varepsilon\) value) or even if they have the same sum of constraint violation, they are compared using their objective function values.
- If both solutions are infeasible, they are compared based on their sum of constraint violation.
- If \(\varepsilon = \infty\), the \(\varepsilon\)-level comparison works by using only the objective function values as the comparison criteria.
- If \(\varepsilon = 0\), then the \(\varepsilon\)-level comparisons \(<_0\) and \(\leq_0\) are equivalent to a lexicographical ordering (i.e., \(\phi(\bar{x})\) precedes \(f(\bar{x})\)).
Adapted to other NIAs

- Takahama and Sakai used a similar approach called $\alpha$-constrained method into a GA [123]. Even mathematical programming methods have been used with this approach (Nealder-Mead) [124].
- Wang and Li also adopted the $\alpha$-constrained method in [140], using DE as their search engine.
- Takahama and Sakai adopted the $\varepsilon$-constrained method in PSO [125], and mainly in DE [126].
- Takahama et al. used their constraint-handling technique in a hybrid PSO-GA [130].
Adapted to other NIAs

- Takahama and Sakai used a similar approach called \( \alpha \)-constrained method into a GA [123]. Even mathematical programming methods have been used with this approach (Nealder-Mead) [124].

- Wang and Li also adopted the \( \alpha \)-constrained method in [140], using DE as their search engine.

- Takahama and Sakai adopted the \( \varepsilon \)-constrained method in PSO [125], and mainly in DE [126].

- Takahama et al. used their constraint-handling technique in a hybrid PSO-GA [130].
Adapted to other NIAs

- Takahama and Sakai used a similar approach called $\alpha$-constrained method into a GA [123]. Even mathematical programming methods have been used with this approach (Nealder-Mead) [124].
- Wang and Li also adopted the $\alpha$-constrained method in [140], using DE as their search engine.
- Takahama and Sakai adopted the $\varepsilon$-constrained method in PSO [125], and mainly in DE [126].
- Takahama et al. used their constraint-handling technique in a hybrid PSO-GA [130].
Adapted to other NIAs

- Takahama and Sakai used a similar approach called $\alpha$-constrained method into a GA [123]. Even mathematical programming methods have been used with this approach (Nealder-Mead) [124].
- Wang and Li also adopted the $\alpha$-constrained method in [140], using DE as their search engine.
- Takahama and Sakai adopted the $\varepsilon$-constrained method in PSO [125], and mainly in DE [126].
- Takahama et al. used their constraint-handling technique in a hybrid PSO-GA [130].
The $\epsilon$ value is fine-tuned by Takahama and Sakai with dynamic [124], and adaptive parameter control mechanisms [127].

- Zeng et al. [155] also proposed a dynamic decreasing mechanism inspired in [36].
- A gradient-based mutation was added to the DE-based approach by the same authors in [126] and by Zhang et al. in an EA in [157].
- In [128], Takahama and Sakai improved their approach by adding a decreasing probability on the use of the gradient-based mutation. They also introduced two new mechanisms to deal with boundary constraints (reflecting back and assigning the limit value).
- The authors in [129] added an archive to store solutions and the ability of a vector to generate more than one trial vector.
The $\varepsilon$ value is fine-tuned by Takahama and Sakai with dynamic [124], and adaptive parameter control mechanisms [127]. Zeng et al. [155] also proposed a dynamic decreasing mechanism inspired in [36]. A gradient-based mutation was added to the DE-based approach by the same authors in [126] and by Zhang et al. in an EA in [157]. In [128], Takahama and Sakai improved their approach by adding a decreasing probability on the use of the gradient-based mutation. They also introduced two new mechanisms to deal with boundary constraints (reflecting back and assigning the limit value). The authors in [129] added an archive to store solutions and the ability of a vector to generate more than one trial vector.
The $\varepsilon$ value is fine-tuned by Takahama and Sakai with dynamic [124], and adaptive parameter control mechanisms [127].

Zeng et al. [155] also proposed a dynamic decreasing mechanism inspired in [36].

A gradient-based mutation was added to the DE-based approach by the same authors in [126] and by Zhang et al. in an EA in [157].

In [128], Takahama and Sakai improved their approach by adding a decreasing probability on the use of the gradient-based mutation. They also introduced two new mechanisms to deal with boundary constraints (reflecting back and assigning the limit value).

The authors in [129] added an archive to store solutions and the ability of a vector to generate more than one trial vector.
-constrained method

**Improvements**

- The $\varepsilon$ value is fine-tuned by Takahama and Sakai with dynamic [124], and adaptive parameter control mechanisms [127].
- Zeng et al. [155] also proposed a dynamic decreasing mechanism inspired in [36].
- A gradient-based mutation was added to the DE-based approach by the same authors in [126] and by Zhang et al. in an EA in [157].
- In [128], Takahama and Sakai improved their approach by adding a decreasing probability on the use of the gradient-based mutation. They also introduced two new mechanisms to deal with boundary constraints (reflecting back and assigning the limit value).
- The authors in [129] added an archive to store solutions and the ability of a vector to generate more than one trial vector.
The $\varepsilon$ value is fine-tuned by Takahama and Sakai with dynamic [124], and adaptive parameter control mechanisms [127]. Zeng et al. [155] also proposed a dynamic decreasing mechanism inspired in [36].

A gradient-based mutation was added to the DE-based approach by the same authors in [126] and by Zhang et al. in an EA in [157].

In [128], Takahama and Sakai improved their approach by adding a decreasing probability on the use of the gradient-based mutation. They also introduced two new mechanisms to deal with boundary constraints (reflecting back and assigning the limit value).

The authors in [129] added an archive to store solutions and the ability of a vector to generate more than one trial vector.
NIAs turning to the $\varepsilon$-constrained method

- $\varepsilon$-jDE by Brest et al. [12], where different DE variants, parameter self-adaptation (including $\varepsilon$), and population reduction were employed.

- An improved version called jDEsoco was proposed by Brest et al. in [13], where an ageing mechanism to replace those solutions stagnated in a local optimum was added. Moreover, only the 60% of the population was compared by the $\varepsilon$-constrained method and the remaining 40% was compared by only using the objective function value.

- Mezura-Montes et al. used the $\varepsilon$-constrained method in ABC [86]. A dynamic mechanism for the equality constraints tolerance was considered. The results obtained outperformed those reported by a previous ABC version with the feasibility rules [82].
NIAs turning to the $\varepsilon$-constrained method

- $\varepsilon$-jDE by Brest et al. [12], where different DE variants, parameter self-adaptation (including $\varepsilon$), and population reduction were employed.

- An improved version called jDEsoco was proposed by Brest et al. in [13], where an ageing mechanism to replace those solutions stagnated in a local optimum was added. Moreover, only the 60% of the population was compared by the $\varepsilon$-constrained method and the remaining 40% was compared by only using the objective function value.

- Mezura-Montes et al. used the $\varepsilon$-constrained method in ABC [86]. A dynamic mechanism for the equality constraints tolerance was considered. The results obtained outperformed those reported by a previous ABC version with the feasibility rules [82].
**\(\varepsilon\)-constrained method**

### NIAs turning to the \(\varepsilon\)-constrained method

- \(\varepsilon\)-jDE by Brest et al. [12], where different DE variants, parameter self-adaptation (including \(\varepsilon\)), and population reduction were employed.

- An improved version called jDEsoco was proposed by Brest et al. in [13], where an ageing mechanism to replace those solutions stagnated in a local optimum was added. Moreover, only the 60% of the population was compared by the \(\varepsilon\)-constrained method and the remaining 40% was compared by only using the objective function value.

- Mezura-Montes et al. used the \(\varepsilon\)-constrained method in ABC [86]. A dynamic mechanism for the equality constraints tolerance was considered. The results obtained outperformed those reported by a previous ABC version with the feasibility rules [82].
Outline

1. Introduction
   - The problem of interest
   - Some important concepts
   - Mathematical-programming methods
   - Why alternative methods?

2. The early years
   - Penalty functions
   - Decoders
   - Special operators
   - Separation of objective function and constraints
   - General comments

3. Current constraint-handling techniques
   - Feasibility rules
   - Stochastic ranking
   - $\varepsilon$-constrained method
   - Novel penalty functions
   - Novel special operators
   - Multi-objective concepts
   - Ensemble of constraint-handling techniques

4. Summary and current trends
   - A bird’s eye view
   - Current trends
The most popular approach recently found regarding penalization-based approaches is the adaptive penalty function. Dynamic penalty functions, which adopt the current generation number to control the decrement of the penalty factor, are still popular.
Novel penalty functions

- The most popular approach recently found regarding penalization-based approaches is the adaptive penalty function.
- Dynamic penalty functions, which adopt the current generation number to control the decrement of the penalty factor, are still popular.
Farmani and Wright [31] proposed a two-part adaptive penalty function. The first part increases the fitness of the infeasible solutions with a better value of the objective function with respect to the best solution in the current population. The second part modifies the fitness values of the worst infeasible solutions.

Tessema and Yen [133] used the number of feasible solutions in the current population to determine penalization values so as to favor slightly infeasible solutions having a good objective function value.
Adaptive penalty functions

Farmani and Wright [31] proposed a two-part adaptive penalty function. The first part increases the fitness of the infeasible solutions with a better value of the objective function with respect to the best solution in the current population. The second part modifies the fitness values of the worst infeasible solutions.

Tessema and Yen [133] used the number of feasible solutions in the current population to determine penalization values so as to favor slightly infeasible solutions having a good objective function value.
Novel penalty functions

Adaptive penalty functions


- He et al. [39] used two PSO algorithms, one to co-evolve penalty factors and the other one to evolve solutions to the optimization problem.

- Wu [151] proposed an artificial immune system (AIS) where an adaptive penalty function was defined to assign its affinity to each antibody.
Novel penalty functions

Adaptive penalty functions


- He et al. [39] used two PSO algorithms, one to co-evolve penalty factors and the other one to evolve solutions to the optimization problem.

- Wu [151] proposed an artificial immune system (AIS) where an adaptive penalty function was defined to assign its affinity to each antibody.

He et al. [39] used two PSO algorithms, one to co-evolve penalty factors and the other one to evolve solutions to the optimization problem.

Wu [151] proposed an artificial immune system (AIS) where an adaptive penalty function was defined to assign its affinity to each antibody.
Novel penalty functions

Dynamic penalty functions

- Tasgetiren and Suganathan [132] used a dynamic penalty function coupled with a multi-population DE algorithm where each populations evolved independently.

- Puzzi and Carpinteri [104] explored a dynamic penalty function based on multiplications instead of summations in a GA-based approach.
Novel penalty functions

Dynamic penalty functions

- Tasgetiren and Suganathan [132] used a dynamic penalty function coupled with a multi-population DE algorithm where each population evolved independently.
- Puzzi and Carpinteri [104] explored a dynamic penalty function based on multiplications instead of summations in a GA-based approach.
Deb and Datta [26] obtained suitable penalty factors as follows:

- A bi-objective problem (original objective function and sum of constraint violation $\phi$, restricted by a tolerance value) was solved by a MOEA.
- A cubic curve to approximate the current obtained Pareto front was generated by using four points whose $\phi$ values were below a small tolerance.
- The penalty factor was then defined by calculating the corresponding slope at $\phi = 0$.
- After that, a traditional static penalty function was used to solve the original CNOP by using a local search algorithm (Matlab’s `fmincon()` procedure was used by the authors) using the solution with the lowest $\phi$ value from the population of the MOEA as the starting point for the search.
Deb and Datta [26] obtained suitable penalty factors as follows:

A bi-objective problem (original objective function and sum of constraint violation $\phi$, restricted by a tolerance value) was solved by a MOEA.

A cubic curve to approximate the current obtained Pareto front was generated by using four points whose $\phi$ values were below a small tolerance.

The penalty factor was then defined by calculating the corresponding slope at $\phi = 0$.

After that, a traditional static penalty function was used to solve the original CNOP by using a local search algorithm (Matlab’s `fmincon()` procedure was used by the authors) using the solution with the lowest $\phi$ value from the population of the MOEA as the starting point for the search.
Novel penalty functions

Static penalty functions

- Deb and Datta [26] obtained suitable penalty factors as follows:
- A bi-objective problem (original objective function and sum of constraint violation $\phi$, restricted by a tolerance value) was solved by a MOEA.
- A cubic curve to approximate the current obtained Pareto front was generated by using four points whose $\phi$ values were below a small tolerance.
- The penalty factor was then defined by calculating the corresponding slope at $\phi = 0$.
- After that, a traditional static penalty function was used to solve the original CNOP by using a local search algorithm (Matlab’s `fmincon()` procedure was used by the authors) using the solution with the lowest $\phi$ value from the population of the MOEA as the starting point for the search.
Novel penalty functions

Static penalty functions

- Deb and Datta [26] obtained suitable penalty factors as follows:
- A bi-objective problem (original objective function and sum of constraint violation $\phi$, restricted by a tolerance value) was solved by a MOEA
- A cubic curve to approximate the current obtained Pareto front was generated by using four points whose $\phi$ values were below a small tolerance.
- The penalty factor was then defined by calculating the corresponding slope at $\phi = 0$.
- After that, a traditional static penalty function was used to solve the original CNOP by using a local search algorithm (Matlab’s \texttt{fmincon}() procedure was used by the authors) using the solution with the lowest $\phi$ value from the population of the MOEA as the starting point for the search.
Deb and Datta [26] obtained suitable penalty factors as follows:

A bi-objective problem (original objective function and sum of constraint violation $\phi$, restricted by a tolerance value) was solved by a MOEA.

A cubic curve to approximate the current obtained Pareto front was generated by using four points whose $\phi$ values were below a small tolerance.

The penalty factor was then defined by calculating the corresponding slope at $\phi = 0$.

After that, a traditional static penalty function was used to solve the original CNOP by using a local search algorithm (Matlab’s `fmincon()` procedure was used by the authors) using the solution with the lowest $\phi$ value from the population of the MOEA as the starting point for the search.
Novel penalty functions

Static penalty functions

- In [23], Datta and Deb extended their approach to deal with equality constraints.

- Two main changes:
  - The punishment provided by the penalty value obtained by the bi-objective problem was increased if the local search failed to generate a feasible solution.
  - The small tolerance used for choosing the four points employed to approximate the cubic curve was relaxed.
Novel penalty functions

Static penalty functions

- In [23], Datta and Deb extended their approach to deal with equality constraints.
- Two main changes:
  - The punishment provided by the penalty value obtained by the bi-objective problem was increased if the local search failed to generate a feasible solution.
  - The small tolerance used for choosing the four points employed to approximate the cubic curve was relaxed.
Outline

1. Introduction
   - The problem of interest
   - Some important concepts
   - Mathematical-programming methods
   - Why alternative methods?

2. The early years
   - Penalty functions
   - Decoders
   - Special operators
   - Separation of objective function and constraints
   - General comments

3. Current constraint-handling techniques
   - Feasibility rules
   - Stochastic ranking
   - $\varepsilon$-constrained method
   - Novel penalty functions
   - Novel special operators
   - Multi-objective concepts
   - Ensemble of constraint-handling techniques

4. Summary and current trends
   - A bird’s eye view
   - Current trends
The recent proposals based on the use of special operators that have been revised here emphasize the current focus on generating proposals which are easier to generalize.

Leguizamón and Coello Coello [62] proposed a boundary operator based on conducting a binary search between a feasible and an infeasible solution. Furthermore, the authors designed a strategy to select which constraint (if more than one is present in a CNOP) is analyzed.

The search algorithm was an ACO variant for continuous search spaces.

The approach needed an additional constraint-handling technique (a penalty function was used in this case).
Novel special operators

- The recent proposals based on the use of special operators that have been revised here emphasize the current focus on generating proposals which are easier to generalize.

- Leguizamón and Coello Coello [62] proposed a boundary operator based on conducting a binary search between a feasible and an infeasible solution. Furthermore, the authors designed a strategy to select which constraint (if more than one is present in a CNOP) is analyzed.

- The search algorithm was an ACO variant for continuous search spaces.

- The approach needed an additional constraint-handling technique (a penalty function was used in this case).
The recent proposals based on the use of special operators that have been revised here emphasize the current focus on generating proposals which are easier to generalize.

Leguizamón and Coello Coello [62] proposed a boundary operator based on conducting a binary search between a feasible and an infeasible solution. Furthermore, the authors designed a strategy to select which constraint (if more than one is present in a CNOP) is analyzed.

The search algorithm was an ACO variant for continuous search spaces.

The approach needed an additional constraint-handling technique (a penalty function was used in this case).
Novel special operators

- The recent proposals based on the use of special operators that have been revised here emphasize the current focus on generating proposals which are easier to generalize.

- Leguizamón and Coello Coello [62] proposed a boundary operator based on conducting a binary search between a feasible and an infeasible solution. Furthermore, the authors designed a strategy to select which constraint (if more than one is present in a CNOP) is analyzed.

- The search algorithm was an ACO variant for continuous search spaces.

- The approach needed an additional constraint-handling technique (a penalty function was used in this case).
Huang et al. [45] proposed a boundary operator in a two-population approach.

The first population evolves by using DE as the search engine, based only on the objective function value (regardless of feasibility).

The second population stores only feasible solutions and the boundary operator uses solutions from both populations to generate new solutions, through the application of the bisection method in the boundaries of the feasible region.

The Nelder-Mead simplex method was used as a local search operator.

It does not require an additional constraint-handling technique, but a feasible solutions is needed at the beginning of the process.
Huang et al. [45] proposed a boundary operator in a two-population approach.

The first population evolves by using DE as the search engine, based only on the objective function value (regardless of feasibility).

The second population stores only feasible solutions and the boundary operator uses solutions from both populations to generate new solutions, through the application of the bisection method in the boundaries of the feasible region.

The Nelder-Mead simplex method was used as a local search operator.

It does not require an additional constraint-handling technique, but a feasible solution is needed at the beginning of the process.
Huang et al. [45] proposed a boundary operator in a two-population approach.

The first population evolves by using DE as the search engine, based only on the objective function value (regardless of feasibility).

The second population stores only feasible solutions and the boundary operator uses solutions from both populations to generate new solutions, through the application of the bisection method in the boundaries of the feasible region.

The Nelder-Mead simplex method was used as a local search operator.

It does not require an additional constraint-handling technique, but a feasible solution is needed at the beginning of the process.
Huang et al. [45] proposed a boundary operator in a two-population approach.

The first population evolves by using DE as the search engine, based only on the objective function value (regardless of feasibility).

The second population stores only feasible solutions and the boundary operator uses solutions from both populations to generate new solutions, through the application of the bisection method in the boundaries of the feasible region.

The Nelder-Mead simplex method was used as a local search operator.

It does not require an additional constraint-handling technique, but a feasible solution is needed at the beginning of the process.
Huang et al. [45] proposed a boundary operator in a two-population approach.

The first population evolves by using DE as the search engine, based only on the objective function value (regardless of feasibility).

The second population stores only feasible solutions and the boundary operator uses solutions from both populations to generate new solutions, through the application of the bisection method in the boundaries of the feasible region.

The Nelder-Mead simplex method was used as a local search operator.

It does not require an additional constraint-handling technique, but a feasible solutions is needed at the beginning of the process.
Wanner et al. [149] proposed the Constraint Quadratic Approximation (CQA), which is a special operator designed to restrict an evolutionary algorithm (a GA in this case) to sample solutions inside an object with the same dimensions of the feasible region of the search space.

- This is achieved by a second-order approximation of the objective function and one equality constraint.
- A static penalty function was used to guide the GA search and the equality constraint was transformed into two inequality constraints by using a small $\epsilon$ tolerance.
Wanner et al. [149] proposed the Constraint Quadratic Approximation (CQA), which is a special operator designed to restrict an evolutionary algorithm (a GA in this case) to sample solutions inside an object with the same dimensions of the feasible region of the search space.

This is achieved by a second-order approximation of the objective function and one equality constraint.

A static penalty function was used to guide the GA search and the equality constraint was transformed into two inequality constraints by using a small tolerance.
Wanner et al. [149] proposed the Constraint Quadratic Approximation (CQA), which is a special operator designed to restrict an evolutionary algorithm (a GA in this case) to sample solutions inside an object with the same dimensions of the feasible region of the search space.

This is achieved by a second-order approximation of the objective function and one equality constraint.

A static penalty function was used to guide the GA search and the equality constraint was transformed into two inequality constraints by using a small $\epsilon$ tolerance.
Novel special operators

Constraint satisfaction

- Peconick et al. [102] proposed the Constraint Quadratic Approximation for Multiple Equality Constraints (CQA-MEC).
  - An iterative projection algorithm was able to find points satisfying the approximated quadratic constraints with a low computational overhead.
  - It still requires the static penalty function to work.
- Araujo et al. [8] extended the previous approaches to deal with multiple inequality constraints by using a special operator in which the locally convex inequality constraints are approximated by quadratic functions, while the locally non-convex inequality constraints are approximated by linear functions.
- The dependence of the static penalty function remains in this last approach.
Novel special operators

**Constraint satisfaction**

- Peconick et al. [102] proposed the Constraint Quadratic Approximation for Multiple Equality Constraints (CQA-MEC).
  - An iterative projection algorithm was able to find points satisfying the approximated quadratic constraints with a low computational overhead.

- It still requires the static penalty function to work.

- Araujo et al. [8] extended the previous approaches to deal with multiple inequality constraints by using a special operator in which the locally convex inequality constraints are approximated by quadratic functions, while the locally non-convex inequality constraints are approximated by linear functions.

- The dependence of the static penalty function remains in this last approach.
Peconick et al. [102] proposed the Constraint Quadratic Approximation for Multiple Equality Constraints (CQA-MEC).

An iterative projection algorithm was able to find points satisfying the approximated quadratic constraints with a low computational overhead.

It still requires the static penalty function to work.

Araujo et al. [8] extended the previous approaches to deal with multiple inequality constraints by using a special operator in which the locally convex inequality constraints are approximated by quadratic functions, while the locally non-convex inequality constraints are approximated by linear functions.

The dependence of the static penalty function remains in this last approach.
### Constraint satisfaction

- Peconick et al. [102] proposed the Constraint Quadratic Approximation for Multiple Equality Constraints (CQA-MEC).
- An iterative projection algorithm was able to find points satisfying the approximated quadratic constraints with a low computational overhead.
- It still requires the static penalty function to work.
- Araujo et al. [8] extended the previous approaches to deal with multiple inequality constraints by using a special operator in which the locally convex inequality constraints are approximated by quadratic functions, while the locally non-convex inequality constraints are approximated by linear functions.
- The dependence of the static penalty function remains in this last approach.
Peconick et al. [102] proposed the Constraint Quadratic Approximation for Multiple Equality Constraints (CQA-MEC). An iterative projection algorithm was able to find points satisfying the approximated quadratic constraints with a low computational overhead.

It still requires the static penalty function to work.

Araujo et al. [8] extended the previous approaches to deal with multiple inequality constraints by using a special operator in which the locally convex inequality constraints are approximated by quadratic functions, while the locally non-convex inequality constraints are approximated by linear functions.

The dependence of the static penalty function remains in this last approach.
Constraint satisfaction

- Ullah et al. [134] proposed an agent-based memetic algorithm in which the authors adopt a special local operator for equality constraints.

- It is applied to some individuals in the population as follows: the satisfaction of a randomly chosen equality constraint is verified for a given solution. If it is not satisfied, a decision variable, also chosen at random, is updated with the aim to satisfy it. If the constraint is indeed satisfied, two other variables are modified in such a way that the constraint is still satisfied (i.e., the constraint is sampled).

- This special operator is only applied during the early stages of the search because it reduces the diversity in the population.
Constraint satisfaction

- Ullah et al.[134] proposed an agent-based memetic algorithm in which the authors adopt a special local operator for equality constraints.

- It is applied to some individuals in the population as follows: the satisfaction of a randomly chosen equality constraint is verified for a given solution. If it is not satisfied, a decision variable, also chosen at random, is updated with the aim to satisfy it. If the constraint is indeed satisfied, two other variables are modified in such a way that the constraint is still satisfied (i.e., the constraint is sampled).

- This special operator is only applied during the early stages of the search because it reduces the diversity in the population.
Novel special operators

Constraint satisfaction

- Ullah et al. [134] proposed an agent-based memetic algorithm in which the authors adopt a special local operator for equality constraints.

- It is applied to some individuals in the population as follows: the satisfaction of a randomly chosen equality constraint is verified for a given solution. If it is not satisfied, a decision variable, also chosen at random, is updated with the aim to satisfy it. If the constraint is indeed satisfied, two other variables are modified in such a way that the constraint is still satisfied (i.e., the constraint is sampled).

- This special operator is only applied during the early stages of the search because it reduces the diversity in the population.
Spadoni and Stefanini [121] transformed a CNOP into an unconstrained search problem by sampling feasible directions instead of solutions of a CNOP.

Three special operators, related to feasible directions for box constraints, linear inequality constraints, and quadratic inequality constraints, are utilized to generate new solutions by using DE as the search algorithm.

The main contribution of the approach is that it transforms a CNOP into an unconstrained search problem without using a penalty function. However, it cannot deal with nonlinear (either equality or inequality) constraints.
Novel special operators

Feasible directions

- Spadoni and Stefanini [121] transformed a CNOP into an unconstrained search problem by sampling feasible directions instead of solutions of a CNOP.
- Three special operators, related to feasible directions for box constraints, linear inequality constraints, and quadratic inequality constraints, are utilized to generate new solutions by using DE as the search algorithm.
- The main contribution of the approach is that it transforms a CNOP into an unconstrained search problem without using a penalty function. However, it cannot deal with nonlinear (either equality or inequality) constraints.
### Novel special operators

#### Feasible directions

- Spadoni and Stefanini [121] transformed a CNOP into an unconstrained search problem by sampling feasible directions instead of solutions of a CNOP.

- Three special operators, related to feasible directions for box constraints, linear inequality constraints, and quadratic inequality constraints, are utilized to generate new solutions by using DE as the search algorithm.

- The main contribution of the approach is that it transforms a CNOP into an unconstrained search problem without using a penalty function. However, it cannot deal with nonlinear (either equality or inequality) constraints.
Lu and Chen [75] proposed an approach called self-adaptive velocity particle swarm optimization (SAVPSO).

Three elements:
- The position of the feasible region with respect to the whole search space.
- The connectivity and the shape of the feasible region.
- The ratio of the feasible region with respect to the search space.

The velocity formula was modified in such a way that each particle has the ability to self-adjust its velocity according to the aforementioned features of the feasible region.
Novel special operators

General operators made special

- Lu and Chen [75] proposed an approach called self-adaptive velocity particle swarm optimization (SAVPSO).

- Three elements:
  - The position of the feasible region with respect to the whole search space.
  - The connectivity and the shape of the feasible region.
  - The ratio of the feasible region with respect to the search space.

- The velocity formula was modified in such a way that each particle has the ability to self-adjust its velocity according to the aforementioned features of the feasible region.
Lu and Chen [75] proposed an approach called self-adaptive velocity particle swarm optimization (SAVPSO).

Three elements:
- The position of the feasible region with respect to the whole search space.
- The connectivity and the shape of the feasible region.
- The ratio of the feasible region with respect to the search space.

The velocity formula was modified in such a way that each particle has the ability to self-adjust its velocity according to the aforementioned features of the feasible region.
Novel special operators

Constraint satisfaction

- Wu et al. [152] and Li & Li [65] modified variation operators in NIAs in such a way that the recombination of feasible and infeasible solutions led to the generation of more feasible solutions.
- An adaptive mechanism to maintain infeasible solutions was added to the approach.
- This latter version was specifically based on DE’s variation operators [65].
Wu et al. [152] and Li & Li [65] modified variation operators in NIAs in such a way that the recombination of feasible and infeasible solutions led to the generation of more feasible solutions.

An adaptive mechanism to maintain infeasible solutions was added to the approach.

This latter version was specifically based on DE’s variation operators [65].
Novel special operators

Constraint satisfaction

- Wu et al. [152] and Li & Li [65] modified variation operators in NIAs in such a way that the recombination of feasible and infeasible solutions led to the generation of more feasible solutions.

- An adaptive mechanism to maintain infeasible solutions was added to the approach.

- This latter version was specifically based on DE’s variation operators [65].
Outline

1 Introduction
   - The problem of interest
   - Some important concepts
   - Mathematical-programming methods
   - Why alternative methods?

2 The early years
   - Penalty functions
   - Decoders
   - Special operators
   - Separation of objective function and constraints
   - General comments

3 Current constraint-handling techniques
   - Feasibility rules
   - Stochastic ranking
   - $\varepsilon$-constrained method
   - Novel penalty functions
   - Novel special operators
   - Multi-objective concepts
   - Ensemble of constraint-handling techniques

4 Summary and current trends
   - A bird’s eye view
   - Current trends
Multi-objective concepts

Despite the fact that empirical evidence has suggested that multi-objective concepts are not well-suited to solve CNOPs, there are highly competitive constraint-handling techniques based on such concepts.

The use of transformation of a CNOP into a bi-objective optimization problem (objective function and sum of constraint violation) has been preferred over considering each constraint as a separate objective.
Despite the fact that empirical evidence has suggested that multi-objective concepts are not well-suited to solve CNOPs, there are highly competitive constraint-handling techniques based on such concepts.

The use of transformation of a CNOP into a bi-objective optimization problem (objective function and sum of constraint violation) has been preferred over considering each constraint as a separate objective.
Multi-objective concepts

Bi-objective problem

- Ray et al. [109] proposed the Infeasibility Driven Evolutionary Algorithm (IDEA).
- The second objective is the constraint violation measure, (zero value for feasible solutions and a sum of ranking values based on the violation per constraint).
- The union of parents and offspring is split in two sets, one with the feasible solutions and the other with the infeasible ones.
- Non-dominated sorting ranks both sets separately and, based on the proportion of desired feasible solutions, they are chosen first from the infeasible set and later on, the best ranked feasible solutions are chosen.
- SQP was added to IDEA in the Infeasibility Empowered Memetic Algorithm (IMEA) [120].
Multi-objective concepts

Bi-objective problem

- Ray et al. [109] proposed the Infeasibility Driven Evolutionary Algorithm (IDEA).
  
  The second objective is the constraint violation measure, (zero value for feasible solutions and a sum of ranking values based on the violation per constraint).

- The union of parents and offspring is split in two sets, one with the feasible solutions and the other with the infeasible ones.

- Non-dominated sorting ranks both sets separately and, based on the proportion of desired feasible solutions, they are chosen first from the infeasible set and later on, the best ranked feasible solutions are chosen.

- SQP was added to IDEA in the Infeasibility Empowered Memetic Algorithm (IMEA) [120].
Ray et al. [109] proposed the Infeasibility Driven Evolutionary Algorithm (IDEA).

The second objective is the constraint violation measure, (zero value for feasible solutions and a sum of ranking values based on the violation per constraint).

The union of parents and offspring is split in two sets, one with the feasible solutions and the other with the infeasible ones.

Non-dominated sorting ranks both sets separately and, based on the proportion of desired feasible solutions, they are chosen first from the infeasible set and later on, the best ranked feasible solutions are chosen.

SQP was added to IDEA in the Infeasibility Empowered Memetic Algorithm (IMEA) [120].
Multi-objective concepts

Bi-objective problem

- Ray et al. [109] proposed the Infeasibility Driven Evolutionary Algorithm (IDEA).
- The second objective is the constraint violation measure, (zero value for feasible solutions and a sum of ranking values based on the violation per constraint).
- The union of parents and offspring is split in two sets, one with the feasible solutions and the other with the infeasible ones.
- Non-dominated sorting ranks both sets separately and, based on the proportion of desired feasible solutions, they are chosen first from the infeasible set and later on, the best ranked feasible solutions are chosen.
- SQP was added to IDEA in the Infeasibility Empowered Memetic Algorithm (IMEA) [120].
Multi-objective concepts

Bi-objective problem

- Ray et al. [109] proposed the Infeasibility Driven Evolutionary Algorithm (IDEA).
- The second objective is the constraint violation measure, (zero value for feasible solutions and a sum of ranking values based on the violation per constraint).
- The union of parents and offspring is split in two sets, one with the feasible solutions and the other with the infeasible ones.
- Non-dominated sorting ranks both sets separately and, based on the proportion of desired feasible solutions, they are chosen first from the infeasible set and later on, the best ranked feasible solutions are chosen.
- SQP was added to IDEA in the Infeasibility Empowered Memetic Algorithm (IMEA) [120].
Wang et al. [147], in their adaptive trade-off model (ATM) divided the search in three phases based on the feasibility of solutions in the population:

- Only infeasible solutions (Pareto dominance)
- Feasible and infeasible solutions (fitness value based on feasible solutions ratio).
- Only feasible solutions (objective function).

Wang et al. [146] used the ATM with a NIA in which the variation operators were simplex crossover and one of two mutations.
Multi-objective concepts

Bi-objective problem

- Wang et al. [147], in their adaptive trade-off model (ATM) divided the search in three phases based on the feasibility of solutions in the population:
  - Only infeasible solutions (Pareto dominance)
  - Feasible and infeasible solutions (fitness value based on feasible solutions ratio).
  - Only feasible solutions (objective function).

- Wang et al. [146] used the ATM with a NIA in which the variation operators were simplex crossover and one of two mutations.
Multi-objective concepts

Bi-objective problem

- Wang et al. [145] added a shrinking mechanism to ATM in the Accelerated ATM (AATM).
- The ATM was coupled with DE in a recent approach [143], showing an improvement in the results.
- Liu et al. [72] used the ATM in an EA but with two main differences:
  - Good point set crossover was used to generate offspring.
  - Feasibility rules were the criteria to select solutions in the second stage of the ATM.
Multi-objective concepts

Bi-objective problem

- Wang et al. [145] added a shrinking mechanism to ATM in the Accelerated ATM (AATM).
- The ATM was coupled with DE in a recent approach [143], showing an improvement in the results.
- Liu et al. [72] used the ATM in an EA but with two main differences:
  - Good point set crossover was used to generate offspring.
  - Feasibility rules were the criteria to select solutions in the second stage of the ATM.
Multi-objective concepts

Bi-objective problem

- Wang et al. [145] added a shrinking mechanism to ATM in the Accelerated ATM (AATM).
- The ATM was coupled with DE in a recent approach [143], showing an improvement in the results.
- Liu et al. [72] used the ATM in an EA but with two main differences:
  - Good point set crossover was used to generate offspring.
  - Feasibility rules were the criteria to select solutions in the second stage of the ATM.
Multi-objective concepts

Bi-objective problem

- Li et al. [64] used a PSO algorithm in which Pareto dominance was used as a criterion in the pbest update process and in the selection of the local-best leaders in a neighborhood. The sum of constraint violation worked as a tie-breaker.

- Venter and Haftka [139] also adopted PSO as their search algorithm. However, the leader selection was based most of the time on the sum of constraint violation, while the rest of the time the criterion was one of the three following choices:
  - The original objective function.
  - The crowding distance.
  - Pareto dominance.
Multi-objective concepts

Bi-objective problem

Li et al. [64] used a PSO algorithm in which Pareto dominance was used as a criterion in the pbest update process and in the selection of the local-best leaders in a neighborhood. The sum of constraint violation worked as a tie-breaker.

Venter and Haftka [139] also adopted PSO as their search algorithm. However, the leader selection was based most of the time on the sum of constraint violation, while the rest of the time the criterion was one of the three following choices:

- The original objective function.
- The crowding distance.
- Pareto dominance.
Multi-objective concepts

Bi-objective problem

- Wang et al. [141] used a hybrid selection mechanism based on Pareto dominance and tournament selection into a Adaptive Bacterial Foraging Algorithm (ABFA).

- Wang et al. [144] proposed the use of Pareto dominance in a Hybrid Constrained EA (HCOEA). A global search carried out by an EA is coupled to a local search operator based on SPX.

- Wang et al. [148] proposed a steady state EA by applying orthogonal crossover to a randomly chosen set of solutions in the current population. After that, the non-dominated solutions obtained from the set of offspring are chosen. Alternative, solutions can also be chosen if they have a lower sum of constraint violation.
Multi-objective concepts

Bi-objective problem

- Wang et al. [141] used a hybrid selection mechanism based on Pareto dominance and tournament selection into a Adaptive Bacterial Foraging Algorithm (ABFA).

- Wang et al. [144] proposed the use of Pareto dominance in a Hybrid Constrained EA (HCOEA). A global search carried out by an EA is coupled to a local search operator based on SPX.

- Wang et al. [148] proposed a steady state EA by applying orthogonal crossover to a randomly chosen set of solutions in the current population. After that, the non-dominated solutions obtained from the set of offspring are chosen. Alternative, solutions can also be chosen if they have a lower sum of constraint violation.
Multi-objective concepts

Bi-objective problem

Wang et al. [141] used a hybrid selection mechanism based on Pareto dominance and tournament selection into a Adaptive Bacterial Foraging Algorithm (ABFA).

Wang et al. [144] proposed the use of Pareto dominance in a Hybrid Constrained EA (HCOEA). A global search carried out by an EA is coupled to a local search operator based on SPX.

Wang et al. [148] proposed a steady state EA by applying orthogonal crossover to a randomly chosen set of solutions in the current population. After that, the non-dominated solutions obtained from the set of offspring are chosen. Alternative, solutions can also be chosen if they have a lower sum of constraint violation.
Multi-objective concepts

Three-objective problem

- Reynoso-Meza et al. [111] proposed the spherical-pruning multi-objective optimization differential evolution (sp-MODE).
- The second objective was the sum of constraint violation for inequality constraints and the third objective was the sum of constraint violation for equality constraints.
- An external archive was used to store non-dominated solutions.
- The sphere-pruning operator aims to find the best trade-off between feasibility and the optimization of the objective function.
Reynoso-Meza et al. [111] proposed the spherical-pruning multi-objective optimization differential evolution (sp-MODE).

The second objective was the sum of constraint violation for inequality constraints and the third objective was the sum of constraint violation for equality constraints.

An external archive was used to store non-dominated solutions.

The sphere-pruning operator aims to find the best trade-off between feasibility and the optimization of the objective function.
Multi-objective concepts

Three-objective problem

- Reynoso-Meza et al. [111] proposed the spherical-pruning multi-objective optimization differential evolution (sp-MODE).
- The second objective was the sum of constraint violation for inequality constraints and the third objective was the sum of constraint violation for equality constraints.
- An external archive was used to store non-dominated solutions.
- The sphere-pruning operator aims to find the best trade-off between feasibility and the optimization of the objective function.
Multi-objective concepts

Three-objective problem

- Reynoso-Meza et al. [111] proposed the spherical-pruning multi-objective optimization differential evolution (sp-MODE).
- The second objective was the sum of constraint violation for inequality constraints and the third objective was the sum of constraint violation for equality constraints.
- An external archive was used to store non-dominated solutions.
- The sphere-pruning operator aims to find the best trade-off between feasibility and the optimization of the objective function.
Multi-objective concepts

Three-objective problem

- Zeng et al. [154] proposed converting a constrained problem into a dynamic constrained three-objective optimization problem.
  - The original objective.
  - The constraint-violation (decreasing).
  - A niche count (decreasing).

- Three evolutionary multi-objective optimization algorithms were tested in the approach.
Multi-objective concepts

Three-objective problem

- Zeng et al. [154] proposed converting a constrained problem into a dynamic constrained three-objective optimization problem.
  - The original objective.
  - The constraint-violation (decreasing).
  - A niche count (decreasing).
- Three evolutionary multi-objective optimization algorithms were tested in the approach.
Multi-objective concepts

Many-objective problem

- Gong and Cai [33] used Pareto dominance in the space defined by the constraints of a problem as a constraint-handling mechanism in a DE-based approach.
- An orthogonal process was employed for both, generating the initial population and for applying crossover.
- An external archive stored non-dominated solutions.
Multi-objective concepts

Many-objective problem

- Gong and Cai [33] used Pareto dominance in the space defined by the constraints of a problem as a constraint-handling mechanism in a DE-based approach.
- An orthogonal process was employed for both, generating the initial population and for applying crossover.
- An external archive stored non-dominated solutions.
Gong and Cai [33] used Pareto dominance in the space defined by the constraints of a problem as a constraint-handling mechanism in a DE-based approach.

An orthogonal process was employed for both, generating the initial population and for applying crossover.

An external archive stored non-dominated solutions.
Outline

1. Introduction
   - The problem of interest
   - Some important concepts
   - Mathematical-programming methods
   - Why alternative methods?

2. The early years
   - Penalty functions
   - Decoders
   - Special operators
   - Separation of objective function and constraints
   - General comments

3. Current constraint-handling techniques
   - Feasibility rules
   - Stochastic ranking
   - $\varepsilon$-constrained method
   - Novel penalty functions
   - Novel special operators
   - Multi-objective concepts
   - Ensemble of constraint-handling techniques

4. Summary and current trends
   - A bird’s eye view
   - Current trends
Mallipeddi and Suganthan [78] proposed an ensemble of four constraint techniques (ECHT):

- Feasibility rules.
- Stochastic ranking.
- A self-adaptive penalty function.
- The $\varepsilon$-constrained method.

A four sub-population scheme was considered.

One EP-based and one DE-based versions were designed.

Each constraint-handling technique was used to evolve an specific sub-population.

All sub-populations share all of their offspring.
Mallipeddi and Suganthan [78] proposed an ensemble of four constraint techniques (ECHT):

- Feasibility rules.
- Stochastic ranking.
- A self-adaptive penalty function.
- The $\varepsilon$-constrained method.

A four sub-population scheme was considered.

- One EP-based and one DE-based versions were designed.
- Each constraint-handling technique was used to evolve an specific sub-population.
- All sub-populations share all of their offspring,
Mallipeddi and Suganthan [78] proposed an ensemble of four constraint techniques (ECHT):

- Feasibility rules.
- Stochastic ranking.
- A self-adaptive penalty function.
- The $\varepsilon$-constrained method.

A four sub-population scheme was considered.

One EP-based and one DE-based versions were designed.

Each constraint-handling technique was used to evolve an specific sub-population.

All sub-populations share all of their offspring,
Mallipeddi and Suganthan [78] proposed an ensemble of four constraint techniques (ECHT):

- Feasibility rules.
- Stochastic ranking.
- A self-adaptive penalty function.
- The $\varepsilon$-constrained method.

A four sub-population scheme was considered.

One EP-based and one DE-based versions were designed.

Each constraint-handling technique was used to evolve an specific sub-population.

All sub-populations share all of their offspring,
Mallipeddi and Suganthan [78] proposed an ensemble of four constraint techniques (ECHT):

- Feasibility rules.
- Stochastic ranking.
- A self-adaptive penalty function.
- The $\varepsilon$-constrained method.

A four sub-population scheme was considered.
One EP-based and one DE-based versions were designed.
Each constraint-handling technique was used to evolve an specific sub-population.
All sub-populations share all of their offspring,
Elsayed et al. [29] proposed a DE-based algorithm where the combination of four DE-mutations, two DE recombinations and two constraint-handling techniques (feasibility rules and $\varepsilon$-constrained method) generated sixteen variants which were assigned to each individual in a single-population algorithm.

The rate of usage for each variant was based on its improvement measured by its ability to generate better solutions.

A local search algorithm was applied.
Elsayed et al. [29] proposed a DE-based algorithm where the combination of four DE-mutations, two DE recombinations and two constraint-handling techniques (feasibility rules and $\varepsilon$-constrained method) generated sixteen variants which were assigned to each individual in a single-population algorithm.

The rate of usage for each variant was based on its improvement measured by its ability to generate better solutions.

A local search algorithm was applied.
Elsayed et al. [29] proposed a DE-based algorithm where the combination of four DE-mutations, two DE recombinations and two constraint-handling techniques (feasibility rules and $\varepsilon$-constrained method) generated sixteen variants which were assigned to each individual in a single-population algorithm.

The rate of usage for each variant was based on its improvement measured by its ability to generate better solutions.

A local search algorithm was applied.
A similar idea was presented in a combination of two DE variants and a variable neighborhood search with three constraint-handling techniques (feasibility rules, $\varepsilon$-constrained method, and an adaptive penalty function) by Tasgetiren et al. [131].
Ensemble of constraint-handling techniques

- The ECHT opens a new paradigm in constraint-handling techniques.
- The design of mechanisms which allow the combination of approaches that can be seen as complementary (in terms of the way in which they operate).
- However, as the combination of several techniques considerably enhances the capabilities of an approach, it is also required to define parameter values for each of these techniques.
- Parameter control [74] becomes an important issue when designing ensemble approaches.
The ECHT opens a new paradigm in constraint-handling techniques.

The design of mechanisms which allow the combination of approaches that can be seen as complementary (in terms of the way in which they operate).

However, as the combination of several techniques considerably enhances the capabilities of an approach, it is also required to define parameter values for each of these techniques.

Parameter control [74] becomes an important issue when designing ensemble approaches.
Ensemble of constraint-handling techniques

- The ECHT opens a new paradigm in constraint-handling techniques.
- The design of mechanisms which allow the combination of approaches that can be seen as complementary (in terms of the way in which they operate).
- However, as the combination of several techniques considerably enhances the capabilities of an approach, it is also required to define parameter values for each of these techniques.
- Parameter control [74] becomes an important issue when designing ensemble approaches.
The ECHT opens a new paradigm in constraint-handling techniques.

The design of mechanisms which allow the combination of approaches that can be seen as complementary (in terms of the way in which they operate).

However, as the combination of several techniques considerably enhances the capabilities of an approach, it is also required to define parameter values for each of these techniques.

Parameter control [74] becomes an important issue when designing ensemble approaches.
## A bird’s eye view

<table>
<thead>
<tr>
<th>Technique</th>
<th>Core concept</th>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>FR</td>
<td>Three criteria for pairwise selection</td>
<td>Simple to add into a NIA&lt;br&gt;No extra parameters</td>
<td>May cause premature convergence</td>
</tr>
<tr>
<td>SR</td>
<td>Ranking process</td>
<td>Easy to implement</td>
<td>Not all NIAs have ordering in their processes&lt;br&gt;One extra parameter</td>
</tr>
<tr>
<td>$\varepsilon$-CM</td>
<td>Transforms a constrained problem into an unconstrained problem</td>
<td>Very competitive performance</td>
<td>Extra parameters&lt;br&gt;Local search for high performance</td>
</tr>
<tr>
<td>NPF</td>
<td>Focus on adaptive and dynamic approaches</td>
<td>Well-known transformation process</td>
<td>Some of them add extra parameters</td>
</tr>
<tr>
<td>NSO</td>
<td>Focus on boundary operators and equality constraints</td>
<td>Tendency to design easy to generalize operators</td>
<td>Still limited usage</td>
</tr>
<tr>
<td>MOC</td>
<td>Focused on bi-objective transformation of a CNOP&lt;br&gt;Pareto dominance</td>
<td>Both, Pareto ranking and dominance still popular</td>
<td>May require an additional constraint-handling technique</td>
</tr>
<tr>
<td>ECHT</td>
<td>Combination of two or more constraint-handling techniques</td>
<td>Very competitive performance</td>
<td>Requires the definition of several parameter values</td>
</tr>
</tbody>
</table>
How important is the search algorithm?

- DE is the most preferred algorithm, usually coupled with the feasibility rules.
- GAs are popular when coupled with penalty functions.
- PSO has been mainly coupled with the feasibility rules as well.
- ES has been usually coupled with the stochastic ranking.
- EP, ACO scarcely used.
- Among novel algorithms, ABC with feasibility rules has been particularly popular.
- AIS recently coupled with the feasibility rules.
- Gradient-based local search frequently found.
- Special operators focused on equality constraints.
- Multi-operator algorithms preferred over hybrid approaches.
How important is the search algorithm?

- DE is the most preferred algorithm, usually coupled with the feasibility rules.
- GAs are popular when coupled with penalty functions.
- PSO has been mainly coupled with the feasibility rules as well.
- ES has been usually coupled with the stochastic ranking.
- EP, ACO scarcely used.
- Among novel algorithms, ABC with feasibility rules has been particularly popular.
- AIS recently coupled with the feasibility rules.
- Gradient-based local search frequently found.
- Special operators focused on equality constraints.
- Multi-operator algorithms preferred over hybrid approaches.
DE is the most preferred algorithm, usually coupled with the feasibility rules.

GAs are popular when coupled with penalty functions.

PSO has been mainly coupled with the feasibility rules as well.

ES has been usually coupled with the stochastic ranking.

EP, ACO scarcely used.

Among novel algorithms, ABC with feasibility rules has been particularly popular.

AIS recently coupled with the feasibility rules.

Gradient-based local search frequently found.

Special operators focused on equality constraints.

Multi-operator algorithms preferred over hybrid approaches.
How important is the search algorithm?

- DE is the most preferred algorithm, usually coupled with the feasibility rules.
- GAs are popular when coupled with penalty functions.
- PSO has been mainly coupled with the feasibility rules as well.
- ES has been usually coupled with the stochastic ranking.
- EP, ACO scarcely used.
- Among novel algorithms, ABC with feasibility rules has been particularly popular.
- AIS recently coupled with the feasibility rules.
- Gradient-based local search frequently found.
- Special operators focused on equality constraints.
- Multi-operator algorithms preferred over hybrid approaches.
How important is the search algorithm?

- DE is the most preferred algorithm, usually coupled with the feasibility rules.
- GAs are popular when coupled with penalty functions.
- PSO has been mainly coupled with the feasibility rules as well.
- ES has been usually coupled with the stochastic ranking.
- EP, ACO scarcely used.
- Among novel algorithms, ABC with feasibility rules has been particularly popular.
- AIS recently coupled with the feasibility rules.
- Gradient-based local search frequently found.
- Special operators focused on equality constraints.
- Multi-operator algorithms preferred over hybrid approaches.
How important is the search algorithm?

- DE is the most preferred algorithm, usually coupled with the feasibility rules.
- GAs are popular when coupled with penalty functions.
- PSO has been mainly coupled with the feasibility rules as well.
- ES has been usually coupled with the stochastic ranking.
- EP, ACO scarcely used.
- Among novel algorithms, ABC with feasibility rules has been particularly popular.
- AIS recently coupled with the feasibility rules.
- Gradient-based local search frequently found.
- Special operators focused on equality constraints.
- Multi-operator algorithms preferred over hybrid approaches.
How important is the search algorithm?

- DE is the most preferred algorithm, usually coupled with the feasibility rules.
- GAs are popular when coupled with penalty functions.
- PSO has been mainly coupled with the feasibility rules as well.
- ES has been usually coupled with the stochastic ranking.
- EP, ACO scarcely used.
- Among novel algorithms, ABC with feasibility rules has been particularly popular.
- AIS recently coupled with the feasibility rules.
- Gradient-based local search frequently found.
- Special operators focused on equality constraints.
- Multi-operator algorithms preferred over hybrid approaches.
How important is the search algorithm?

- DE is the most preferred algorithm, usually coupled with the feasibility rules.
- GAs are popular when coupled with penalty functions.
- PSO has been mainly coupled with the feasibility rules as well.
- ES has been usually coupled with the stochastic ranking.
- EP, ACO scarcely used.
- Among novel algorithms, ABC with feasibility rules has been particularly popular.
- AIS recently coupled with the feasibility rules.
- Gradient-based local search frequently found.
- Special operators focused on equality constraints.
- Multi-operator algorithms preferred over hybrid approaches.
How important is the search algorithm?

- DE is the most preferred algorithm, usually coupled with the feasibility rules.
- GAs are popular when coupled with penalty functions.
- PSO has been mainly coupled with the feasibility rules as well.
- ES has been usually coupled with the stochastic ranking.
- EP, ACO scarcely used.
- Among novel algorithms, ABC with feasibility rules has been particularly popular.
- AIS recently coupled with the feasibility rules.
- Gradient-based local search frequently found.
- Special operators focused on equality constraints.
- Multi-operator algorithms preferred over hybrid approaches.
How important is the search algorithm?

- DE is the most preferred algorithm, usually coupled with the feasibility rules.
- GAs are popular when coupled with penalty functions.
- PSO has been mainly coupled with the feasibility rules as well.
- ES has been usually coupled with the stochastic ranking.
- EP, ACO scarcely used.
- Among novel algorithms, ABC with feasibility rules has been particularly popular.
- AIS recently coupled with the feasibility rules.
- Gradient-based local search frequently found.
- Special operators focused on equality constraints.
- Multi-operator algorithms preferred over hybrid approaches.
## Benchmarks

### The first one

<table>
<thead>
<tr>
<th>Function</th>
<th>n</th>
<th>Type of function</th>
<th>$\rho$</th>
<th>LI</th>
<th>NI</th>
<th>LE</th>
<th>NE</th>
<th>a</th>
</tr>
</thead>
<tbody>
<tr>
<td>g01</td>
<td>13</td>
<td>quadratic</td>
<td>0.0003%</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>g02</td>
<td>20</td>
<td>nonlinear</td>
<td>99.9973%</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>g03</td>
<td>10</td>
<td>nonlinear</td>
<td>0.0026%</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>g04</td>
<td>5</td>
<td>quadratic</td>
<td>27.0079%</td>
<td>4</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>g05</td>
<td>4</td>
<td>nonlinear</td>
<td>0.0000%</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>g06</td>
<td>2</td>
<td>nonlinear</td>
<td>0.0057%</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>g07</td>
<td>10</td>
<td>quadratic</td>
<td>0.0000%</td>
<td>3</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>g08</td>
<td>2</td>
<td>nonlinear</td>
<td>0.8581%</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>g09</td>
<td>7</td>
<td>nonlinear</td>
<td>0.5199%</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>g10</td>
<td>8</td>
<td>linear</td>
<td>0.0020%</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>g11</td>
<td>2</td>
<td>quadratic</td>
<td>0.0973%</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>g12</td>
<td>3</td>
<td>quadratic</td>
<td>4.7697%</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>g13</td>
<td>5</td>
<td>nonlinear</td>
<td>0.0000%</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>g14</td>
<td>10</td>
<td>nonlinear</td>
<td>0.0000%</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>g15</td>
<td>3</td>
<td>quadratic</td>
<td>0.0000%</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>g16</td>
<td>5</td>
<td>nonlinear</td>
<td>0.0204%</td>
<td>4</td>
<td>34</td>
<td>0</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>g17</td>
<td>6</td>
<td>nonlinear</td>
<td>0.0000%</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>g18</td>
<td>9</td>
<td>quadratic</td>
<td>0.0000%</td>
<td>0</td>
<td>13</td>
<td>0</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>g19</td>
<td>15</td>
<td>nonlinear</td>
<td>33.4761%</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>g20</td>
<td>24</td>
<td>linear</td>
<td>0.0000%</td>
<td>0</td>
<td>6</td>
<td>2</td>
<td>12</td>
<td>16</td>
</tr>
<tr>
<td>g21</td>
<td>7</td>
<td>linear</td>
<td>0.0000%</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>g22</td>
<td>22</td>
<td>linear</td>
<td>0.0000%</td>
<td>0</td>
<td>1</td>
<td>8</td>
<td>11</td>
<td>19</td>
</tr>
<tr>
<td>g23</td>
<td>9</td>
<td>linear</td>
<td>0.0000%</td>
<td>0</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>g24</td>
<td>2</td>
<td>linear</td>
<td>79.6556%</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>
### Benchmarks

#### The second one

<table>
<thead>
<tr>
<th>Problem/Search Range</th>
<th>Type of Objective</th>
<th>Number of Constraints</th>
<th>Feasibility Region (ρ)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>E</td>
<td>I</td>
</tr>
<tr>
<td>C01 [0, 0.10]^D</td>
<td>Non Separable</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>C02 [-5 12.5, 5 12]^D</td>
<td>Separable</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>C03 [-1000, 1000]^D</td>
<td>Non Separable</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>C04 [-50, 50]^D</td>
<td>Separable</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>C05 [-600, 600]^D</td>
<td>Separable</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>C06 [-600, 600]^D</td>
<td>Separable</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>
## Benchmarks

The second one

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Separability</th>
<th>Separable</th>
<th>Non Separable</th>
<th>Rotated</th>
<th>Value 1</th>
<th>Value 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>C07 [-140,140]</td>
<td>Non Separable</td>
<td>0</td>
<td>1 Separable</td>
<td></td>
<td>0.505123</td>
<td>0.503725</td>
</tr>
<tr>
<td>C08 [-140,140]</td>
<td>Non Separable</td>
<td>0</td>
<td>1 Rotated</td>
<td></td>
<td>0.379512</td>
<td>0.375278</td>
</tr>
<tr>
<td>C09 [-500,500]</td>
<td>Non Separable</td>
<td>1 Separable</td>
<td>0</td>
<td></td>
<td>0.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>C10 [-500,500]</td>
<td>Non Separable</td>
<td>1 Rotated</td>
<td>0</td>
<td></td>
<td>0.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>C11 [-100,100]</td>
<td>Rotated</td>
<td>1 Non Separable</td>
<td>0</td>
<td></td>
<td>0.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>C12 [-1000,1000]</td>
<td>Separable</td>
<td>1 Non Separable</td>
<td>1 Separable</td>
<td></td>
<td>0.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>C13 [-500,500]</td>
<td>Separable</td>
<td>0</td>
<td>2 Separable, 1 Non Separable</td>
<td>3</td>
<td>0.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>C14 [-1000,1000]</td>
<td>Non Separable</td>
<td>0</td>
<td>3 Separable</td>
<td></td>
<td>0.003112</td>
<td>0.006123</td>
</tr>
<tr>
<td>C15 [-1000,1000]</td>
<td>Non Separable</td>
<td>0</td>
<td>3 Rotated</td>
<td></td>
<td>0.003210</td>
<td>0.006023</td>
</tr>
<tr>
<td>C16 [-10,10]</td>
<td>Non Separable</td>
<td>2 Separable</td>
<td>1 Separable, 1 Non Separable</td>
<td>2</td>
<td>0.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>C17 [-10,10]</td>
<td>Non Separable</td>
<td>1 Separable</td>
<td>2 Non Separable</td>
<td></td>
<td>0.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>C18 [-50,50]</td>
<td>Non Separable</td>
<td>1 Separable</td>
<td>1 Separable</td>
<td></td>
<td>0.000010</td>
<td>0.000000</td>
</tr>
</tbody>
</table>
### Benchmarks

#### The most recent

<table>
<thead>
<tr>
<th>Problem/Search Range</th>
<th>Type of Objective</th>
<th>Number of Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>C01 [-100,100]</td>
<td>Non Separable</td>
<td>0</td>
</tr>
<tr>
<td>C02 [-100,100]</td>
<td>Non Separable, Rotated</td>
<td>0</td>
</tr>
<tr>
<td>C03 [-100,100]</td>
<td>Non Separable</td>
<td>1</td>
</tr>
<tr>
<td>C04 [-10,10]</td>
<td>Separable</td>
<td>0</td>
</tr>
<tr>
<td>C05 [-10,10]</td>
<td>Non Separable</td>
<td>0</td>
</tr>
<tr>
<td>C06 [-20,20]</td>
<td>Separable</td>
<td>6</td>
</tr>
<tr>
<td>C07 [-50,50]</td>
<td>Separable</td>
<td>2</td>
</tr>
<tr>
<td>C08 [-100,100]</td>
<td>Separable</td>
<td>2</td>
</tr>
<tr>
<td>C09 [-10,10]</td>
<td>Separable</td>
<td>0</td>
</tr>
<tr>
<td>C10 [-100,100]</td>
<td>Separable</td>
<td>2</td>
</tr>
<tr>
<td>C11 [-100,100]</td>
<td>Separable</td>
<td>2</td>
</tr>
<tr>
<td>C12 [-100,100]</td>
<td>Separable</td>
<td>0</td>
</tr>
<tr>
<td>C13 [-100,100]</td>
<td>Non Separable</td>
<td>0</td>
</tr>
<tr>
<td>C14 [-100,100]</td>
<td>Non Separable</td>
<td>1</td>
</tr>
<tr>
<td>C15 [-100,100]</td>
<td>Separable</td>
<td>1</td>
</tr>
<tr>
<td>C16 [-100,100]</td>
<td>Separable</td>
<td>1</td>
</tr>
<tr>
<td>C17 [-100,100]</td>
<td>Non Separable</td>
<td>1</td>
</tr>
<tr>
<td>C18 Separable</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>
## Benchmarks

### The most recent

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Domain</th>
<th>Separability</th>
<th>Separability Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>C19</td>
<td>[-50,50]$^D$</td>
<td>Separable</td>
<td>0</td>
</tr>
<tr>
<td>C20</td>
<td>[-100,100]$^D$</td>
<td>Non Separable</td>
<td>0</td>
</tr>
<tr>
<td>C21</td>
<td>[-100,100]$^D$</td>
<td>Rotated</td>
<td>3</td>
</tr>
<tr>
<td>C22</td>
<td>[-100,100]$^D$</td>
<td>Rotated</td>
<td>2</td>
</tr>
<tr>
<td>C23</td>
<td>[-100,100]$^D$</td>
<td>Rotated</td>
<td>1</td>
</tr>
<tr>
<td>C24</td>
<td>[-100,100]$^D$</td>
<td>Rotated</td>
<td>1</td>
</tr>
<tr>
<td>C25</td>
<td>[-100,100]$^D$</td>
<td>Rotated</td>
<td>1</td>
</tr>
<tr>
<td>C26</td>
<td>[-100,100]$^D$</td>
<td>Rotated</td>
<td>2</td>
</tr>
<tr>
<td>C27</td>
<td>[-100,100]$^D$</td>
<td>Rotated</td>
<td>2</td>
</tr>
<tr>
<td>C28</td>
<td>[-50,50]$^D$</td>
<td>Rotated</td>
<td>0</td>
</tr>
</tbody>
</table>
Performance measures

- **Evals** (number of solution evaluations to find a feasible solution).
- Progress ratio (difference between the objective function value of the first and best feasible solutions found).
- AFES (average number of solution evaluations in a set of successful runs).
- **FP** (percentage of feasible runs).
- **P** (percentage of successful runs).
- **SP** (successful performance computed by AFES divided by P).
Performance measures

- **Evals**: (number of solution evaluations to find a feasible solution).
- **Progress ratio**: (difference between the objective function value of the first and best feasible solutions found).
- **AFES**: (average number of solution evaluations in a set of successful runs).
- **FP**: (percentage of feasible runs).
- **P**: (percentage of successful runs).
- **SP**: (successful performance computed by AFES divided by P).
Performance measures

- Evals (number of solution evaluations to find a feasible solution).
- Progress ratio (difference between the objective function value of the first and best feasible solutions found).
- AFES (average number of solution evaluations in a set of successful runs).

  - FP (percentage of feasible runs).
  - P (percentage of successful runs).
  - SP (successful performance computed by AFES divided by P).
Performance measures

- **Evals**: (number of solution evaluations to find a feasible solution).
- **Progress ratio**: (difference between the objective function value of the first and best feasible solutions found).
- **AFES**: (average number of solution evaluations in a set of successful runs).
- **FP**: (percentage of feasible runs).
  - **P**: (percentage of successful runs).
  - **SP**: (successful performance computed by AFES divided by P).
Performance measures

- Evals (number of solution evaluations to find a feasible solution).
- Progress ratio (difference between the objective function value of the first and best feasible solutions found).
- AFES (average number of solution evaluations in a set of successful runs).
- FP (percentage of feasible runs).
- P (percentage of successful runs).
- SP (successful performance computed by AFES divided by P).
Performance measures

- Evals (number of solution evaluations to find a feasible solution).
- Progress ratio (difference between the objective function value of the first and best feasible solutions found).
- AFES (average number of solution evaluations in a set of successful runs).
- FP (percentage of feasible runs).
- P (percentage of successful runs).
- SP (successful performance computed by AFES divided by P).
Outline

1. Introduction
   - The problem of interest
   - Some important concepts
   - Mathematical-programming methods
   - Why alternative methods?

2. The early years
   - Penalty functions
   - Decoders
   - Special operators
   - Separation of objective function and constraints
   - General comments

3. Current constraint-handling techniques
   - Feasibility rules
   - Stochastic ranking
   - $\varepsilon$-constrained method
   - Novel penalty functions
   - Novel special operators
   - Multi-objective concepts
   - Ensemble of constraint-handling techniques

4. Summary and current trends
   - A bird’s eye view
   - Current trends
Current trends

Constraint-handling for EMO

- EMO approaches usually adopt constraint-handling techniques for single-objective optimization.

- Topics of interest:
  - Performance measures.
  - Diversity mechanisms.
  - Boundary operators.
  - Many-objective multi-constrained optimization.
Current trends

Constraint-handling for EMO

- EMO approaches usually adopt constraint-handling techniques for single-objective optimization.
- Topics of interest:
  - Performance measures.
  - Diversity mechanisms.
  - Boundary operators.
  - Many-objective multi-constrained optimization.
Current trends

Constraint approximation

- Fitness approximation methods have been extensively applied to unconstrained optimization problems.
- Jin [47] proposed to enlarge the feasible region by using surrogates to ease the generation of feasible solutions.
- Regis [110] used radial basis functions as surrogates to approximate constraints and objective functions in constrained multi-objective optimization problems.
Current trends

Constraint approximation

- Fitness approximation methods have been extensively applied to unconstrained optimization problems.
- Jin [47] proposed to enlarge the feasible region by using surrogates to ease the generation of feasible solutions.
- Regis [110] used radial basis functions as surrogates to approximate constraints and objective functions in constrained multi-objective optimization problems.
Current trends

Constraint approximation

- Fitness approximation methods have been extensively applied to unconstrained optimization problems.
- Jin [47] proposed to enlarge the feasible region by using surrogates to ease the generation of feasible solutions.
- Regis [110] used radial basis functions as surrogates to approximate constraints and objective functions in constrained multi-objective optimization problems.
Current trends

Constraint approximation

- Datta and Regis [24] proposed an evolution strategy coupled with cubic radial basis functions to solve constrained multi-objective optimization problems.

- Miranda-Varela and Mezura-Montes [97] added feasibility information in the evolution control of a surrogate-assisted differential evolution to solve constrained optimization problems.
Current trends

Constraint approximation

- Datta and Regis [24] proposed an evolution strategy coupled with cubic radial basis functions to solve constrained multi-objective optimization problems.

- Miranda-Varela and Mezura-Montes [97] added feasibility information in the evolution control of a surrogate-assisted differential evolution to solve constrained optimization problems.
Dynamic constraints

- There is a considerable amount of research devoted to deal with unconstrained dynamic optimization problems.
- Initial efforts have focused on constrained dynamic optimization problems.
Current trends

Dynamic constraints

- There is a considerable amount of research devoted to deal with unconstrained dynamic optimization problems.
- Initial efforts have focused on constrained dynamic optimization problems.
Current trends

Dynamic constraints

Nguyen and Yao [99] started the research on DCOPs, by providing a benchmark and an initial comparison of algorithms based mainly on hypermutation and repair methods.
Current trends

Benchmark

Table 1: Main features of the test problems (Nguyen and Yao, 2012).

<table>
<thead>
<tr>
<th>Problem</th>
<th>Obj. Function</th>
<th>Constraints</th>
<th>DFR</th>
<th>SwO</th>
<th>bNAO</th>
<th>OICB</th>
<th>OISB</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>g24.u</td>
<td>Dynamic</td>
<td>No Constraints</td>
<td>1</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>g24.1</td>
<td>Dynamic</td>
<td>Static</td>
<td>2</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>N/A</td>
</tr>
<tr>
<td>g24.f</td>
<td>Static</td>
<td>Static</td>
<td>2</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>N/A</td>
</tr>
<tr>
<td>g24.uf</td>
<td>Static</td>
<td>No Constraints</td>
<td>1</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>N/A</td>
</tr>
<tr>
<td>g24.2</td>
<td>Dynamic</td>
<td>Static</td>
<td>2</td>
<td>Yes</td>
<td>No</td>
<td>Yes and No</td>
<td>Yes and No</td>
<td>N/A</td>
</tr>
<tr>
<td>g24.2u</td>
<td>Dynamic</td>
<td>No Constraints</td>
<td>1</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>N/A</td>
</tr>
<tr>
<td>g24.3</td>
<td>Static</td>
<td>Dynamic</td>
<td>2-3</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>N/A</td>
</tr>
<tr>
<td>g24.3b</td>
<td>Dynamic</td>
<td>Dynamic</td>
<td>2-3</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>N/A</td>
</tr>
<tr>
<td>g24.3f</td>
<td>Static</td>
<td>Static</td>
<td>1</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>N/A</td>
</tr>
<tr>
<td>g24.4</td>
<td>Dynamic</td>
<td>Dynamic</td>
<td>2-3</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>N/A</td>
</tr>
<tr>
<td>g24.5</td>
<td>Dynamic</td>
<td>Dynamic</td>
<td>2-3</td>
<td>Yes</td>
<td>No</td>
<td>Yes and No</td>
<td>Yes and No</td>
<td>N/A</td>
</tr>
<tr>
<td>g24.6a</td>
<td>Dynamic</td>
<td>Static</td>
<td>2</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Hard</td>
</tr>
<tr>
<td>g24.6b</td>
<td>Dynamic</td>
<td>Static</td>
<td>1</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>g24.6c</td>
<td>Dynamic</td>
<td>Static</td>
<td>2</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Easy</td>
</tr>
<tr>
<td>g24.6d</td>
<td>Dynamic</td>
<td>Static</td>
<td>2</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Hard</td>
</tr>
<tr>
<td>g24.7</td>
<td>Static</td>
<td>Dynamic</td>
<td>2</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>N/A</td>
</tr>
<tr>
<td>g24.8a</td>
<td>Dynamic</td>
<td>No Constraints</td>
<td>1</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>N/A</td>
</tr>
<tr>
<td>g24.8b</td>
<td>Dynamic</td>
<td>Static</td>
<td>2</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>N/A</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DFR</th>
<th>Number of disconnected feasible regions</th>
</tr>
</thead>
<tbody>
<tr>
<td>SwO</td>
<td>Switched global optimum between disconnected regions</td>
</tr>
<tr>
<td>bNAO</td>
<td>Better newly appear optimum without changing existing ones</td>
</tr>
<tr>
<td>OICB</td>
<td>Global optimum is in the constraint boundary</td>
</tr>
<tr>
<td>OISB</td>
<td>Global optimum is in the search boundary</td>
</tr>
<tr>
<td>Path</td>
<td>Indicate if it is easy or difficult to use mutation to travel between feasible regions</td>
</tr>
<tr>
<td>Dynamic</td>
<td>The function is dynamic</td>
</tr>
<tr>
<td>Static</td>
<td>There is no change</td>
</tr>
</tbody>
</table>

* In some change periods, the landscape either is a plateau or contains infinite number of optima and all optima (including the existing optimum) lie in a line parallel to one of the axes.
Pal et al. [101] proposed one of the first competitive algorithms for DCOPs based on the gravitational search algorithm and a repair method.

Ameca-Alducin et al. [3] proposed a DE-based approach with a repair mechanism based on sampling to solve DCOPs. Immigrants and change of DE variants were used as well.
Current trends

Dynamic constraints

- Pal et al. [101] proposed one of the first competitive algorithms for DCOPs based on the gravitational search algorithm and a repair method.

- Ameca-Alducin et al. [3] proposed a DE-based approach with a repair mechanism based on sampling to solve DCOPs. Immigrants and change of DE variants were used as well.
Current trends

Dynamic constraints

- Sharma & Sharma [119] used special operators and Tabu search concepts to deal with DCOPs.
- Aragón et al. [5] proposed a T-cell-inspired approach to solve DCOPs where four sub-populations with different goals interacted in the dynamic search space.
- Bu et al. [15] proposed two new benchmarks and a dynamic species-based PSO with an ensemble of tracking feasible regions strategies.
Current trends

Dynamic constraints

- Sharma & Sharma [119] used special operators and Tabu search concepts to deal with DCOPs.

- Aragón et al. [5] proposed a T-cell-inspired approach to solve DCOPs where four sub-populations with different goals interacted in the dynamic search space.

- Bu et al. [15] proposed two new benchmarks and a dynamic species-based PSO with an ensemble of tracking feasible regions strategies.
Current trends

Dynamic constraints

- Sharma & Sharma [119] used special operators and Tabu search concepts to deal with DCOPs.
- Aragón et al. [5] proposed a T-cell-inspired approach to solve DCOPs where four sub-populations with different goals interacted in the dynamic search space.
- Bu et al. [15] proposed two new benchmarks and a dynamic species-based PSO with an ensemble of tracking feasible regions strategies.
Ensembles/multi-operator NIAs

- This topic is still in its starting phase.
- More combinations and adaptive mechanisms within the ensembles of constraint-handling techniques and/or multi-operator NIAs are expected.
Current trends

Ensembles/multi-operator NIAs

- This topic is still in its starting phase.
- More combinations and adaptive mechanisms within the ensembles of constraint-handling techniques and/or multi-operator NIAs are expected.
Current trends

**Theory**

- There is some work on runtime analysis in constrained search spaces with EAs [158] and also in the usefulness of infeasible solutions in the search process [153].
- Other theoretical studies have focused on some ES variants, such as the (1+1)-ES [10] and more recently the (1,λ)-ES [9].
- More research in this area is required.
Current trends

Theory

- There is some work on runtime analysis in constrained search spaces with EAs [158] and also in the usefulness of infeasible solutions in the search process [153].

- Other theoretical studies have focused on some ES variants, such as the (1+1)-ES [10] and more recently the (1,λ)-ES [9].

- More research in this area is required.
Current trends

Theory

- There is some work on runtime analysis in constrained search spaces with EAs [158] and also in the usefulness of infeasible solutions in the search process [153].
- Other theoretical studies have focused on some ES variants, such as the (1+1)-ES [10] and more recently the (1,\(\lambda\))-ES [9].
- More research in this area is required.
References

A Socio-Behavioural Simulation Model for Engineering Design Optimization.

M. M. Ali and Z. Kajee-Bagdadi.
A local exploration-based differential evolution algorithm for constrained global optimization.

M.-Y. Ameca-Alducin, E. Mezura-Montes, and N. Cruz-Ramírez.
Differential evolution with combined variants and a repair method to solve dynamic constrained optimization problems.

A. Angantyr, J. Andersson, and J.-O. Aidanpaa.
Constrained Optimization based on a Multiobjective Evolutionary Algorithms.

V. Aragón, S. Esquivel, and C. Coello-Coello.
Artificial immune system for solving dynamic constrained optimization problems.

Artificial immune system for solving constrained optimization problems.
References II

A novel model of artificial immune system for solving constrained optimization problems with dynamic tolerance factor.

Constrained Optimization Based on Quadratic Approximations in Genetic Algorithms.

D. Arnold.
On the behaviour of the (1,λ)-ES for a simple constrained problem.

D. V. Arnold and D. Brauer.
On the Behaviour of the (1+1)-ES for a Simple Constrained Problem.

H. J. Barbosa and A. C. Lemonge.
An adaptive penalty scheme in genetic algorithms for constrained optimization problems.
Motivation

The idea of handling constraints in evolutionary optimization algorithms has been a topic of great interest to researchers in the field of computational intelligence. Constraints in optimization problems often represent real-world limitations or conditions that must be satisfied. Handling these constraints effectively is crucial for obtaining solutions that are both feasible and optimal. This section focuses on various approaches and techniques that have been developed to address constrained optimization problems using evolutionary algorithms.


damaged image

References

J. Brest.
Constrained real-parameter optimization with \( \epsilon \)-self-adaptive differential evolution.

J. Brest, B. Bošković, and V. Žumer.

J. Brest, V. Zumer, and M. S. Maucec.

C. Bu, W. Luo, and L. Yue.
Continuous dynamic constrained optimization with ensemble of locating and tracking feasible regions strategies.

L. Cagnina, S. Esquivel, and C. Coello-Coello.
A Bi-population PSO with a Shake-Mechanism for Solving Constrained Numerical Optimization.

A Particle Swarm Optimizer for Constrained Numerical Optimization.
References IV

C. A. C. Coello.

C. A. Coello Coello.
Treating Constraints as Objectives for Single-Objective Evolutionary Optimization. 

C. A. Coello Coello.

W. A. Crossley and E. A. Williams. 
A Study of Adaptive Penalty Functions for Constrained Genetic Algorithm Based Optimization. 

N. Cruz-Cortés, D. Trejo-Pérez, and C. A. C. Coello.
Handling Constraints in Global Optimization using an Artificial Immune System. 
Lecture Notes in Computer Science Vol. 3627.

R. Datta and K. Deb.
A bi-objective based hybrid evolutionary-classical algorithm for handling equality constraints. 
LNCS Vol. 6576.
R. Datta and R. G. Regis.
A surrogate-assisted evolution strategy for constrained multi-objective optimization.

K. Deb.
An Efficient Constraint Handling Method for Genetic Algorithms.

K. Deb and R. Datta.
A Fast and Accurate Solution of Constrained Optimization Problems Using a Hybrid Bi-Objective and Penalty Function Approach.

S. Elsayed, R. Sarker, and D. Essam.
A comparative study of different variants of genetic algorithms for constrained optimization.

S. Elsayed, R. Sarker, and D. Essam.
Ga with a new multi-parent crossover for constrained optimization.

S. Elsayed, R. Sarker, and D. Essam.
Integrated strategies differential evolution algorithm with a local search for constrained optimization.
Improved differential evolution based on stochastic ranking for robust layout synthesis of mems components.

R. Farmani and J. A. Wright.
Self-Adaptive Fitness Formulation for Constrained Optimization.

L. Fonseca, P. Capriles, H. Barbosa, and A. Lemonge.
A stochastic rank-based ant system for discrete structural optimization.

W. Gong and Z. Cai.
A Multiobjective Differential Evolution Algorithm for Constrained Optimization.

A. B. Hadj-Alouane and J. C. Bean.
A Genetic Algorithm for the Multiple-Choice Integer Program.

H. Hamda and M. Schoenauer.
Adaptive techniques for Evolutionary Topological Optimum Design.
S. B. Hamida and M. Schoenauer.
ASCHEA: New Results Using Adaptive Segregational Constraint Handling.

N. Hamza, S. Elsayed, D. Essam, and R. Sarker.
Differential evolution combined with constraint consensus for constrained optimization.

Q. He and L. Wang.
A hybrid particle swarm optimization with a feasibility-based rule for constrained optimization.

Q. He, L. Wang, and F.-Z. Huang.
Nonlinear Constrained Optimization by Enhanced Co-evolutionary PSO.

Handling Constraints using Multiobjective Optimization Concepts.

R. Hinterding and Z. Michalewicz.
F. Hoffmeister and J. Sprave.
Problem-independent handling of constraints by use of metric penalty functions.

Constrained Optimization via Genetic Algorithms.

G. Huan-Tong, S. Qing-Xi, J. Feng, and S. Yi-Jie.
An evolution strategy with stochastic ranking for solving reactive power optimization.

F.-Z. Huang, L. Wang, and Q. He.
A Hybrid Differential Evolution with Double Populations for Constrained Optimization.

V. L. Huang, A. K. Qin, and P. N. Suganthan.

Y. Jin.
Surrogate-assisted evolutionary computation: Recent advances and future challenges.
J. Joines and C. Houck.
On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GAs.

D. Karaboga and B. Akay.
A modified artificial bee colony (abc) algorithm for constrained optimization problems.

D. Karaboga and B. Basturk.
Artificial bee colony (abc) optimization algorithm for solving constrained optimization problems.

S. Kazarlis and V. Petridis.
Varying Fitness Functions in Genetic Algorithms: Studying the Rate of Increase of the Dynamic Penalty Terms.
Lecture Notes in Computer Science Vol. 1498.

D. G. Kim and P. Husbands.
D. G. Kim and P. Husbands.
Landscape Changes and the Performance of Mapping Based Constraint Handling Methods.
Lecture Notes in Computer Science Vol. 1498.

D. G. Kim and P. Husbands.
Mapping Based Constraint Handling for Evolutionary Search; Thurston’s Circle Packing and Grid Generation.

R. Kowalczyk.
Constraint Consistent Genetic Algorithms.

S. Koziel and Z. Michalewicz.
A Decoder-based Evolutionary Algorithm for Constrained Parameter Optimization Problems.
Lecture Notes in Computer Science Vol. 1498.

S. Koziel and Z. Michalewicz.
Evolutionary Algorithms, Homomorphous Mappings, and Constrained Parameter Optimization.
S. Kukkonen and J. Lampinen.
Constrained Real-Parameter Optimization with Generalized Differential Evolution.

A. Kuri-Morales and C. V. Quezada.
A Universal Eclectic Genetic Algorithm for Constrained Optimization.

J. Lampinen.

R. Landa Becerra and C. A. Coello Coello.
Cultured differential evolution for constrained optimization.

G. Leguizamón and C. A. C. Coello.
Boundary Search for Constrained Numerical Optimization Problems with an Algorithm Inspired on the Ant Colony Metaphor.

G. Leguizamón and C. Coello-Coello.
A Boundary Search based ACO Algorithm Coupled with Stochastic Ranking.
L. D. Li, X. Li, and X. Yu.

S. Li and Y. Li.
A new self-adaption differential evolution algorithm based component model.

Z. Li, J. Liang, X. He, and Z. Shang.
Differential evolution with dynamic constraint-handling mechanism.

J. J. Liang and P. N. Suganthan.
Dynamic Multi-Swarm Particle Swarm Optimizer with a Novel Constrain-Handling Mechanism.

Coevolutionary comprehensive learning particle swarm optimizer.

B. Liu, H. Ma, X. Zhang, B. Liu, and Y. Zhou.
A memetic co-evolutionary differential evolution algorithm for constrained optimization.
Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization.  

SRaDE: An Adaptive Differential Evolution Based on Stochastic Ranking.  

An organizational evolutionary algorithm for numerical optimization.  

Stochastic ranking based differential evolution algorithm for constrained optimization problem.  

F. G. Lobo, C. F. Lima, and Z. Michalewicz, editors.  
*Parameter Setting in Evolutionary Algorithms*.  

H. Lu and W. Chen.  
Self-adaptive velocity particle swarm optimization for solving constrained optimization problems.  
References XIV

H. Ma and D. Simon.
Blended biogeography-based optimization for constrained optimization.  

R. Mallipeddi, P. Suganthan, and B. Qu.
Diversity enhanced adaptive evolutionary programming for solving single objective constrained problems.  

R. Mallipeddi and P. N. Suganthan.
Ensemble of Constraint Handling Techniques.  

A. Mani and C. Patvardhan.
A novel hybrid constraint-handling technique for evolutionary optimization.  

S. M. Elsayed, R. A. Sarker, and D. L. Essam.
Multi-operator based evolutionary algorithms for solving constrained optimization problems.  

A. Menchaca-Méndez and C. A. Coello Coello.
A new proposal to hybridize the nelder-mead method to a differential evolution algorithm for constrained optimization.  
E. Mezura-Montes and O. Cetina-Domínguez.
Exploring promising regions of the search space with the scout bee in the artificial bee colony for constrained optimization.

E. Mezura-Montes and C. A. Coello Coello.
A Simple Multimembered Evolution Strategy to Solve Constrained Optimization Problems.

E. Mezura-Montes and C. A. Coello-Coello.
Constrained optimization via multiobjective evolutionary algorithms.


Smart flight and dynamic tolerances in the artificial bee colony for constrained optimization.

E. Mezura-Montes and J. I. Flores-Mendoza.
Improved particle swarm optimization in constrained numerical search spaces.
E. Mezura-Montes and B. Hernández-Ocaña.
Modified bacterial foraging optimization for engineering design.

E. Mezura-Montes and A. G. Palomeque-Ortiz.
Parameter Control in Differential Evolution for Constrained Optimization.

E. Mezura-Montes, J. Velázquez-Reyes, and C. A. C. Coello.
Promising Infeasibility and Multiple Offspring Incorporated to Differential Evolution for Constrained Optimization.

E. Mezura-Montes, J. Velázquez-Reyes, and C. A. C. Coello.
Modified Differential Evolution for Constrained Optimization.

E. Mezura-Montes and R. E. Velez-Koeppel.
Elitist artificial bee colony for constrained real-parameter optimization.
References XVII

Z. Michalewicz.
*Genetic Algorithms + Data Structures = Evolution Programs.*

Z. Michalewicz and N. F. Attia.
Evolutionary Optimization of Constrained Problems.

Z. Michalewicz and G. Nazhiyath.
Genocop III: A co-evolutionary algorithm for numerical optimization with nonlinear constraints.

Z. Michalewicz and M. Schoenauer.

M.-E. Miranda-Varela and E. Mezura-Montes.
Surrogate-assisted differential evolution with an adaptive evolution control based on feasibility to solve constrained optimization problems.

PESO+ for Constrained Optimization.
L.-T. Nguyen and X. Yao.
Continuous dynamic constrained optimization the challenges.


Dynamic constrained optimization with offspring repair based gravitational search algorithm.

Projection-based local search operator for multiple equality constraints within genetic algorithms.

D. Powell and M. M. Skolnick.
Using genetic algorithms in engineering design optimization with non-linear constraints.

S. Puzzi and A. Carpinteri.
A double-multiplicative dynamic penalty approach for constrained evolutionary optimization.
K. Rasheed.
An Adaptive Penalty Approach for Constrained Genetic-Algorithm Optimization.

An Evolutionary Algorithm for Constrained Optimization.

T. Ray and K. Liew.
A Swarm with an Effective Information Sharing Mechanism for Unconstrained and Constrained Single Objective Optimization Problems.

T. Ray and K. Liew.
Society and Civilization: An Optimization Algorithm Based on the Simulation of Social Behavior.

Infeasibility driven evolutionary algorithm for constrained optimization.
R. G. Regis.
Multi-objective constrained black-box optimization using radial basis function surrogates.

Multiobjective optimization algorithm for solving constrained single objective problems.

A Segregated Genetic Algorithm for Constrained Structural Optimization.

T. P. Runarsson.
Constrained Evolutionary Optimization by Approximate Ranking and Surrogate Models.

T. P. Runarsson and X. Yao.
Stochastic Ranking for Constrained Evolutionary Optimization.

T. P. Runarsson and X. Yao.
Search biases in constrained evolutionary optimization.
M. Schoenauer and Z. Michalewicz.
Evolutionary Computation at the Edge of Feasibility.

M. Schoenauer and Z. Michalewicz.
Boundary Operators for Constrained Parameter Optimization Problems.

M. Schoenauer and S. Xanthakis.
Constrained GA Optimization.

A. Sharma and D. Sharma.
A constraint guided search for improving evolutionary algorithms.

Performance of infeasibility empowered memetic algorithm for cec 2010 constrained optimization problems.
M. Spadoni and L. Stefanini.
Handling box, linear and quadratic-convex constraints for boundary optimization with differential evolution algorithms.

J. Sun and J. M. Garibaldi.
A Novel Memetic Algorithm for Constrained Optimization.

T. Takahama and S. Sakai.
Constrained Optimization by $\alpha$ Constrained Genetic Algorithm ($\alpha$GA).

T. Takahama and S. Sakai.
Constrained Optimization by Applying the $\alpha$ Constrained Method to the Nonlinear Simplex Method with Mutations.

T. Takahama and S. Sakai.
Constrained optimization by $\varepsilon$ constrained particle swarm optimizer with $\varepsilon$-level control.

T. Takahama and S. Sakai.
Constrained Optimization by the $\varepsilon$ Constrained Differential Evolution with Gradient-Based Mutation and Feasible Elites.
T. Takahama and S. Sakai.
Constrained optimization by $\varepsilon$-constrained differential evolution with dynamic $\varepsilon$-level control.

T. Takahama and S. Sakai.
Solving difficult constrained optimization problems by the $\varepsilon$-constrained differential evolution with gradient-based mutation.

T. Takahama and S. Sakai.
Constrained optimization by the $\varepsilon$-constrained differential evolution with an archive and gradient-based mutation.

T. Takahama, S. Sakai, and N. Iwane.
Constrained optimization by the epsilon constrained hybrid algorithm of particle swarm optimization and genetic algorithm.
Lecture Notes in Artificial Intelligence Vol. 3809.

M. Tasgetiren, P. Suganthan, Q. Pan, R. Mallipeddi, and S. Sarman.
An ensemble of differential evolution algorithms for constrained function optimization.

M. F. Tasgetiren and P. N. Suganthan.
A Multi-Populated Differential Evolution Algorithm for Solving Constrained Optimization Problem.
An adaptive penalty formulation for constrained evolutionary optimization.

A. S. B. Ullah, R. Sarker, and C. Lokan.
An agent-based memetic algorithm (AMA) for nonlinear optimization with equality constraints.

A. S. S. M. B. Ullah, R. Sarker, and D. Cornforth.
Search Space Reduction Technique for Constrained Optimization with Tiny Feasible Space.

An Agent-based Memetic Algorithm (AMA) for Solving Constrained Optimization Problems.

AMA: a new approach for solving constrained real-valued optimization problems.

S. Venkatraman and G. G. Yen.
G. Venter and R. T. Haftka.
Constrained particle search optimization using a bi-objective formulation.

L. Wang and L. po Li.
An effective differential evolution with level comparison for constrained engineering design.

An Adaptive Bacterial Foraging Algorithm For Constrained Optimization.

Y. Wang and Z. Cai.
A hybrid multi-swarm particle swarm optimization to solve constrained optimization problems.

Y. Wang and Z. Cai.
Hybrid differential evolution and adaptive trade-off model to solve constrained optimization problems.

Y. Wang, Z. Cai, G. Guo, and Y. Zhou.
Multiobjective optimization and hybrid evolutionary algorithm to solve constrained optimization problems.

Y. Wang, Z. Cai, and Y. Zhou.
Accelerating adaptive trade-off model using shrinking space technique for constrained evolutionary optimization.
Y. Wang, Z. Cai, Y. Zhou, and Z. Fan.

Y. Wang, Z. Cai, Y. Zhou, and W. Zeng.

Y. Wang, H. Liu, Z. Cai, and Y. Zhou.


B. Wu and X. Yu.

J.-Y. Wu.
Y. Wu, Y. Li, and X. Xu.
A Novel Component-Based Model and Ranking Strategy in Constrained Evolutionary Optimization.

On the Usefulness of Infeasible Solutions in Evolutionary Search: A Theoretical Study.

S. Zeng, R. Jiao, C. Li, X. Li, and J. S. Alkasassbeh.
A general framework of dynamic constrained multiobjective evolutionary algorithms for constrained optimization.

A Lower-dimensional-Search Evolutionary Algorithm and Its Application in Constrained Optimization Problem.

Differential evolution with dynamic stochastic selection for constrained optimization.

Constrained Optimization by the Evolutionary Algorithm with Lower Dimensional Crossover and Gradient-Based Mutation.
Y. Zhou and J. He.
A runtime analysis of evolutionary algorithms for constrained optimization problems. 

K. Zielinski and R. Laur.

K. Zielinski and R. Laur.
Stopping criteria for differential evolution in constrained single-objective optimization. 

K. Zielinski, S. P. Vudathu, and R. Laur.
Influence of Different Deviations Allowed for Equality Constraints on Particle Swarm Optimization and Differential Evolution. 

K. Zielinski, X. Wang, and R. Laur.
Comparison of Adaptive Approaches for Differential Evolution. 