A Computational Study on Different Penalty Functions with DIRECT Algorithm

Ana Maria A.C. Rocha[1,2] and Rita Vilaça[2]

[1] Department of Production and Systems, School of Engineering
arocha@dps.uminho.pt
[2] Algoritmi R&D Centre
University of Minho, 4710-057 Braga, Portugal
rita.pinto.vilaca@gmail.com

Abstract. The most common approach for solving constrained optimization problems is based on penalty functions, where the constrained problem is transformed into an unconstrained problem by penalizing the objective function when constraints are violated. In this paper, we analyze the implementation of penalty functions, within the DIRECT algorithm. In order to assess the applicability and performance of the proposed approaches, some benchmark problems from engineering design optimization are considered.

Keywords: Global optimization, constrained optimization, DIRECT algorithm, penalty function.

1 Introduction

This paper aims to illustrate the behavior of some penalty functions combined with the DIRECT algorithm for solving nonlinear global optimization problems of the form:

$$\begin{aligned} &\text{minimize} \quad f(x) \\ &\text{subject to } g_i(x) \leq 0,\ i = 1, \ldots, m \\ &\qquad\qquad x \in \Omega, \end{aligned} \tag{1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$, $g : \mathbb{R}^n \to \mathbb{R}^m$ are nonlinear continuous functions, $\Omega = \{x \in \mathbb{R}^n : l \leq x \leq u\}$ and x corresponds to the main variables, decision or controllable through which will optimize $f(x)$. We do not assume that functions f and g are convex. There may exist many local minima in the feasible region. This class of global optimization problems arises frequently in engineering applications. Specially for large scale problems of type (1), derivative-free and stochastic methods are the most well-known and used methods.

DIRECT is a deterministic global optimization algorithm developed by Jones, Perttunen and Stuckman [19] and the name "DIviding RECTangles" describes the way the algorithm moves towards the optimum. The DIRECT method starts the iterative process by dividing the solution space into hyperrectangles, using criteria (size and value of the function center) well defined. Hence, the algorithm,

B. Murgante et al. (Eds.): ICCSA 2013, Part I, LNCS 7971, pp. 318–332, 2013.

based on a space-partitioning scheme, performs both global exploration and local exploitation [9].

DIRECT was initially developed to solve difficult global optimization problems with simple bound constraints [9,10]. However, over the years the method has been improved and some modifications were done in order to solve other kind of problems [11,12].

There is a set of strategies for solving nonlinear constrained problems consisting of transforming the constrained problem into a sequence of unconstrained subproblems, whose solutions are related in some way to the solution of the original problem. Solutions of the successive unconstrained subproblems will eventually converge to the solution of the original constrained problem. The subproblems created by penalty methods involve a penalty function that incorporates the objective function and the constraints of the problem. The penalty function may include one or more penalty parameters which determine the relative importance of each constraint, or set of constraints. When these parameters are suitably modified, the effects of constraints become increasingly evident in the sequence of generated problems.

Penalty methods are simple to implement, are applicable to a broad class of problems and take advantage of powerful unconstrained minimization methods. Thus, they have been widely accepted in practice as an effective class of methods for constrained optimization. The appropriate selection of penalty parameter values is vital for a fast convergence. A function that determines its value at the beginning of each round of optimization may be defined. Different penalty functions may emerge depending on the way penalties vary throughout the minimization process.

In this paper we intend to analyze the performance of different penalty approaches embedded in the DIRECT algorithm, namely, the l_1 penalty function, Quadratic Penalty function, Dynamic Penalty function, Hyperbolic Penalty function and Augmented Lagrangian Penalty function. Some preliminary results are presented when solving a benchmark of engineering design optimization problems with the proposed methods.

The remainder of this paper is as follows. We briefly describe the DIRECT algorithm in Section 2. In Section 3 we briefly introduce the penalty function technique and in Section 4 the proposed penalized DIRECT algorithm approaches are outlined. Section 5 describes the experimental results and finally we draw the conclusions of this study in Section 6.

2 DIRECT Algorithm

The DIRECT optimization algorithm was first presented in [19], and emerged by a modification to Lipschitzian optimization, that has proven to be effective in a wide range of application domains. The motivation for the DIRECT algorithm comes from a different way of looking at the Lipschitz constant. In particular, the Lipschitz constant is viewed as a weighting parameter that indicates how much emphasis is to be placed on global versus local search.

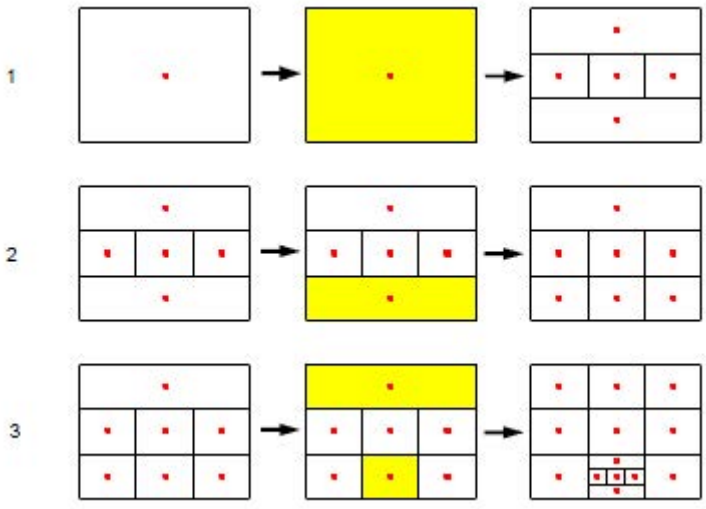

Fig. 1. First three iterations of DIRECT on a sample problem

DIRECT is a sampling algorithm and requires no knowledge of the objective function gradient. Furthermore, the algorithm samples points in the search space, and uses the information it has obtained to decide where to search next. DIRECT can be very useful when the objective function is a "black box" function or simulation. An example of using DIRECT to solve a large industrial problem can be found in [4].

DIRECT begins the optimization by transforming the domain of the problem into the unit hypercube and its center point, c, is evaluated. Then the hypercube is divided in three hyperrectangles along the coordinate with best objective function value. That is, the evaluation of the function at points $c \pm \delta e_i$ for $i = 1, 2, ..., n$ where c is the central point of the hypercube, δ is a third side of the hypercube and e_i is the ith unit vector. Then, the new points evaluated in this coordinate will be the central points of the new hyperrectangles. At each iteration, a set of *potentially optimal hyperrectangles* are defined for further divisions [10]. Subsequently, the central hyperrectangle that contains c must be partitioned into another three hyperrectangles along the coordinate with the second best value of the objective function and so on. Figure 1 shows an example of DIRECT algorithm.

DIRECT searches locally and globally by dividing all hyperrectangles that meet the following criteria. Thus, for a hyperrectangle i, the value of the objective function at c_i, the center of the hyperrectangle, and d_i, the distance from the center point to the vertices are required. A hyperrectangle j is said to be potentially optimal if there exists some $K > 0$, thought as the Lipschitz approximation value, such that,

$$f(c_j) - Kd_j \leq f(c_i) - Kd_i, \text{ for all } i \neq j \quad (2)$$

$$f(c_j) - Kd_j \leq f_{min} - \epsilon\,|f_{min}| \quad (3)$$

where f_{min} is the current best function value and $\epsilon > 0$ is a positive constant [11]. The parameter $\epsilon > 0$ was introduced in order to avoid an exhaustive local search. Tests performed with classic problems of global optimization showed that values between 10^{-7} and 10^{-2} did not degrade the performance of DIRECT. The value of 10^{-4} was suggested in [19] presenting results more robust, in practice.

In the last years many modifications have been suggested by some authors with the aim of improving the DIRECT [11,12,14]. For global optimization of noisy functions, [12] presented a new approach, designated by, noisy DIRECT algorithm. This algorithm can be divided into two phases. In the first phase, the algorithm must find multiple promising regions of interest. In the second phase, local search algorithms must be applied around those promising regions aiming to refine solutions. A method involving a non-traditional penalty function and a heuristic for determining the penalty parameters, was proposed by [14]. The method only solves inequality constrained problems. For the same type of problems, a traditional exact l_1 penalty function approach was implemented with DIRECT in [9], where the penalty parameter is maintained constant along the optimization process.

3 Penalty Function Technique

The most common approach to solve constrained optimization problems is based on penalty functions. In this approach, penalty terms are added to the objective function to penalize the objective function value of any solution that violates the constraints.

Penalty methods were originally proposed by Courant in 1940s [7] and later expanded by Carroll [3] and Fiacco & McCormick [8]. The idea of this method is to transform a constrained optimization problem into a sequence of unconstrained subproblems by adding a certain value to the objective function based on the amount of constraint violation present in a certain solution. The solutions of the successive unconstrained subproblems will eventually converge to the solution of the original constrained problem.

Although penalty functions are very simple and easy to implement they often require one or more penalty parameters that are usually problem dependent and chosen with a priori knowledge by users. The basic penalty approach defines a penalty function for each point x, herein denoted by $\Phi(x)$, by adding to the objective function value a penalty term, $P(x, \mu)$, that aims to penalize infeasible solutions

$$\Phi(x) = f(x) + P(x, \mu), \quad (4)$$

where $f(x)$ is the objective function of the constrained problem and μ is a positive real number denoted by penalty parameter. The goal of the penalty method is to solve a sequence of unconstrained subproblems, minimizing the penalty function.

Ideally the penalty parameter should be kept as low as possible, just above the limit below which infeasible solutions are near-optimal (this is called the minimum penalty rule) [5]. This is due to the fact that if the penalty is too high or too low, then the problem may be difficult to solve. A large penalty discourages the exploration of the infeasible region since the very beginning of the search process. On the other hand, if the penalty is too low, a lot of the search time will be spent exploring the infeasible region because the penalty will be negligible with respect to the objective function. Thus, the selection of appropriate penalty parameter values is vital for a fast convergence and accuracy. Further, the initial penalty parameter value also has a key role in the convergence behavior of the method.

According to (1) the level of constraints violation is measured by the vector

$$v_j(x) = \max\{0, g_j(x)\},\ j = 1, \ldots, m\,. \tag{5}$$

and a measure of constraints violation for a point x, is given by

$$\text{viol} = \sum_{j=1}^{m} v_j(x).$$

If viol $= 0$ then x is a feasible point, otherwise is infeasible.

4 The Proposed Penalized DIRECT Algorithm

Different penalty functions have been presented in the past which can be classified according to the way the penalties are added [1,5,13,23]. In this study, the l_1 penalty, the quadratic, the dynamic, the hyperbolic and the augmented Lagrangian penalty functions are implemented within the DIRECT algorithm. Different penalties may emerge depending on the way penalties vary throughout the minimization process. A brief description of these penalties follows.

4.1 l_1 Penalty

A traditional exact l_1 penalty function approach was already implemented in DIRECT [9]. In the linear penalty method the penalty term is the l_1 norm of the constraint violation. Although continuous, the penalty function l_1 is not differentiable at all points. This is the major disadvantage of the l_1 penalty method [18].

In this method the penalty term [9] is given by

$$P(x, \mu) = \mu \sum_{i=1}^{m} \max(0, g_i(x)) \tag{6}$$

where μ is the penalty parameter. Finally, the sequence of subproblems, parameterized by the penalty parameter μ, is solved by DIRECT algorithm. We remark that in [9] the penalty parameter is maintained constant during the optimization process.

4.2 Quadratic Penalty

In the quadratic penalty, the penalty term is based in the square of the constraint violation [18]. The penalty term of quadratic penalty method can be defined as,

$$P(x, \mu) = 2\mu \sum_{i=1}^{m} \max(0, g_i(x))^2 \tag{7}$$

where μ is the penalty parameter that tends to infinite. The update of the penalty parameter, in a k iteration, is given by

$$\mu^{k+1} = \min(\mu^{\max}, \alpha\mu^k)$$

where $\alpha > 1$ and $\mu^{\max}$ is an upper bound to the penalty parameter μ.

4.3 Dynamic Penalty

Joines and Houck [13] proposed that the penalty parameters should vary dynamically along the search according to exogenous schedule, as

$$P(x, \mu) = \mu \sum_{i=1}^{m} \max(0, g_i(x))^{\gamma(max(0, g_i(x)))} \tag{8}$$

where μ is a dynamically modified penalty parameter. This penalty parameter is updated by

$$\mu^{k+1} = \begin{cases} (C(k+1))^{\alpha} & \text{if viol} > 0 \\ \mu^k & \text{otherwise} \end{cases} \tag{9}$$

where k represents the iteration counter, and the constants C and α are set as 0.5 and 2, respectively. The power of the constraint violation, $\gamma(.)$, is a violation dependent constant: $\gamma(z) = 1$ if $z \leq 1$, and $\gamma(z) = 2$, otherwise. See, examples of dynamic penalties in [17,20].

Another interesting and quite efficient rule for the penalty update, found in the literature [20], is given by $\mu^{k+1} = k\sqrt{k}$. Note that, the penalty parameter does not depend on the number of constraints although the pressure on infeasible solutions increases as k increases.

4.4 Hyperbolic Penalty

In the hyperbolic penalty method the sequence of subproblems is obtained by controlling two parameters in two different phases of the optimization process. In the first phase, the initial parameter λ increases, thus causing an increase in the penalty, P, to the points outside the feasible region and directing the search to the feasible region. This phase continues until a feasible point is obtained. From this point on, λ remains constant and the values of τ decrease sequentially.

In this context, [26] proposed the hyperbolic penalty function below, where only problems with inequality constraints are considered

$$P(x, \lambda, \tau) = \sum_{i=1}^{m} (\lambda_i g_i(x) + \sqrt{(\lambda_i)^2 g_i(x)^2 + (\tau_i)^2}) \tag{10}$$

with $\lambda \geq 0$, $\tau \geq 0$ are penalty parameters and $\lambda \to \infty$, $\tau \to 0$.

The penalty parameters are updated by

$$\begin{cases} \lambda_i^{k+1} = \min(\lambda^{\max}, \gamma_\lambda \lambda_i^k) \text{ and } \tau_i^{k+1} = \tau_i^k \text{ if } \max(0, g_i(x^{k+1})) > 0, \\ \tau_i^{k+1} = \max(\tau^{\min}, \gamma_\tau \tau_i^k) \text{ and } \lambda_i^{k+1} = \lambda_i^k, \text{ otherwise} \end{cases}$$

for $i = 1, \ldots, m$, where $\lambda^{\max}$ and $\tau^{\min}$ are upper and lower bounds respectively to the penalty parameters λ and τ. The goal is to define safeguards to prevent the subproblems from becoming ill-conditioned and more difficult to solve as λ increases or τ decreases.

We remark, the hyperbolic penalty function allows the use of optimization methods which use derivatives information, such as quasi-Newton method, to obtain the solution of the problem, since this is a continuously differentiable function.

4.5 Augmented Lagrangian Penalty

The augmented Lagrangian method uses the Lagrangian multipliers vector which reduces the possibility of generating ill-conditioned subproblems [2].

Combining the quadratic penalty function and the Lagrangian function, it is possible to obtain the penalty term of the augmented Lagrangian function [16],

$$P(x, \delta, \mu) = \frac{1}{2\mu} \sum_{i=1}^{m} (\max(0, \delta_i + \mu g_i(x))^2 - \delta_i^2) \tag{11}$$

where μ is the penalty parameter and δ is the Lagrange multipliers vector associated with the inequality constraints. The initial values set to the Lagrange multipliers and penalty parameter need not be large to occur a good approximation to the solution of the first subproblem of the sequence of subproblems to be solved [18].

An augmented Lagrangian penalty method is a multiplier-based method requiring [2]:

- The sequence $\{\delta^k\}$ must be bounded;
- The sequence of penalty values $\{\mu^k\}$ must satisfy $0 < \mu^k \leq \mu^{k+1}$, for all k and $\mu^k \to \infty$.

The updating scheme for the Lagrange multipliers δ_i associated with the constraints $g_i(x) \leq 0$, $i = 1, \ldots, m$ is given by

$$\delta_i^{k+1} = \max(0, \delta_i^k + \mu^k g_i(x^{k+1}))$$

and the penalty parameter is updated by

$$\mu^{k+1} = \min(\mu^{\max}, \alpha\mu^k)$$

where $\alpha > 1$ and $\mu^{\max}$ is an upper bound to the penalty parameter.

4.6 The Penalized DIRECT Algorithm

The general steps of the penalized DIRECT algorithm is described in Algorithm 1. At the end of the algorithm ϕ_{min} is the global optimal solution.

Algorithm 1. Penalized DIRECT algorithm

```
normalize the original domain Ω to the unit hypercube in ℝ^n with center c_1.
evaluate φ(c_1); φ_min = φ(c_1); set k = 1
evaluate φ(c_1 ± (1/3)e_i), 1 ≤ i ≤ n and divide hypercube
while stopping criterion is not reached do
    identify the set S of all potentially optimal hyperrectangles/cubes
    for all j ∈ S do
        identify the longest side(s) of hyperrectangle j
        update the penalty parameter
        evaluate φ at centers of new hyperrectangles and divide hyperrectangle j into
        smaller hyperrectangles
        update φ_min
    end for
    set k = k + 1
end while
```

The stopping criterion could be based on the maximum number of iterations and a maximum number of function evaluations.

5 Numerical Experiments

In this section, we report the numerical results obtained by running the Algorithm 1 based on the linear, quadratic, dynamic, hyperbolic and augmented Lagrangian penalty functions using six benchmark engineering design problems.

The implementation details of the proposed approach could be found in [25], as well as its application on a well-known and hard optimization problem from the chemical and bio-process engineering area.

Problems of practical interest are important for assessing the effectiveness of any algorithm. Thus, Table 1 contains a summary of the characteristics of the selected problems, where all of them have simple bounds and inequality constraints [6,15,21,22] and where n is the number of variables of the problem, m is the number of inequality constraints and $f(x^*)$ is the optimal solution known in the literature.

Table 1. Characteristics of the engineering design problems

Problem	n	Type of Objective Function	m	$f(x^*)$
Spring	3	quadratic	4	0.0126
Speed	7	nonlinear	11	2994.4991
Brake	4	quadratic	6	0.1274
Tubular	2	linear	2	26.5313
3-Bar	2	linear	3	263.896
4-Bar	4	linear	1	1400

5.1 Parameter Sensitivity Analysis

The choice of penalty parameters is a hard and very important task since the performance of DIRECT depends on the magnitude of the penalty parameters. On some problems, an extremely large penalty parameter is necessary for the algorithm to converge to a feasible point. However, a large penalty parameter is a critical issue for DIRECT, since could bias the algorithm away from hyper-rectangles near the boundary of feasibility [9].

Regarding this, it turns out to be very important to perform a sensitivity analysis in order to find the best values to provide to each problem. Depending on the penalty function and the problem, the values of the parameters could be different.

Below, it is graphically shown the influence of the tolerance ϵ, see (3), when solving different problems, for the local performance assessment of the algorithm. Figure 2 shows results for the l_1 penalty function when solving the 3-Bar problem. The bars of the right plot represent the number of function evaluations using the l_1 penalty and the circles represents the value of the objective

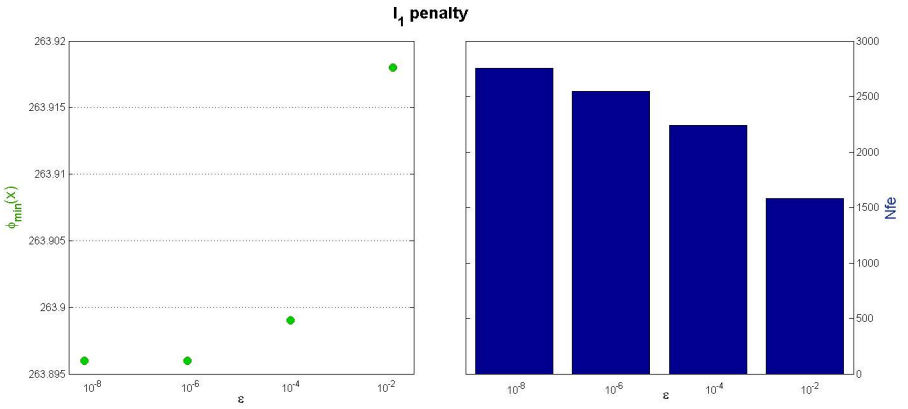

Fig. 2. Sensitivity analysis of parameter ϵ for the 3-Bar problem when using l_1 penalty function

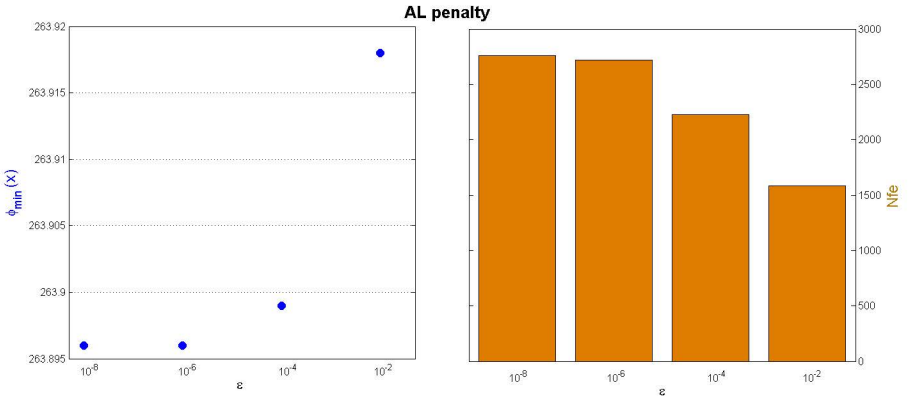

Fig. 3. Sensitivity analysis of parameter ϵ for the 3-Bar problem when using the augmented Lagrangian function

function attained for each ϵ value. Similar quantities obtained by the augmented Lagrangian function are shown in Fig. 3.

Figures 2-3 show that for lower values of ϵ, the value of the achieved penalty function value is, also, lower, although spending higher number of function evaluations. Thus, it is concluded that a reasonable value of ϵ, $\epsilon = 10^{-6}$, would be used for both penalties.

In addition to the parameter ϵ, which balances between local and global search, other parameters are associated with the penalty algorithm that affect its performance. For example, with the augmented Lagrangian penalty, initial values for the penalty parameter μ, for the multipliers vector δ, and for the penalty update α are required.

Table 2 shows the influence of these parameters in the augmented Lagrangian penalty algorithm, for 3-Bar problem. In the table, δ^1, refers to the initial value for all coordinates of the multipliers vector, 'Fopt' is the obtained solution and

Table 2. Penalty and multiplier values of augmented Lagrangian function for 3-Bar problem

μ^1	δ^1	α	Fopt	Nfe
1	1	2	141.9291*	1487
10	10	2	195.4431*	7659
100	100	2	231.6233*	5137
1000	1	1.5	263.8959	3567
1000	1	2	263.8959	2609
1000	10	2	263.8959	2985
1000	1000	2	263.8959	5011

* infeasible solution

'Nfe' denotes the number of function evaluations required to achieve a solution with a tolerance of $\epsilon = 10^{-6}$.

The solutions marked with * violate the constraints. By analyzing the Table 2 it appears that with initial higher values of μ, a better solution is obtained with fewer number of function evaluations. On the other hand, the algorithm seems to be more efficient with small initial values of Lagrange multipliers than with large ones.

5.2 Comparing Different Penalty Functions

To assess the performance of the penalty functions, the previously referred engineering design problems are used. The results are reported in Table 3 and include the number of iterations (Nit), the number of objective function evaluations (Nfe) and the optimal solution found (Fopt) with the l_1, quadratic, hyperbolic, dynamic and augmented Lagrangian penalty functions. The stopping criterion was based on three conditions: a relative tolerance to the known optimal solution of 0.001, a maximum of 250000 function evaluations or a maximum of 3000 iterations. We remark that we also stopped the algorithm when the penalty parameters stabilized and the solution could not improve, since DIRECT is a deterministic algorithm.

Table 3. Comparison results of the penalty functions for the engineering problems

		Spring	Speed	Brake	Tubular	3-Bar	4-Bar
l_1	Fopt	0.0144	2995.1358	0.1303	26.5313	263.8958	1400.0013
	Nfe	38659	249917	164389	113713	50191	2679
	Nit	890	2599	35	2785	1181	50
Quadratic	Fopt	0.0144	2995.1358	0.1274	26.5313	263.8958	1400.0001
	Nfe	157199	249979	14319	3841	48721	3497
	Nit	3000	2616	158	108	1251	57
Dynamic	Fopt	0.0225	2995.6809	0.1362	26.5425	263.8969	1400.0003
	Nfe	106443	224611	295167	179545	142283	3167
	Nit	3000	2074	39	3000	3000	54
Hyperbolic	Fopt	0.0225	2994.5967	0.1303	26.5316	263.8962	1400.0003
	Nfe	114339	200599	101887	53035	5327	1943
	Nit	3000	3000	3000	1172	199	55
Aug. Lagrangian	Fopt	0.0201	2995.7613	0.1362	26.5313	263.8958	1400.0001
	Nfe	89387	289381	450969	6449	2609	3615
	Nit	3000	3000	42	177	110	58

The results reported in Table 3 show that, in general, all functions show good results in all problems, *i.e.*, there is no evidence of a penalty function obtaining the best results for all problems. However, the quadratic penalty function achieved the best results for Brake, Tubular and 4-Bar problems. The hyperbolic,

dynamic and augmented Lagrangian penalty functions only achieved good results for one problem. A study based on the adaptive penalty function when solving these engineering problems could be found in [24].

During the experiments, we noticed, for some problems, the need to use a rather low penalty parameter, *i.e.*, the penalty function need not be so penalizing. For example, in the dynamic penalty the update of the μ parameter is based on the number of iterations and it was realized that after some iterations it was obtained a large penalty parameter, which does not benefit the output solution of the algorithm.

5.3 Behavior in Search Space

For a better understanding of the algorithm's performance, the search space visited by DIRECT combined with the penalty functions is presented below.

Since the 3-Bar problem is a bidimensional problem it is possible to represent the points in a Cartesian plane. Figure 4 contains a graphical representation of the search points of the algorithm in the space of the 3-Bar problem. The feasible points are marked with blue stars and the infeasible ones are marked with red points for all tested penalties. Figure 4a) plots the feasible and infeasible points visited by the DIRECT algorithm when combined with the augmented Lagrangian during the 110 iterations; Fig. 4b) shows the visited points by DIRECT with quadratic penalty function (1251 iterations); and Fig. 4c) depicts the points generated by DIRECT with the dynamic penalty (3000 iterations). The contours of the 3-Bar problem's functions (the objective and three constraints) are exhibited in Figure 4d) where the red star locates the global optimal solution.

We may conclude that DIRECT with the tested penalties performed an efficient and effective search around the area of the global optimal solution. However, the augmented Lagrangian function is able to achieve a better performance, visiting fewer points, since the plot presents a less dense cloud when compared with the quadratic and dynamic penalty functions.

6 Conclusion

This paper presents the performance of the DIRECT algorithm combined with quadratic, dynamic, hyperbolic and augmented Lagrangian penalty functions when solving six constrained engineering design problems.

In order to achieve the best solutions found by each algorithm, a sensitivity analysis to some parameters of the algorithm, namely the ϵ tolerance of DIRECT and penalty parameters of the tested penalty functions is carried out. We conclude that a consistent value for the penalty parameters appropriate for all tested penalty functions and for all problems is difficult to be found.

Generally, we may conclude that the obtained results with the different penalty functions showed competitive results when compared with the results from l_1 penalty function implemented in the DIRECT [9].

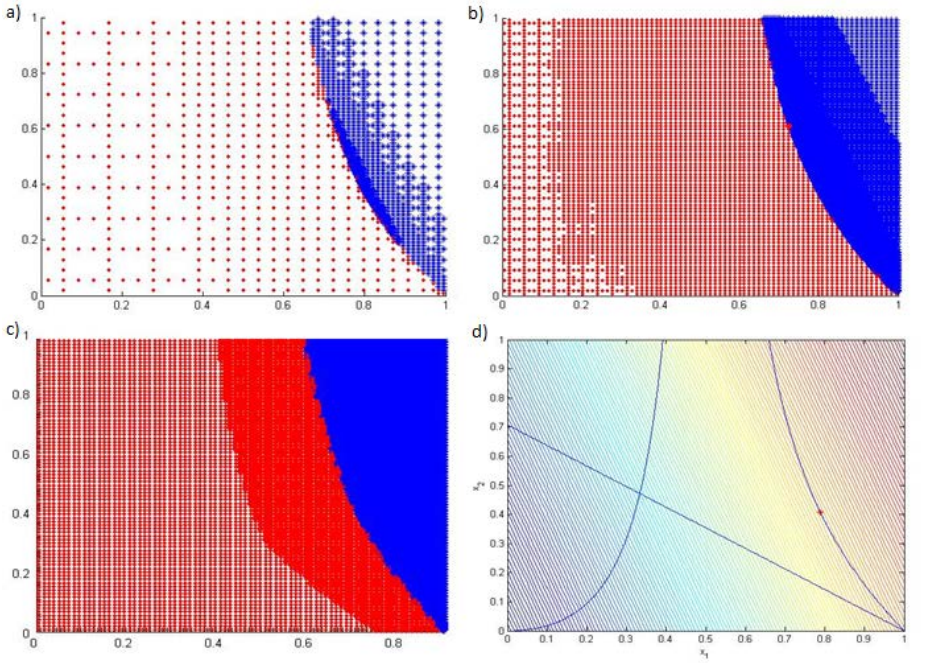

Fig. 4. Graphical representation of the points visited by DIRECT, for the 3-Bar problem. a) Augmented Lagrangian penalty function. b) Quadratic penalty. c) Dynamic penalty. d) Contours of the 3-Bar problem's functions.

Acknowledgment. The authors would like to thank the financial support from FEDER COMPETE (Operational Programme Thematic Factors of Competitiveness) and FCT (Portuguese Foundation for Science and Technology) Project FCOMP-01-0124-FEDER-022674.

References

1. Barbosa, H.J.C., Lemonge, A.C.C.: An adaptive penalty method for genetic algorithms in constrained optimization problems. In: Iba, H. (ed.) Frontiers in Evolutionary Robotics, pp. 9–34. I-Tech Education Publ., Austria (2008)
2. Bertsekas, D.P.: Constrained Optimization and Lagrange Multipliers Methods. Academic Press, New York (1982)
3. Carroll, C.W.: The created response surface technique for optimizing nonlinear restrained systems. Operations research 184, 9–169 (1961)
4. Carter, R.G., Gablonsky, J.M., Patrick, A., Kelley, C.T., Eslinger, O.J.: Algorithms for noisy problems in gas transmission pipeline optimization. Optimization and Engineering 2, 139–157 (2002)

5. Coello Coello, C.A.: Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. Computer Methods in Applied Mechanics and Engineering 191, 1245–1287 (2002)
6. Costa, M.F.P., Fernandes, E.M.G.P.: Efficient solving of engineering design problems by an interior point 3-D filter line search method. In: Simos, T.E., Psihoyios, G., Tsitouras, C. (eds.) AIP Conference Proceedings, vol. 1048, pp. 197–200 (2008)
7. Courant, R.: Variational methods for the solution of problems of equilibrium and vibrations. Bulletin of the American Mathematical Society 49, 1–23 (1943)
8. Fiacco, A.V., McCormick, G.P.: Extensions of SUMT for nonlinear programming: equality constraints and extrapolation. Management Science 12(11), 816–828 (1966)
9. Finkel, D.E.: Global optimization with the DIRECT algorithm. PhD thesis, North Carolina state University (2005)
10. Finkel, D.E., Kelley, C.T.: Convergence Analysis of the DIRECT Algorithm. North Carolina State University: Center for Research in Scientific Computation, Raleigh (2004)
11. Gablonsky, J.M.: Modifications of the DIRECT algorithm. PhD Thesis, North Carolina State University, Raleigh, North Carolina (2001)
12. Henderson, S.G., Biller, B., Hsieh, M.-H., Shortle, J., Tew, J.D., Barton, R.R.: Extension of the direct optimization algorithm for noisy functions. In: Proceedings of the 2007 Winter Simulation Conference (2007)
13. Joines, J., Houck, C.: On the use of nonstationary penalty functions to solve nonlinear constrained optimization problems with GAs. In: Proceedings of the First IEEE Congress on Evolutionary Computation, Orlando, FL, pp. 579–584 (1994)
14. Jones, D.R.: The DIRECT Global Optimization Algorithm. The Encyclopedia of Optimization. Kluwer Academic (1999)
15. Lee, K.S., Geem, Z.W.: A new meta-heuristic algorithm for continuous engineering optimization: Harmony search theory and practice. Computer Methods in Applied Mechanics and Engineering 194, 3902–3933 (2005)
16. Lewis, R., Torczon, V.: A Globally Convergent Augmented Lagrangian Pattern Search Algorithm for Optimization with General Constraints and Simple Bounds. SIAM Journal on Optimization 4(4), 1075–1089 (2012)
17. Liu, J.-L., Lin, J.-H.: Evolutionary computation of unconstrained and constrained problems using a novel momentum-type particle swarm optimization. Engineering Optimization 39, 287–305 (2007)
18. Nocedal, J., Wright, S.: Numerical Optimization. Springer Series in Operations Research (1999)
19. Jones, D.R., Perttunen, C.D., Stuckman, B.E.: Lipschitzian optimization without the lipschitz constant. Journal of Optimization Theory and Application 79(1), 157–181 (1993)
20. Petalas, Y.G., Parsopoulos, K.E., Vrahatis, M.N.: Memetic particle swarm optimization. Annals of Operations Research 156, 99–127 (2007)
21. Ray, T., Liew, K.M.: A swarm metaphor for multiobjective design optimization. Engineering Optimization 34(2), 141–153 (2002)
22. Rocha, A.M.A.C., Fernandes, E.M.G.P.: Hybridizing the electromagnetism-like algorithm with descent search for solving engineering design problems. International Journal of Computer Mathematics 86, 1932–1946 (2009)
23. Tessema, B., Yen, G.G.: A Self Adaptive Penalty Function Based Algorithm for Constrained Optimization. In: IEEE Congress on Evolutionary Computation, pp. 246–253 (2006)

24. Vilaça, R., Rocha, A.M.A.C.: An Adaptive Penalty Method for DIRECT Algorithm in Engineering Optimization. In: Simos, T.E., Psihoyios, G., Tsitouras, C., Anastassi, Z. (eds.) AIP Conference Proceedings, vol. 1479, pp. 826–829 (2012)
25. Vilaça, R.: Sofia Pinto: Extending the DIRECT algorithm to solve constrained nonlinear optimization problems: a case study. MSc Thesis, University of Minho (2012)
26. Xavier, A.E.: Hyperbolic penalty: a new method for nonlinear programming with inequalities. International Transactions in Operational Research 8, 659–671 (2001)