# Adaptive Penalty and Barrier function based on Fuzzy Logic

João Matias, Aldina Correia, Pedro Mestre, Carlos Serodio, Pedro Couto, Christophe Teixeira, Pedro Melo-Pinto

## Abstract

Optimization methods have been used in many areas of knowledge, such as Engineering, Statistics, Chemistry, among others, to solve optimization problems. In many cases it is not possible to use derivative methods, due to the characteristics of the problem to be solved and/or its constraints, for example if the involved functions are non-smooth and/or their derivatives are not know. To solve this type of problems a Java based API has been implemented, which includes only derivative-free optimization methods, and that can be used to solve both constrained and unconstrained problems. For solving constrained problems, the classic Penalty and Barrier functions were included in the API. In this paper a new approach to Penalty and Barrier functions, based on Fuzzy Logic, is proposed. Two penalty functions, that impose a progressive penalization to solutions that violate the constraints, are discussed. The implemented functions impose a low penalization when the violation of the constraints is low and a heavy penalty when the violation is high. Numerical results, obtained using twenty-eight test problems, comparing the proposed Fuzzy Logic based functions to six of the classic Penalty and Barrier functions are presented.
Considering the achieved results, it can be concluded that the proposed penalty functions besides being very robust also have a very good performance.

## Keywords

Applications; Fuzzy mathematical programming; Mathematics; Derivative free optimization; Direct search methods; Penalty and Barrier functions; Fuzzy Logic

## 1. Introduction

Optimization, also known as Mathematical Programming, is used in many decision making processes. In these processes the main objective is to determine the best use of available resources in order to obtain the best results for a given reality. So, optimization has been used in many scientific areas such as Engineering, Statistics and Chemistry, among others.

Problems are defined by models consisting of one or more functions (which need to be minimized or maximized), called objective function, and at least one variable, the decision variable(s). An unconstrained optimization problem can therefore be defined as in (1).

$$\min_{x \in \mathbb{R}^n} \quad f(x)$$

where $f: \mathbb{R}^n \rightarrow \mathbb{R}$ is the objective function.

In some optimization problems the objective function variables may be subject to some conditions, defined by the problem constraint functions. These constrained problems can be defined as in (2):

$$\begin{aligned} &\min_{x \in \mathbb{R}^n} f(x) \\ &\text{subject to } c_i(x) = 0, \ i \in \mathcal{E} \\ &\qquad\qquad c_i(x) \leqslant 0, \ i \in \mathcal{I} \end{aligned} \tag{2}$$

where:

- $f: \mathbb{R}^n \rightarrow \mathbb{R}$ is the objective function;
- $c_i(x) = 0, i \in \mathcal{E}$, with $E = \{1, 2, \ldots, t\}$, define the problem equality constraints;
- $c_i(x) \leqslant 0, i \in \mathcal{I}$, with $I = \{t+1, t+2, \ldots, m\}$, represent the inequality constraints;
- $\Omega = \{x \in \mathbb{R}^n : c_i = 0, i \in \mathcal{E} \wedge c_i(x) \leqslant 0, i \in \mathcal{I}\}$ is the set of all feasible points, i.e., the feasible region.

Unconstrained and constrained optimization problems can be found in many real life problems, and in many cases: the objective function values and/or its constraints are the result of complex and time consuming simulations; its values are obtained experimentally or by natural phenomena observation; the analytic functions might be too complex or even not be available; the samples have noise. There are also some cases where the objective function is non-smooth or non-differentiable, or even non-continuous. In such cases derivative based methods cannot be used to solve these problems, as presented in Conn, Scheinberg, and Vicente (2009).

Some of the possible solutions, whenever this type of problem must be solved, include the use of heuristic methods such as particle swarm (Wu et al., 2014), hybrid methods, e.g. particle swarm method followed by a global minimization method (Vaz and Vicente, 2007 and Vaz and Vicente, 2009), tabu search and simulated annealing algorithms as presented in Hedar et al., 2002 and Hedar and Fukushima, 2006, or genetic algorithms (Boudjelaba, Ros, & Chikouche, 2014).

These methods can be used if the cost of the objective function evaluation is negligible, otherwise they must be avoided. Instead, other derivative free algorithms, namely deterministics algorithms, can be used. Such algorithms include direct search methods that do not use derivatives or approximations to them (Lewis et al., 2000, Kolda et al., 2003 and Hooke and Jeeves, 1961).

In Correia et al., 2010 and Mestre et al., 2010 it was presented a Java API (Application Programming Interface), with remote access, which includes only Direct Search Optimization Methods for solving both unconstrained and constrained optimization problems. The objective of this API is to be included in other software packages, to solve problems where derivative based methods cannot be used. It was used in location estimation problems by the authors in Mestre et al., 2012 and Mestre et al., 2013 to tune the LEA (Location Estimation Algorithm) and adapt them to the mobile terminals. While inMestre et al. (2012) a Fuzzy Logic based LEA was implemented and the API was used to tune the parameters/transitions of membership functions and adjust the weights of OWA (Ordered Weighted Averaging), in Mestre et al. (2013) the API was used to tune the internal parameters of the Weighted k-Nearest Neighbour algorithm and a scaling factor for the RSSI (Received Signal Strength) values. In both cases, because of the nature of the data, derivative based methods could not be used.
In this context, the most used techniques to solve constrained problems consist on transforming constrained problems into unconstrained problems that are easier to solve, which solution is related with the solution of the original problem. One of such techniques consists in using Penalty and Barrier methods, which are used in this work.

Penalty and Barrier functions have been widely used and studied in the last years, for example by Byrd et al., 2008, Chen and Goldfarb, 2006, Fletcher, 1997, Gould et al., 2003, Leyffer et al., 2006, Klatte and Kummer, 2002, Mongeau and Sartenaer, 1995 and Zaslavski, 2005, due to its ability to deal with Degenerated Problems.
Exact Penalty Methods have been successfully used to solve Mathematical Programs with Complementary Constraints, by Benson et al., 2006, Leyffer et al., 2006, Rodrigues and Monteiro, 2006 and Rodrigues et al., 2009. They were also used in Constrained NonLinear Programming to assure the admissibility of sub-problems and the iteration reliability by Byrd et al., 2008 and Chen and Goldfarb, 2006.
Combination of Penalty Methods and Fuzzy concepts have been used by several authors such as Bustince et al., 2013, Chen et al., 2013, Gouicem et al., 2012, Bogdana and Milan, 2009 and Jamison and Lodwick, 2001. In Bustince et al. (2013) the concept of penalty function is used to determine which aggregation function should be applied, i.e., the objective is to choose which aggregation function will output the best results. Chen et al. (2013) use Penalty concepts to construct a new objective function and avoid monotonicity and coincident clustering results. In Gouicem et al. (2012), Fuzzy is used for image

reconstruction and Penalty is used in edge detection to penalise pixels for which it was not detected an edge. Penalty is used to transform Linear constrained problems into unconstrained problems by Bogdana and Milan,
2009 and Jamison and Lodwick, 2001.
These authors add the Penalty concept to Fuzzy, i.e., Penalty is used as an auxiliary tool used together with Fuzzy algorithms. The objective of the present work is different: use Fuzzy as a tool to generate Penalty functions that can be used together with Direct Search Optimization Methods. An improved version of a method presented by the authors in Matias et al. (2012), and a new Penalty function based on Fuzzy Logic, which adapts the penalty to apply based on its previous value, are presented. By using Fuzzy Logic to generate a Penalty Function it is possible to do a progressive penalization of the objective function when the problem constraints are violated.
With these two penalty functions it is expected to obtain a better performance of direct search methods, i.e., a lower number of objective function evaluations.

## 2. Penalty and Barrier functions

Let us consider the above presented constrained problem (2), Penalty and Barrier methods were developed to solve such problems by solving a sequence of unconstrained problems, i.e., a transformation of the original problem into a sequence of unconstrained problems is made.
Penalty and Barrier Methods comprise two processes (Fig. 1):

External Process (EP) – where a succession of Unconstrained Optimization Problems is created from a constrained problem;

Internal Process (IP) – where each of the previously generated problems, the Unconstrained Optimization Problems are solved.

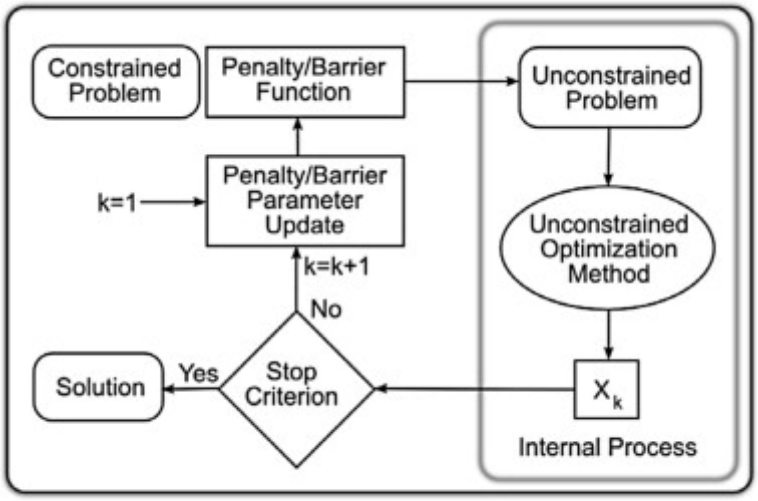


Fig. 1.
Penalty and Barrier methods - general process.

When Penalty and Barrier functions are used a new objective function, $\Phi_k$, based on information from the original problem is created. Therefore a succession of Unconstrained Optimization Problems are obtained. These new problems depend on a positive parameter, $r_k$ (Penalty/Barrier parameter) which solutions $x^*(r_k)$ converge to the solution of the original problems $x^*$ (*External Process*).
Direct Search methods are then used to solve the resulting Unconstrained Optimization Problems (*Internal Process*). At each iteration *k* the problem to be solved by the Internal Process is:

$$\min_{x_k \in \mathbb{R}^n} \Phi(x_k, r_k) = \min_{x_k \in \mathbb{R}^n} (f(x_k) + r_k p(x))$$

where *r* is a penalty parameter and *p* is a Penalty/Barrier function that penalises/refuses points that lie outside the feasible region.
Barrier methods can be used only when we have an initial feasible point. These methods were widely presented in Doyle (2003).
According to Freund (2004) a barrier function can be defined as a function $b: R^n \rightarrow R$, that satisfies the following conditions:

- $b(x) = \infty$, for all *x* that does not satisfies $c_i(x) < 0$ for any $i \in \{1, 2, \ldots, m+t\}$;
- $0 \leqslant b(x) \leqslant +\infty$, for all *x* that satisfies $c_i(x) < 0,\ \forall i \in \{1, 2, \ldots, m+t\}$;
- $b(x) \rightarrow \infty$ when $\max_i \{c_i(x)\} \rightarrow 0$.

On the other hand when we have infeasible initial points and when an infeasible solution can be tolerated, Penalty methods can be used. Its objective is to penalise the constraints violation, allowing infeasible points to exist in the iterative process (although penalized), instead of creating a barrier on the constraint boundary.

According to Freund (2004) a Penalty Function can be defined as a function $p: R^n \rightarrow R$ that satisfies:

- $p(x)=0$ if $c_i(x) \leqslant 0$;
- $p(x)>0$ if $c_i(x)>0$.

In this paper two Fuzzy Logic based penalty functions using different approaches are presented. These functions impose a progressive penalty to the constraints violation by mixing the concepts of a barrier (when the constraints are highly violated) and a penalty (when the constraints violation is low).

For performance analysis purposes the following classic Penalty and Barrier functions were implemented:

- Extreme Barrier Function (EB);
- Progressive Barrier Function (PB);
- Classical Penalty Function (CP);
- Static/Dynamic Penalty Function (SP);
- $\ell_1$ Penalty Function ($\ell_1$).

In the last years alternatives to these classic Penalty and Barrier methods have been developed. Polyakova and Karelin (2014) and Wang and Yuan (2014) presented methods based on Exact Penalty, assuming that the objective and constraint functions are Lipschitz quasidifferentiable/differentiable functions. Chen, Lu, and Kei Pong (2014)presented Exact Penalty methods for Non-Lipschitz Optimization, which considers that the objective function is continuous, and uses approximations to the derivatives. Xu and Hesthaven (2014) used multi-domain spectral Penalty methods for partial differential equations, based on approximate fractional derivatives. In Jayswal and Choudhury (2014) it was used an Exact Penalty Function method, for multiobjective optimization, which requires approximations to the derivatives. Therefore the above mentioned methods cannot be used to solve the type of problems that are the objective of our research.

Results obtained with classic Penalty and Barrier functions (EB, PB, CP, SP and $\ell_1$) were compared with those obtained using the Fuzzy Penalty Functions (FLP1 and FLP2) presented in subsections 2.1.1 and 2.1.2.

## 2.1. Fuzzy penalty functions

As presented by the authors in Matias et al. (2012) the main objective of using Fuzzy Logic to build these Penalty Functions is to progressively penalise the constraints violation. This allows to heavily penalise functions that have a high constraint violation and impose a low penalty to functions that barely violate the constraints. The transition between low penalty (Penalty Function like behavior) and a high penalty (Barrier Function like behavior) must be progressive.

### *2.1.1. Fuzzy Logic penalty function 1 (FLP1)*

In this work simple trapezoidal membership functions and a simple inference engine are used, to demonstrate the feasibility of the proposed approach and define its initial framework. Therefore more complex membership functions and other set of rules can be defined to cope with the specificities of the problem to be solved.

In both Fuzzy Logic based functions (FLP1 and FLP2) the first step is to determine the degree of constraint violation. After calculating the value for the constraint violation, the next step is the fuzzification procedure. This is achieved using the set of membership functions presented in Fig. 2, that classify the constraints violation as ‘Low’, ‘Medium’, ‘High ’ or ‘Very High’.

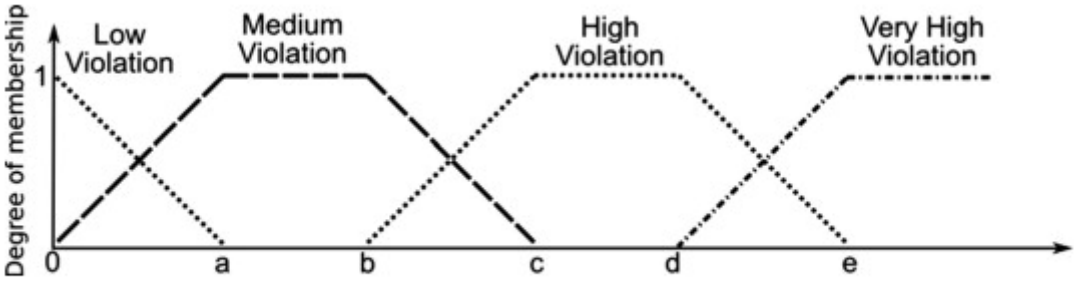


Fig. 2.
Membership functions.

After calculating the grades of membership ($\mu_1, \mu_2, \mu_3$ and $\mu_4$, corresponding to the 'Low', 'Medium', 'High ' or 'Very High' membership functions) the next step is to calculate the value of the penalty to be applied to the constraint violation, using an inference engine. For this step the following simple IF–THEN rules are used:

- IF the constraint violation is 'Low' THEN apply a 'Low' penalty;
- IF the constraint violation is 'Medium' THEN apply a 'Medium' penalty;
- IF the constraint violation is 'High' THEN apply a 'High' penalty;
- IF the constraint violation is 'Very High' THEN apply a 'Very High' penalty;

Weights ($w_1, w_2, w_3$ and $w_4$) such that $w_1 > w_2 > w_3 > w_4$ are assigned for each degree of constraint violation ('Very High', 'High', 'Medium' and 'Low'). At this point, using this set of weights and the values of the membership $V_1, V_2, V_3$ and $V_4$ functions (related to $\mu_1, \mu_2, \mu_3$ and $\mu_4$), an OWA operator, (Yager, 1988, Yager, 1992, Yager and Kacprzyk, 1997, Beliakov et al., 2007 and Liu, 2006), is used to calculate the final value of the penalty to be applied to the constraints violations.

Considering $W=\{w_1, w_2, w_3, w_4\}$ a set of weights, ordered according to the grade of violation, and $M=\{V_1, V_2, V_3, V_4\}$ the set of computed membership values, the OWA operator is used according to the following expression $p_j = \sum_{i=1}^{n} w_i V_i$ with $n=4$ and $p_j$ the penalization of constraint $j$ . Values for the weights and the values for $a, b, c, d$ and $e$ were chosen empirically. Also the number of membership functions was chosen empirically.

As depicted in Fig. 3, the above steps are repeated for each constraint of the problem ($c_1 \ldots c_n$) for the input point $x$ . If no violation occurs, a null penalty will applied, otherwise the penalty ($p_j$) is calculated. In the first Fuzzy Logic Penalty Function (FLP1) the final value of the penalty ($p(x)$) to be applied to the function, i.e., the value that is returned to the optimization method, is obtained by the summation of all the $p_j$ values.

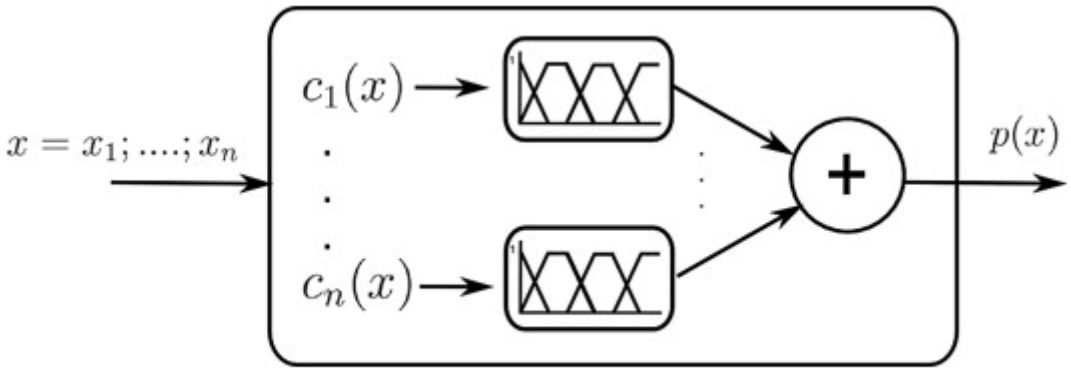


Fig. 3.
Block diagram for the FLP1.

### *2.1.2. Fuzzy Logic penalty function 2 (FLP2)*

In this second approach the penalty value calculated by the inference engine is not returned to the optimization method. Instead there is a new step that scales up or down the penalty based of the last and the current values of the penalty.

The objective is to loose the penalty if the new constraints violation for iteration *n* is lower than the violation at iteration $n-1$, and to penalise even further the objective function if the constraints violation at the current iteration is higher than the previous iteration. Therefore, it is an adaptive penalty function. In Fig. 4 is presented the principle of operation of this penalty function. The first step is to calculate the penalty for the current iteration (*n* ), using the same procedure as in FLPL1, and then compare it with the value obtained during the previous iteration ($p(x)_{n-1}$). The result of this comparison ($p(x)_n/p(x)_{n-1}$) is fed to an inference engine similar to the one used on the first phase (the only difference between them is the value for the weights ($w_1 \ldots w_4$) and the values for $a, b, c, d$ and *e*).

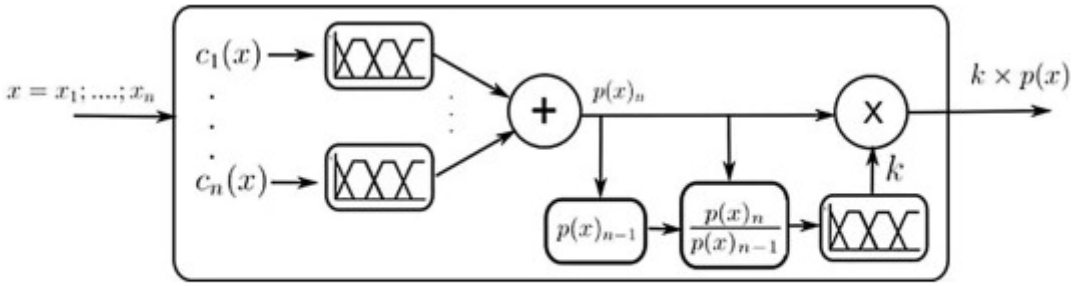


Fig. 4.
Block diagram for the FLP2.

This second inference engine calculates the scaling factor (*k*) to be applied to the current penalty. Therefore the final penalty value to be used is:

$$p_2(x) = k \times p_1(x) \tag{4}$$

where $p_2(x)$ is the new penalty value, *k* the scaling factor and $p_1(x)$ is the penalty calculated by the FLP1 inference engine.

## 3. Derivative-free methods

As above mentioned, constrained optimization problems can be solved by using Penalty/Barrier Methods which transform these problems into a sequence of problems of the form of problem define in (1), i.e., problems are solved (in the Internal Process) using methods that are usually used for solving Unconstrained Optimization Problems, when the derivatives are not known.

In the Java API, used to do the tests presented in this paper, the following derivative-free optimization algorithms were implemented:

- Opportunistic Coordinated search method;
- Hooke and Jeeves method;
- A version of Audet et al. method;
- Nelder–Mead method;
- A Convergent Simplex method.

These are well known optimization methods, that can be found in the literature. Opportunistic Coordinated search, Hooke and Jeeves and Audet et al. methods are Pattern Search or Directional Direct-Search Methods, (Conn et al., 2009), that determine possible points using fixed search directions during the iterative process: starting at an iteration $x_k$, the next iteration will be found in a pattern or grid of points, in the fixed directions, at a distance $s_k$, called step size. Nelder–Mead and the Convergent Simplex methods are Simplex Methods, (Conn et al., 2009), characterized by constructing an initial simplex and change the directions of search in each iteration, using reflection, expansion and contraction movements and shrunk steps.

## 4. Results and discussion

In this section are presented a set of tests that were made to assess the feasibility of the two proposed methods and compare their performance to the classic methods. For this comparison it was selected the PA problem (Audet & Dennis, 2004), two problems from the CUTE collection (Bongartz, Conn, Gould, & Toint, 1995), problems C801 and C802, and a set of problems from the Schittkowski collection (Schittkowski, 1987). From the Schittkowski collection, the following twenty-five problems were selected: S215; S218; S222; S223; S224; S225; S226; S227; S228; S229; S230; S231; S232; S233; S249; S250; S257; S264; S270; S315; S323; S325; S326; S337 and S343.
These last problems are also present in the *princetonlib* collection as presented in Rios and Sahinidis (2013). Details about these problems are presented

in Table 1, where *n* is the dimension of the problem, *i.c.* the number of inequality constraints and *e.c.*represents the number of equality constraints (the number of simple bounds is included in the number of inequality constraints).

Table 1.
List of test problems.

| Problem | *n* | *i.c.* | *e.c.* | Collection |
|---|---|---|---|---|
| PA | 2 | 3 | 0 | |
| C801 | 2 | 6 | 0 | CUTE |
| C802 | 2 | 6 | 0 | CUTE |
| S215 | 2 | 2 | 0 | *princetonlib* |
| S218 | 2 | 2 | 0 | *princetonlib* |
| S222 | 2 | 3 | 0 | *princetonlib* |
| S223 | 2 | 6 | 0 | *princetonlib* |
| S224 | 2 | 8 | 0 | *princetonlib* |
| S225 | 2 | 5 | 0 | *princetonlib* |
| S226 | 2 | 4 | 0 | *princetonlib* |
| S227 | 2 | 2 | 0 | *princetonlib* |
| S228 | 2 | 2 | 0 | *princetonlib* |
| S229 | 2 | 4 | 0 | *princetonlib* |
| S230 | 2 | 2 | 0 | *princetonlib* |
| S231 | 2 | 2 | 0 | *princetonlib* |
| S232 | 2 | 5 | 0 | *princetonlib* |
| S233 | 2 | 1 | 0 | *princetonlib* |
| S249 | 3 | 2 | 0 | *princetonlib* |
| S250 | 3 | 8 | 0 | *princetonlib* |
| S257 | 4 | 2 | 0 | *princetonlib* |
| S264 | 4 | 3 | 0 | *princetonlib* |
| S270 | 5 | 5 | 0 | *princetonlib* |
| S315 | 2 | 3 | 0 | *princetonlib* |
| S323 | 2 | 4 | 0 | *princetonlib* |
| S325 | 2 | 2 | 1 | *princetonlib* |
| S326 | 2 | 4 | 0 | *princetonlib* |
| S337 | 3 | 3 | 0 | *princetonlib* |
| S343 | 3 | 8 | 0 | *princetonlib* |

Each one of these problems is solved using the five internal process methods (presented in Section 3) and for each internal method the problem is solved using all the Penalty and Barrier functions presented in Section 2.

### 4.1. Input parameters and return values

Internal parameters of the used optimization methods and the stopping criteria (with a tolerance of $10^{-5}$) were set as in Correia et al. (2010). For the first Fuzzy Penalty function (FLP1) and the first inference engine of FLP2 it was used $w_1 = 100000,\ w_2 = 1000,\ w_3 = 10,\ w_4 = 0.01$ and values for $a, b, c, d$ and $e$ parameters were chosen according to the tests presented in the "Results" subsection. For the second inference engine of FLP2 $w_1 = 1.0,\ w_2 = 0.75,\ w_3 = 0.5,\ w_4 = 0.25,\ a = 0.2,\ b = 0.4,\ c = 0.6,\ d = 0.8$ and $e = 1.0$ were used.

The implemented Penalty and Barrier methods return the following values:

- Number of EP iterations – *k*;
- Number of Penalty/Barrier function evaluations – *nEvals*;
- Last iteration – $x_k$;
- Value of the Penalty/Barrier function at the last iteration – $\Phi(x_k)$;
- Best feasible solution found (if any) – $x_{kf}$;
- Value of the objective function at the best feasible solution found – $f(x_{kf})$;
- Iteration where the best feasible solution was found – *kf*;
- Best infeasible solutions found (if any) – $x_{ki}$;
- Value of the objective function at the best infeasible solution found – $f(x_{ki})$;
- Value of the Penalty/Barrier function value at the best infeasible solution – $\Phi(x_{ki})$;
- Iteration were the best infeasible solution was found – *ki*;
- Constraint violation value at the best infeasible solution, $V = \Phi(x_{ki}) - f(x_{ki})$.

The above presented parameters were recorded for the used test problems, and these data were used to assess the performance of the proposed Penalty/Barrier functions and compare their performance to that of the classic Barrier/Penalty functions.

### 4.2. Results

In this section are presented the results obtained for the above mentioned test problems. For the tables presented in this section, the values for the Penalty and Barrier function ($PF$) are: $PF_1$ = Extreme Barrier; $PF_2$ = Progressive Barrier; $PF_3$ = Classical Penalty; $PF_4$ = Static Penalty; $PF_5$ = Dynamic Penalty; $PF_6 = \ell_1$ Penalty; $PF_7$ = Fuzzy Logic Penalty 1; $PF_8$ = Fuzzy Logic Penalty 2.

To analyse the impact that the chosen membership functions have on the final results, it was made a first set of tests were different values were given to parameters $a, b, c, d$ and $e$ of Fig. 2, as follows:

- Test 1 - $a=0.25, b=0.5, c=0.75, d=1, e=1.25$;
- Test 2 - $a=0.5, b=1, c=1.5, d=2, e=2.5$;
- Test 3 - $a=0.75, b=1.5, c=2.25, d=3, e=3.75$;
- Test 4 - $a=1, b=2, c=3, d=4, e=5$;
- Test 5 - $a=1.25, b=2.5, c=3.75, d=5, e=6.25$;
- Test 6 - $a=1.5, b=3, c=4.5, d=6, e=7.5$;
- Test 7 - $a=1.75, b=3.5, c=5.25, d=7, e=8.75$;
- Test 8 - $a=2, b=4, c=6, d=8, e=10$.

In Table 2 are presented the results obtained when solving all the test problems using the above mentioned parameters. These results are presented by Internal Method (IM) and Test number and there are two types of results presented: percentage of successful runs (Suc.), i.e., percentage of problems that were solved and have a feasible solution; percentage of best objective function evaluations, i.e., percentage of problems where the lower number of objective functions evaluations (B.#.E.) was obtained.

Table 2.
Performance comparison for different membership functions.

| | IM1 | | IM2 | | IM3 | | IM4 | | IM5 | |
|---|---|---|---|---|---|---|---|---|---|---|
| Test# | Suc. (%) | B.#.E. (%) | Suc. (%) | B.#.E. (%) | Suc. (%) | B.#.E. (%) | Suc. (%) | B.#.E. (%) | Suc. (%) | B.#.E. (%) |
| 1 | 85.7 | 67.9 | 85.7 | 64.3 | 85.7 | 71.4 | 82.1 | 25.0 | 82.1 | 25.0 |
| 2 | 85.7 | 71.4 | 82.1 | 71.4 | 82.1 | 64.3 | 82.1 | 25.0 | 82.1 | 25.0 |
| 3 | 78.6 | 64.3 | 75.0 | 60.7 | 75.0 | 53.6 | 82.1 | 25.0 | 82.1 | 25.0 |
| 4 | 78.6 | 67.9 | 75.0 | 53.6 | 75.0 | 50.0 | 78.6 | 17.9 | 82.1 | 21.4 |
| 5 | 71.4 | 64.3 | 67.9 | 60.7 | 75.0 | 50.0 | 78.6 | 28.6 | 82.1 | 10.7 |
| 6 | 71.4 | 60.7 | 67.9 | 57.1 | 67.9 | 46.4 | 71.4 | 21.4 | 78.6 | 21.4 |
| 7 | 71.4 | 60.7 | 67.9 | 57.1 | 67.9 | 50.0 | 71.4 | 32.1 | 75.0 | 28.6 |
| 8 | 71.4 | 67.9 | 64.3 | 57.1 | 67.9 | 42.9 | 71.4 | 17.9 | 75.0 | 28.6 |

IM1, IM2, IM3, IM4 and IM5 are “Opportunistic Coordinated search method”, “Hooke and Jeeves method”, “A version of Audet et al. method”, “Nelder–Mead method” and “A Convergent Simplex method” respectively.

As it can be observed the best overall performance was achieved using $a=0.25$, $b=0.5$, $c=0.75$, $d=1$, and $e=1.25$, so these values were the parameters chosen for the rest of the tests below presented.

In Table 3 are presented the results obtained when solving all test problems using the proposed Fuzzy Logic based and classic penalty/barrier functions. These results are presented by Internal Method (IM) and *PF* function, and as above are presented the percentage of successful runs (Suc.) and the percentage of best objective function evaluations.

Table 3.
Performance comparison of the used Penalty and Barrier functions per Internal Method.

| | IM1 | | IM2 | | IM3 | | IM4 | | IM5 | |
|---|---|---|---|---|---|---|---|---|---|---|
| PF | Suc. (%) | B.#.E. (%) | Suc. (%) | B.#.E. (%) | Suc. (%) | B.#.E. (%) | Suc. (%) | B.#.E. (%) | Suc. (%) | B.#.E. (%) |
| 1 | 82.1 | 67.9 | 78.6 | 67.9 | 82.1 | 64.3 | 78.6 | 25.0 | 78.6 | 35.7 |
| 2 | 14.3 | 14.3 | 14.3 | 10.7 | 10.7 | 3.6 | 7.1 | 7.1 | 60.7 | 21.4 |
| 3 | 14.3 | 14.3 | 71.4 | 10.7 | 10.7 | 3.6 | 14.3 | 7.1 | 64.3 | 17.9 |
| 4 | 14.3 | 14.3 | 71.4 | 10.7 | 10.7 | 3.6 | 14.3 | 7.1 | 60.7 | 17.9 |
| 5 | 14.3 | 14.3 | 14.3 | 10.7 | 10.7 | 3.6 | 10.7 | 7.1 | 57.1 | 10.7 |
| 6 | 75.0 | 39.3 | 85.7 | 32.1 | 71.4 | 25.0 | 60.7 | 28.6 | 71.4 | 25.0 |
| 7 | 85.7 | 64.3 | 85.7 | 71.4 | 85.7 | 71.4 | 82.1 | 42.9 | 82.1 | 35.7 |
| 8 | 82.1 | 57.1 | 82.1 | 60.7 | 85.7 | 60.7 | 85.7 | 53.6 | 82.1 | 32.1 |

Data shows that when comparing the performance of the “traditional” Penalty and Barrier functions with the first Fuzzy Logic based penalty function ($PF=7$), the last one had always the best results for the percentage of successful runs and the lower number of objective function evaluations. Therefore, for this set of test problems, it can be considered a good penalty function.
Comparing with the performance of PLF1, it can be observed that the addition of another processing stage to the Fuzzy Algorithm (FLP2) had a positive impact on the performance of IM4 when analysing the number of successful results, a neutral impact on IM3 and IM5, and a negative impact on IM1 and IM2. Analysing the number of objective function evaluations, better results were obtained for IM4.

If a comparison is made between FLP2 and the “classic” methods, it can be concluded that the first had better overall performance.

These data are corroborated by the values presented on Table 4, which are presented the overall values for the percentage of successful runs (Suc.) and percentage of best number of objective function evaluations (B.#Ev.).

Table 4.
Overall performance comparison for the used Penalty and Barrier functions.

| PF | Suc. (%) | B.#Ev. (%) |
|---|---|---|
| 1 | 80.00 | 52.14 |
| 2 | 21.43 | 11.43 |
| 3 | 35.00 | 10.71 |
| 4 | 34.29 | 10.71 |
| 5 | 21.43 | 9.29 |
| 6 | 72.86 | 30.00 |
| 7 | 84.29 | 57.14 |
| 8 | 83.57 | 52.86 |

Data confirms that the overall performance of the Fuzzy Logic based penalty functions was better than the other Penalty and Barrier functions. The proposed methods had a very good performance, both in terms of the number of successful runs, and the number problems that were solved using less objective function evaluations.

## 5. Conclusion

Using Fuzzy Logic to build penalty functions allowed to model the uncertainty associated with the degree of constraint violation. A framework to support such a need was developed, which includes a base approach for both membership functions (although other functions are admissible, e.g., gaussian function) and parameters/transitions of the membership functions.

Regarding to the two implemented penalty functions, the first calculates the penalty based only on information about the current constraints violation, and the second also takes into consideration if the penalty for the current iteration is higher or lower than the previous iteration, i.e., it assesses if the current solution is getting out of the infeasible region or getting even deeper into it. Therefore it is an adaptive Penalty function.

Because these Penalty functions can be used with Direct Search Methods, they are suitable for applications where the cost of evaluating the objective function cannot be negligible. Also in applications where black box problems must be solved, the developed methods can be used.

As a conclusion, the proposed functions are suitable as penalty functions to be used with direct search methods. Because of its use with direct search methods, these penalty functions might not be so efficient as the derivative methods, however the combination of direct search methods and penalty functions can be used to solve some types of problems that derivative methods cannot. There might be also an issue related to the scale of the problem, constraints or its variables, which can make the method to have different sensitivity for the same variation of the input variables (at different points). Although this limitation can be overcome by changing the membership functions, as future work a dynamic selection/tuning of membership functions will be implemented.

In addition to the application of Fuzzy extensions in these penalty functions, future research directions also include the use of Fuzzy Logic based penalty functions (together with direct search methods) to tune position tracking algorithms (for indoor location using wireless networks), and adjust the internal parameters of Fuzzy Logic inference engines (e.g. adjust the coefficients of an aggregation function for a specific problem or application, such as image compression for transmission over a low bandwidth wireless link).

## References


1.
   - Audet and Dennis, 2004
   - C. Audet, J. Dennis
   - **A pattern search filter method for nonlinear programming without derivatives**
   - SIAM Journal on Optimization, 5 (2004), pp. 980–1010

2.
   - Beliakov et al., 2007
   - G. Beliakov, A. Pradera, T. Calvo
   - Aggregation Functions: A Guide for Practitioners, vol. 221Springer, Berlin Heidelberg, River Edge, NJ, USA (2007)
   -

3.
   - Benson et al., 2006
   - H. Benson, A. Sen, D. Shanno, R. Vanderbei
   - **Interior-point algorithms, penalty methods and equilibrium problems**
   - Computational Optimization and Applications, 34 (2006), pp. 155–182

4.
- Bogdana and Milan, 2009
- J. Bogdana, S. Milan
- **Penalty method for fuzzy linear programming with trapezoidal numbers**
- Yugoslav Journal of Operations Research, 19 (2009), p. 149
-

5.
- Bongartz et al., 1995
- I. Bongartz, A. Conn, N. Gould, P. Toint
- **Cute: Constrained and unconstrained testing environment**
- ACM Transactions and Mathematical Software, 1 (1995), pp. 123–160

6.
- Boudjelaba et al., 2014
- K. Boudjelaba, F. Ros, D. Chikouche
- **An efficient hybrid genetic algorithm to design finite impulse response filters**
- Expert Systems with Applications, 41 (2014), pp. 5917–5937

7.
- Bustince et al., 2013
- H. Bustince, A. Jurio, A. Pradera, R. Mesiar, G. Beliakov
- **Generalization of the weighted voting method using penalty functions constructed via faithful restricted dissimilarity functions**
- European Journal of Operational Research, 225 (2013), pp. 472–478 http://dx.doi.org/10.1016/j.ejor.2012.10.009

8.
- Byrd et al., 2008
- R. Byrd, J. Nocedal, R. Waltz
- **Steering exact penalty methods for nonlinear programming**
- Optimization Methods & Software, 23 (2008), pp. 197–213

9.
- Chen et al., 2013
- J. Chen, D. Pi, Z. Liu
- **An insensitivity fuzzy c-means clustering algorithm based on penalty factor**
- Journal of Software, 8 (2013)
-

10.

- Chen and Goldfarb, 2006
- L. Chen, D. Goldfarb
- **Interior-point l(2)-penalty methods for nonlinear programming with strong global convergence properties**
- Mathematical Programming, 108 (2006), pp. 1–36

11.
- Chen et al., 2014
- Chen, X., Lu, Z., & Kei Pong, T. (2014). Exact penalty methods for non-lipschitz optimization. ArXiv e-prints,.
-

12.
- Conn et al., 2009
- Conn, A.R., Scheinberg, K., & Vicente, L.N. (2009). Introduction to Derivative-Free Optimization. Philadelphia, USA: MPS-SIAM Series on Optimization, SIAM.
-

13.
- Correia et al., 2010
- A. Correia, J. Matias, P. Mestre, C. Serodio
- **Direct-search penalty barrier methods**
- Lecture Notes in Engineering and Computer Science – World Congress on Engineering, vol. 3IAENG, London, UK (2010) pp. 1729–1734
-

14.
- Doyle, 2003
- Doyle, M. (2003). A barrier algorithm for large nonlinear optimization problems [Ph.D. thesis]. California: Stanford University Stanford.
-

15.
- Fletcher, 1997
- R. Fletcher
- **Nonlinear programming without a penalty function**
- Mathematical Programming, 91 (1997), pp. 239–269
-

16.
- Freund, 2004
- Freund, R. (2004). Penalty and Barrier Methods for Constrained Optimization. Massachusetts Institute of Technology Massachusetts.

- 

17.
    - Gouicem et al., 2012
    - A. Gouicem, K. Benmahammed, R. Drai, M. Yahi, A. Taleb-Ahmed
    - **Multi-objective GA optimization of fuzzy penalty for image reconstruction from projections in X-ray tomography**
    - Digital Signal Processing, 22 (2012), pp. 486–496
18.
    - Gould et al., 2003
    - Gould, N.I.M., Orban, D., & Toint, P.L. (2003). An interior-point $l_1$-penalty method for nonlinear optimization. Technical Report Rutherford Appleton Laboratory Chilton Oxfordshire, UK.
    - 
19.
    - Hedar et al., 2002
    - A.-R. Hedar, M. Fukushima
    - **Hybrid simulated annealing and direct search method for nonlinear unconstrained global optimization**
    - Optimization Methods and Software (2002), pp. 891–912
20.
    - Hedar and Fukushima, 2006
    - A.-R. Hedar, M. Fukushima
    - **Tabu search directed by direct search methods for nonlinear global optimization**
    - European Journal of Operational Research, 170 (2006), pp. 329–349
1.
    - Hooke and Jeeves, 1961
    - R. Hooke, T. Jeeves
    - **Direct search solution of numerical and statistical problems**
    - Journal of the Association for Computing Machinery, 8 (1961), pp. 212–229
2.
    - Jamison and Lodwick, 2001
    - K. Jamison, W.A. Lodwick
    - **Fuzzy linear programming using a penalty method**
    - Fuzzy Sets and Systems, 119 (2001), pp. 97–110
3.

- Jayswal and Choudhury, 2014
- A. Jayswal, S. Choudhury
- **An exact l1 exponential penalty function method for multiobjective optimization problems with exponential-type invexity**
- Journal of the Operations Research Society of China, 2 (2014), pp. 75–91

4.

- Klatte and Kummer, 2002
- D. Klatte, B. Kummer
- **Constrained minima and lipschitzian penalties in metric spaces**
- SIAM Journal on Optimization, 13 (2002), pp. 619–633

5.

- Kolda et al., 2003
- T. Kolda, R. Lewis, V. Torczon
- **Optimization by direct search: New perspectives on some classical and modern methods**
- SIAM Review, 45 (2003), pp. 385–482

6.

- Lewis et al., 2000
- R. Lewis, V. Torczon, M. Trosset
- **Direct search methods: Then and now**
- Journal of Computational and Applied Mathematics, 45 (2000), pp. 191–207

7.

- Leyffer et al., 2006
- S. Leyffer, G. Calva, J. Nocedal
- **Interior methods for mathematical programs with complementarity constraints**
- SIAM Journal on Optimization, 17 (2006), pp. 52–77

8.

- Liu, 2006
- X. Liu
- **Some properties of the weighted OWA operator**
- IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 36 (2006), pp. 118–127

9.

- Matias et al., 2012

- J. Matias, P. Mestre, A. Correia, P. Couto, C. Serodio, P. Melo-Pinto
- **Penalty fuzzy function for derivative-free optimization**
- ,in: P. Melo-Pinto, P. Couto, C. Serodio, J. Fodor, B. De Baets (Eds.), Eurofuse 2011, Advances in Intelligent and Soft Computing, vol. 107, Springer, Berlin/Heidelberg (2012), pp. 293–301 10.1007/978-3-642-24001-0_27

10.

- Mestre et al., 2012
- P. Mestre, L. Coutinho, L. Reigoto, J. Matias, A. Correia, P. Couto, *et al.*
- **Indoor location using fingerprinting and Fuzzy Logic**
- ,in: P. Melo-Pinto, P. Couto, C. Ser odio, J. Fodor, B. De Baets (Eds.), Eurofuse 2011, Advances in Intelligent and Soft Computing, vol. 107, Springer, Berlin Heidelberg (2012), pp. 363–374

11.

- Mestre et al., 2010
- P. Mestre, J. Matias, A. Correia, C. Serodio
- **Direct search optimization application programming interface with remote access**
- IAENG International Journal of Applied Mathematics, 40 (2010), pp. 251–261
-

12.

- Mestre et al., 2013
- P. Mestre, L. Reigoto, L. Coutinho, A. Correia, J. Matias, C. Serodio
- **Calibration procedures for indoor location using fingerprinting**
- ,in: G.-C. Yang, S.-I. Ao, L. Gelman (Eds.), IAENG Transactions on Engineering Technologies, Lecture Notes in Electrical Engineering, vol. 229, Springer, Netherlands (2013), pp. 349–363

13.

- Mongeau and Sartenaer, 1995
- M. Mongeau, A. Sartenaer
- **Automatic decrease of the penalty parameter in exact penalty function methods**
- European Journal of Operational Research, 3 (1995), pp. 686–699

14.

- Polyakova and Karelin, 2014
- Polyakova, L., & Karelin, V. (2014). Exact penalty methods for nonsmooth optimization. In: 20th International Workshop on Beam Dynamics and Optimization (BDO), 2014 (pp. 1–2).
-

15.

- Rios and Sahinidis, 2013
- L. Rios, N. Sahinidis
- **Derivative-free optimization: a review of algorithms and comparison of software implementations**
- Journal of Global Optimization, 56 (2013), pp. 1247–1293 http://dx.doi.org/10.1007/s10898-012-9951-y

16.

- Rodrigues and Monteiro, 2006
- H. Rodrigues, M. Monteiro
- **Solving mathematical programs with complementarity constraints with nonlinear solvers**
- Lecture Notes in Economics and Mathematical Systems – Recent Advances in Optimization, 563 (2006), pp. 415–424

17.

- Rodrigues et al., 2009
- H. Rodrigues, M. Monteiro, A. Vaz
- **A MPCC approach on a stackelberg game in an electric power market: changing the leadership**
- International Journal of Computer Mathematics, 86 (2009), pp. 1921–1931

18.

- Schittkowski, 1987
- K. Schittkowski
- More test examples for nonlinear programming codes, Economics and Mathematical Systems, Springer-Verlag, Berlin (1987)

19.

- Vaz and Vicente, 2007
- A. Vaz, L. Vicente
- **A particle swarm pattern search method for bound constrained global optimization**
- Journal of Global Optimization, 39 (2007), pp. 197–219

20.

- Vaz and Vicente, 2009
- A.I.F. Vaz, L.N. Vicente
- **PSwarm: A hybrid solver for linearly constrained global derivative-free optimization**
- Optimization Methods Software, 24 (2009), pp. 669–685

1.

- Wang and Yuan, 2014
- X. Wang, Y. Yuan
- **An augmented lagrangian trust region method for equality constrained optimization**
- Optimization Methods and Software, 0 (2014), pp. 1–24

2.

- Wu et al., 2014
- G. Wu, D. Qiu, Y. Yu, W. Pedrycz, M. Ma, H. Li
- **Superior solution guided particle swarm optimization combined with local search techniques**
- Expert Systems with Applications, 41 (2014), pp. 7536–7548

3.

- Xu and Hesthaven, 2014
- Q. Xu, J.S. Hesthaven
- **Stable multi-domain spectral penalty methods for fractional partial differential equations**
- Journal of Computational Physics, 257 (Part A) (2014), pp. 241–258

4.

- Yager, 1988
- R.R. Yager
- **On ordered weighted averaging aggregation operators in multicriteria decisionmaking**
- IEEE Transactions on Systems, Man and Cybernetics, 18 (1988), pp. 183–190

5.

- Yager, 1992
- R.R. Yager
- **Applications and extensions of OWA aggregations**
- International Journal of Man-Machine Studies, 37 (1992), pp. 103–122

6.

- Yager and Kacprzyk, 1997
- R.R. Yager, J. Kacprzyk
- **The Ordered Weighted Averaging Operators: Theory and Applications**
- Springer, River Edge, NJ, USA (1997)

7.

- Zaslavski, 2005

- A. Zaslavski
- **A sufficient condition for exact penalty in constrained optimization**
- SIAM Journal on Optimization, 16 (2005), pp. 250–262