Hybridization of a Genetic Algorithm with a Pattern Search Augmented
Lagrangian Method

Lino Costa\textsuperscript{1,2}, Isabel A.C.P. Espírito Santo\textsuperscript{1,2}, Roman Denysiuk\textsuperscript{2} and Edite M.G.P. Fernandes\textsuperscript{1,2}

\textsuperscript{1}Department of Production and Systems
University of Minho, Campus de Gualtar, 4710-057 Braga, Portugal
\{lac,iapinho,emgpf\}@dps.uminho.pt
\textsuperscript{2}Algoritmi R&D Center, Portugal
denysiukr@gmail.com

Abstract
Genetic algorithms as most population based algorithms are good at identifying promising areas of the
search space (exploration), but less good at fine-tuning the approximation to the minimum (exploita-
tion). Conversely, local search algorithms like pattern search are good at improving the accuracy of
that approximation. Thus, a promising idea is combining local and global optimization techniques. We
propose a new hybrid genetic algorithm based on a local pattern search that relies on an augmented
Lagrangian function for constraint-handling. In this study, we test different hybridization schemes con-
cerning population-handling, as well as local search refinements for a better point. We use performance
profiles as proposed by Dolan and Moré in 2002 and a benchmark set of global problems to evaluate the
effect of the proposed hybrid algorithms. Population size effect on the algorithm is also assessed.

Keywords: Global optimization, Augmented Lagrangian, Genetic Algorithm, Pattern Search.

1. Introduction
In this paper, the following problem is under consideration:

\[
\begin{align*}
\minimize_{x \in \Omega} & \quad f(x) \\
\text{subject to} & \quad b_i(x) = 0, \quad i = 1, \ldots, m \\
& \quad g_j(x) \leq 0, \quad j = 1, \ldots, p
\end{align*}
\]

where \( x \) is an \( n \) dimensional vector and \( \Omega \subseteq \mathbb{R}^n \) (\( \Omega = \{ x \in \mathbb{R}^n : l \leq x \leq u \} \)), \( f(x) \) is the objective
function, \( b(x) = 0 \) are the equality constraints and \( g(x) \leq 0 \) are the inequality constraints. In this paper,
we aim to find a solution with the smallest objective function value. This type of constrained global
minimization problem has many applications in engineering. For example, in Robotics [29], Mechanical
Engineering [2] and in Materials Science [23].

There are a variety of methods to solve problems like (1). A common technique in global optimization
literature transforms the constrained problem into an unconstrained one, by adding a penalty term to
the objective function [5, 17]. This technique has some difficulties, namely the selection of appropriate
penalty coefficients for the problem at hand. To overcome this difficulty, some authors proposed to use
variable penalty schemes, e.g., dynamic scheme [16, 21, 24], self-adaptive scheme [9, 18] or implementing
superiority of feasible over infeasible solutions [7]. A comprehensive survey on constraint-handling tech-
niques can be found in [5]. Recently, some other constraint-handling techniques have been implemented
with stochastic algorithms. For instance, the Filter Simulated Annealing [14] that uses a filter-set based
procedure; the Gradient based Repair Genetic Algorithm [4] that derives information from the constraint
set for repairing infeasible solutions, and the Evolutionary Algorithm [25] that uses a multiobjective
approach coupled with stochastic ranking to deal with constraints.

The augmented Lagrangian is an interesting penalty function that avoids the side-effects associated
with ill-conditioning of simpler penalty and barrier functions. Lewis and Torczon [20] proposed this
technique, where a pattern search algorithm is used to solve the unconstrained problem, based on the
augmented Lagrangian presented by Conn \textit{et al.} [6]. Despite the inequality constraints are not considered
in these works, we extended the therein proposed augmented Lagrangian to the inequality constraints,
following the ideas presented in [3, 10].

Recently, there has been work on hybridization of population based algorithms with local search as, for
instance, the Memetic Particle Swarm Optimization algorithm by Petalas \textit{et al.} [24], the Particle Swarm
Optimization algorithm proposed by Zahra and Hu [28], the Hybrid Evolutionary and Gradient Search by Tahk et al. [26] and the Heuristic Pattern Search Simulated Annealing by Hedar and Fukushima [13]. However, to the best of our knowledge, hybridization of augmented Lagrangian, genetic algorithms and local search has not been attempted. Thus, here we propose a Hybrid Genetic Pattern Search Augmented Lagrangian (HGPSAL) algorithm that hybridizes a genetic algorithm with a derivative-free pattern search method to refine the best solution found by the genetic search. We propose four hybridization schemes concerning population-handling. We also present a study on the population size effect. Equality and inequality constraints of the problem are treated by an augmented Lagrangian framework.

This paper is organized as follows. Section 2 describes the HGPSAL algorithm. It introduces the augmented Lagrangian technique for constraint-handling and provides details concerning the genetic algorithm and the pattern search method. In Section 3, we present and analyze the experimental results on 24 constrained test problems. Finally, we conclude the paper with a few remarks and future work in Section 4.

2. Hybrid Genetic Pattern Search Augmented Lagrangian Algorithm

A paradigm based on a hybridization of augmented Lagrangians with genetic algorithms has not been attempted. Based on the good results obtained by algorithms relying on augmented Lagrangian penalties, this seems a very promising framework.

On the other hand, hybridization of global and local optimizers may provide a more effective tradeoff between exploitation and exploration of the search space. It is well-known that overall successful and efficient general solvers do not exist. Moreover, stochastic population based algorithms like genetic algorithms [12] are good at identifying promising areas of the search space (exploration), but less good at fine-tuning approximations to the minimum (exploitation). Conversely, local search algorithms like pattern search are good at improving approximations to the minimum. Thus, a promising idea that may allow to reduce the total number of function evaluations is combining local and global optimization techniques.

2.1. Augmented Lagrangian technique

An augmented Lagrangian technique solves a sequence of very simple subproblems where the objective function penalizes all or some of the constraints violation. This objective function is an augmented Lagrangian that depends on a penalty parameter, as well as on the multiplier vectors, and works like penalty functions. Using the ideas in [3, 6, 20], the herein implemented augmented Lagrangian function is

\[
\Phi(x; \lambda, \delta, \mu) = f(x) + \lambda^T b(x) + \frac{1}{2\mu} \|b(x)\|^2_2 + \frac{\mu}{2} \sum_{i=1}^{p} \left( \max \left\{ 0, \delta_i + \frac{g_i(x)}{\mu} \right\} - \delta_i^2 \right)
\]

where \(\mu\) is a positive penalty parameter, \(\lambda = (\lambda_1, \ldots, \lambda_m)^T\) and \(\delta = (\delta_1, \ldots, \delta_p)^T\) are the Lagrange multiplier vectors associated with the equality and inequality constraints respectively. Function \(\Phi\) aims to penalize solutions that violate the equality and inequality constraints only. The simple bounds \(l \leq x \leq u\) are not integrated into the penalty terms. Hence, the corresponding subproblem is formulated as:

\[
\min_{x \in \Omega} \Phi(x; \lambda^j, \delta^j, \mu^j).
\]  

To simplify the notation, from now on we use \(\Phi^J(x)\) instead of \(\Phi(x; \lambda^j, \delta^j, \mu^j)\), \(||.||\) represents the Euclidean norm and \(v_x = \max \{0, v\}\). The solution of (2) for each set of fixed \(\lambda^j, \delta^j\) and \(\mu^j\), gives an approximation to the solution of (1). Denote this approximation by \(x^{j+1}\). Here the index \(j\) is the counter of the outer iterative process. To ensure global convergence, the penalty parameter must be driven to zero as the Lagrange multipliers estimates must have reasonable behavior. Thus, as \(j \to \infty\) and \(\mu^j \to 0\), the solutions of the subproblems (2) converge to the solution of (1). We refer to [3, 10] for details. The Lagrange multipliers \(\lambda^j\) and \(\delta^j\) are estimated in this iterative process according to proper first-order updating formulae. The traditional augmented Lagrangian methods are locally convergent if the subproblems (2) are solved according to a certain tolerance, herein reported by \(\varepsilon^j\), for sufficiently small values of the penalty parameter. The general augmented Lagrangian algorithm for solving problem (1) is the following.

Algorithm 1 HGPSAL Algorithm
Given $\lambda_{\min} < \lambda_{\max}, \delta_{\max} > 0, 0 < \gamma < 1, \mu_{\min} \ll 1, \eta^* \ll 1, \epsilon^* \ll 1, \lambda_i^0 \in [\lambda_{\min}, \lambda_{\max}], i = 1, \ldots, m, \\
\delta_i^0 \in (0, \delta_{\max}], i = 1, \ldots, p, \mu^0 > 0, \eta^0 > 0, \pi < 1$;

Step 1. Compute $\epsilon^0$;

Step 2. Randomly generate a point $x^0 \in \Omega$ and set $j = 0$;

Step 3. While the stopping criterion is not met do

For a certain tolerance $\epsilon^j$, find an approximate minimizer $x^{j+1}$ to the subproblem (2) using Algorithm 2;

Update $\delta_i^{j+1} = \max \left\{ 0, \min \left\{ \delta_i^j + \frac{2(x_i^{j+1})}{\mu^j}, \delta_{\max} \right\} \right\}$, $i = 1, \ldots, p$;

If $(E(x^{j+1}, \delta^{j+1}) \leq \eta^j)$ then

Set $\lambda_i^{j+1} = \lambda_i^j + \max \left\{ \lambda_{\min}, \min \left\{ \frac{\mu^j(x_i^{j+1})}{\lambda_i^j}, \lambda_{\max} \right\} \right\}$, $i = 1, \ldots, m$;

Update $\mu^{j+1} = \mu^j$;

else

Set $\lambda_i^{j+1} = \lambda_i^j$;

Update $\mu^{j+1} = \max \{ \mu_{\min}, \gamma \mu^j \}$;

EndIf

Update $\eta^{j+1} = \pi \eta^j$;

Compute $\epsilon^{j+1}$;

Set $j = j + 1$.

The stopping criterion is based on an error function, $E(x, \delta)$, that requires rather small feasibility and complementarity levels from the computed approximation, where

$$E(x, \delta) = \max \left\{ \frac{\|b(x)\|_{\infty}}{1 + \|x\|} \cdot \frac{\max \left\{ \|g(x)\|_{\infty}, \max \delta_i |g_i(x)| \right\}}{1 + \|\delta\|} \right\}.$$

If the algorithm finds an approximation $x$ such that

$$E(x, \delta) \leq \eta^* \text{ and } \epsilon \leq \epsilon^*$$

for small and positive constants $\eta^*$ and $\epsilon^*$, then the algorithm stops; otherwise, the algorithm runs until a maximum of (outer) iterations, $j_{\max}$, is reached. The tolerance $\epsilon$ varies with the Lagrange multipliers and penalty parameter values according to

$$\epsilon = \tau \left( 1 + \|\lambda\| + \|\delta\| + (\mu)^{-1} \right)^{-1}, \quad \tau > 0.$$

Note that a decreasing sequence of $\mu$ values will yield a decreasing sequence of $\epsilon$ values forcing more and more accurate solutions to the subproblems (2). As shown in the HGPSAL algorithm, the updating of the Lagrange multipliers rely on safeguards to maintain the multiplier vectors bounded throughout the process. The sequence of penalty parameters should also be maintained far away from zero so that solving the subproblem (2) is an easy task. The main differences between augmented Lagrangian algorithms are located on the framework used to find an approximate global solution, for a defined tolerance $\epsilon^j$, to the subproblem (2). The herein proposed technique for solving (2) uses a population based algorithm, known as genetic algorithm, followed by a local search procedure. The general form for the bound constrained algorithm implemented in this paper is shown below.

Algorithm 2 Hybrid Genetic Pattern Search Bound Constrained Algorithm

Given $x^j \in \Omega$;

Step 1. Find $Y^j \leftarrow GA(x^j)$, using the genetic algorithm presented in Subsection 2.2;

Step 2. Find $X^{j+1} \leftarrow HJ(Y^j)$, using the Hooke and Jeeves version of the pattern search algorithm described in Subsection 2.3;

Step 3. $x^{j+1} \leftarrow$ compute the best point of $X^{j+1}$. 
We remark that since the genetic algorithm is a population based method, \( Y_j \) is a set of \( y^j \) points, with best fitness found by the algorithm. Details concerning each step of the algorithm are presented below.

2.2. Genetic Algorithm

A Genetic Algorithm (GA) is a population based algorithm that uses techniques inspired by evolutionary biology such as inheritance, mutation, selection, and crossover [12]. Thus, unlike conventional algorithms, GAs start from a population of points \( P \) of size \( s \). In spite of the traditional binary representation used by GAs, in our implementation, a real representation was used since we are leading with continuous problems. Therefore, each point of the population \( z^{(l)} \), for \( l = 1, \ldots, s \), is an \( n \) dimensional vector.

A fitness function is defined as evaluation function in order to compare the points of the population and to apply a stochastic selection that guarantees that better points are more likely to be selected. The fitness function corresponds to the objective function of the subproblem (2), i.e., \( \Phi^j(x) \). A tournament selection was considered, i.e., tournaments are played between two points and the best point (with lower fitness value) is chosen for the pool.

New points in the search space are generated by the application of genetic operators (crossover and mutation) to the selected points from population. Elitism was implemented by maintaining, during the search, a given number \( c \), of the best points in the population.

Crossover combines two points in order to generate new ones. A Simulated Binary Crossover (SBX) [8] that simulates the working principle of single-point crossover operator for binary strings was implemented. Two points, \( z^{(1)} \) and \( z^{(2)} \), are randomly selected from the pool and, with probability \( p_c \), two new points, \( w^{(1)} \) and \( w^{(2)} \) are generated according to

\[
\begin{align*}
    w^{(1)}_i &= 0.5 \left( (1 + \beta_i)z^{(1)}_i + (1 - \beta_i)z^{(2)}_i \right) \\
    w^{(2)}_i &= 0.5 \left( (1 - \beta_i)z^{(1)}_i + (1 + \beta_i)z^{(2)}_i \right)
\end{align*}
\]

for \( i = 1, \ldots, n \). The values of \( \beta_i \) are obtained from the following distribution:

\[
\beta_i = \begin{cases} 
    \frac{2r_i}{\eta_m - 1} & \text{if } r_i \leq 0.5 \\
    \frac{1}{\eta_m - 2(1 - r_i)} & \text{if } r_i > 0.5
\end{cases}
\]

where \( r_i \sim U(0,1) \) and \( \eta_m > 0 \) is an external parameter of the distribution. This procedure is repeated until the number of generated points equals the number of points in the pool.

A Polynomial Mutation is applied, with a probability \( p_m \), to the points produced by the crossover operator. Mutation introduces diversity in the population since crossover, exclusively, could not assure the exploration of new regions of the search space. This operator guarantees that the probability of creating a new point \( t^{(l)} \) closer to the previous one \( w^{(l)} \) (\( l = 1, \ldots, s \)) is more than the probability of creating one away from it. It can be expressed by:

\[
t^{(l)}_i = w^{(l)}_i + (u_i - l_i)\eta_i
\]

for \( i = 1, \ldots, n \). The values of \( \eta_i \) are given by:

\[
\eta_i = \begin{cases} 
    \frac{2r_i}{\eta_m - 1} - 1 & \text{if } r_i < 0.5 \\
    1 - (2(1 - r_i))^{\frac{1}{\eta_m - 1}} & \text{if } r_i \geq 0.5
\end{cases}
\]

where \( r_i \sim U(0,1) \) and \( \eta_m > 0 \) is an external parameter of the distribution. The GA proceeds as the following algorithm.

Algorithm 3 Genetic Algorithm

Given \( x^j, \; c, \; s, \; p_c, \; p_m \);

Step 1. Set \( z^{(1)} = x^j \) and randomly generate \( z^{(l)} \in \Omega \), for \( l = 2, \ldots, s \) (Initialization of \( P \));

Step 2. Set \( k = 0 \);
Step 3. While the stopping criterion is not met do
\[
\begin{align*}
\text{Compute } \Phi^j(z^{(l)}), & \quad \text{for } l = 1, \ldots, s \; \text{(Fitness Evaluation)}; \\
\text{Select by tournaments } s - e \; \text{points from } P \; \text{(Selection)}; \\
\text{Apply SBX crossover with probability } p_c \; \text{(Crossover)}; \\
\text{Apply polynomial mutation with probability } p_m \; \text{(Mutation)}; \\
\text{Replace the worst } s - e \; \text{points of } P \; \text{(Elitism)}; \\
\text{Set } y^j = z^j_{\text{best}}; \\
\text{Set } k = k + 1.
\end{align*}
\]

This procedure ends when \(|\Phi^j(z^k_{\text{best}}) - \Phi^j(z^k_{\text{best} - k})| \leq \varepsilon^j \) where \(\Phi^j(z^k_{\text{best}})\) is the fitness value of the best point in population, at iteration \(k\), and \(k_\Delta\) is a parameter that defines a periodicity for testing the criterion. However, if the previous criterion is not satisfied in \(k_{\text{max}}\) iterations the procedure is terminated and the best point in the population is returned.

2.3. The Hooke and Jeeves Pattern Search

A pattern search method is a derivative-free method that performs, at each iteration \(k\), a series of exploratory moves around a current approximation, \(z^k\), in order to find a new approximation \(z^{k+1} = z^k + \Delta^k s^k\), with a lower fitness value. We use \(k\) for the iteration counter of this inner iterative process. For \(k = 0\), the initial approximation to begin the search is \(z^0 = y^j\) (see Algorithm 2), applied to each point in the set \(Y^j\). The scalar \(\Delta^k\) represents the step length and the vector \(s^k\) determines the direction of the step. The exploratory moves to produce \(\Delta^k s^k\) and the updating of \(\Delta^k\) and \(s^k\) define a particular pattern search method and their choices are crucial to the success of the algorithm. When \(\Phi^j(z^{k+1}) < \Phi^j(z^k)\) then the iteration is considered successful; otherwise it is unsuccessful. When an iteration is successful, the step length is not modified, while in an unsuccessful iteration \(\Delta^k\) is reduced. See for example [19, 27].

In our algorithm, \(\Delta^k s^k\) is computed by the Hooke and Jeeves (HJ) search method [15]. This algorithm differs from the traditional coordinate search since it performs two types of moves: the exploratory move and the pattern move. An exploratory move is a coordinate search - a search along the coordinate axes - around a selected approximation, using a step length of \(\Delta^k\). A pattern move is a promising direction that is defined by \(z^k - z^{k-1}\) when the previous iteration was successful and \(z^k\) was accepted as the new approximation. A new trial approximation is then defined as \(z^k + (z^k - z^{k-1})\) and an exploratory move is then carried out around this trial point. If this search is successful, the new approximation is accepted as \(z^{k+1}\). We refer to [15, 19] for details. This HJ iterative procedure terminates, providing a new set of approximations \(X^{j+1}\) to the problem (1), \(x^{j+1} \leftarrow z^{k+1}\), when the following stopping condition is satisfied, \(\Delta^k \leq \varepsilon^j\). However, if this condition can not be satisfied in \(k_{\text{max}}\) iterations, then the procedure is stopped with the last available approximation.

2.4. Forcing feasibility

The inner iterative process must return a feasible approximation (recall (2)). While using the genetic algorithm and the Hooke and Jeeves version of the pattern search, any computed approximation \(x\) that does not lie in the set \(\Omega\) is projected component by component (for all \(i, \ldots, n\)) as follows:

\[
x_{i} = \begin{cases} 
    l_i & \text{if } x_{i} < l_{i} \\
    x_{i} & \text{if } l_{i} \leq x_{i} \leq u_{i} \\
    u_i & \text{if } x_{i} > u_{i}
\end{cases}
\]

3. Numerical Results

3.1. Test problems

In this study, in order to evaluate the performance of the HGPSAL algorithm 24 benchmark problems coded in MATLAB were considered. This set of difficult constrained problems with very distinct properties was taken from [17, 22]. The characteristics of these problems are summarized in Table 1 that indicates the type of objective function, the number of decision variables (\(n\)), the number of inequality constraints (\(p\)), the number of equality constraints (\(m\)), the number of active constraints at the optimum (\(n_{\text{act}}\)) and the optimal value known (\(f_{\text{global}}\)).
Table 1: Test problems.

<table>
<thead>
<tr>
<th>Prob.</th>
<th>Type of $f(x)$</th>
<th>n</th>
<th>p</th>
<th>m</th>
<th>$n_{act}$</th>
<th>$f_{global}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>g01</td>
<td>min quadratic</td>
<td>13</td>
<td>9</td>
<td>0</td>
<td>6</td>
<td>-15.00000</td>
</tr>
<tr>
<td>g02</td>
<td>max nonlinear</td>
<td>20</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0.803619</td>
</tr>
<tr>
<td>g03</td>
<td>max polynomial</td>
<td>10</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1.000000</td>
</tr>
<tr>
<td>g04</td>
<td>min quadratic</td>
<td>5</td>
<td>6</td>
<td>0</td>
<td>2</td>
<td>-30665.54</td>
</tr>
<tr>
<td>g05</td>
<td>min cubic</td>
<td>4</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>5126.498</td>
</tr>
<tr>
<td>g06</td>
<td>min cubic</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>-6061.814</td>
</tr>
<tr>
<td>g07</td>
<td>min quadratic</td>
<td>10</td>
<td>8</td>
<td>0</td>
<td>6</td>
<td>24.30621</td>
</tr>
<tr>
<td>g08</td>
<td>max nonlinear</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0.095825</td>
</tr>
<tr>
<td>g09</td>
<td>min polynomial</td>
<td>7</td>
<td>4</td>
<td>0</td>
<td>2</td>
<td>680.6301</td>
</tr>
<tr>
<td>g10</td>
<td>min linear</td>
<td>8</td>
<td>6</td>
<td>0</td>
<td>6</td>
<td>7049.248</td>
</tr>
<tr>
<td>g11</td>
<td>min quadratic</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0.750000</td>
</tr>
<tr>
<td>g12</td>
<td>max quadratic</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1.000000</td>
</tr>
<tr>
<td>g13</td>
<td>min nonlinear</td>
<td>5</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>0.053945</td>
</tr>
<tr>
<td>g14</td>
<td>min nonlinear</td>
<td>10</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>-47.76488</td>
</tr>
<tr>
<td>g15</td>
<td>max quadratic</td>
<td>3</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>-961.7150</td>
</tr>
<tr>
<td>g16</td>
<td>min nonlinear</td>
<td>5</td>
<td>38</td>
<td>0</td>
<td>4</td>
<td>-1.905155</td>
</tr>
<tr>
<td>g17</td>
<td>min nonlinear</td>
<td>6</td>
<td>0</td>
<td>4</td>
<td>4</td>
<td>8853.539</td>
</tr>
<tr>
<td>g18</td>
<td>max quadratic</td>
<td>9</td>
<td>13</td>
<td>0</td>
<td>6</td>
<td>0.866025</td>
</tr>
<tr>
<td>g19</td>
<td>min nonlinear</td>
<td>15</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>32.65559</td>
</tr>
<tr>
<td>g20</td>
<td>min linear</td>
<td>24</td>
<td>6</td>
<td>14</td>
<td>16</td>
<td>0.204979</td>
</tr>
<tr>
<td>g21</td>
<td>min linear</td>
<td>7</td>
<td>1</td>
<td>5</td>
<td>6</td>
<td>193.7245</td>
</tr>
<tr>
<td>g22</td>
<td>min linear</td>
<td>22</td>
<td>1</td>
<td>19</td>
<td>19</td>
<td>236.4309</td>
</tr>
<tr>
<td>g23</td>
<td>min linear</td>
<td>9</td>
<td>2</td>
<td>4</td>
<td>6</td>
<td>-400.0551</td>
</tr>
<tr>
<td>g24</td>
<td>max linear</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>5.508013</td>
</tr>
</tbody>
</table>

3.2. Performance profile

To compare the performance of the pattern search type algorithms we use the performance profiles as described in Dolan and Moré’s paper [1]. A brief description follows. Let $\mathcal{P}$ and $\mathcal{S}$ be the set of problems and the set of solvers in comparison, respectively, and let $m_{p,s}$ be the performance metric required to solve problem $p \in \mathcal{P}$ by solver $s \in \mathcal{S}$. The comparison is based on the performance ratios defined by

\[
r_{p,s} = \begin{cases} 
1 + \frac{m_{p,s} - \min \{m_{p,s} : s \in \mathcal{S}\}}{\min \{m_{p,s} : s \in \mathcal{S}\}}, & \text{if } \min \{m_{p,s} : s \in \mathcal{S}\} < 10^{-4} \\
\frac{m_{p,s} - \min \{m_{p,s} : s \in \mathcal{S}\}}{\min \{m_{p,s} : s \in \mathcal{S}\}}, & \text{otherwise}
\end{cases}
\]

and the overall assessment of the performance of a particular solver $s$ is given by

\[
\rho_s(\tau) = \frac{\text{no. of problems where } r_{p,s} \leq \tau}{\text{total no. of problems}}.
\]

Thus, $\rho_s(\tau)$ gives the probability (for $s \in \mathcal{S}$) that $r_{p,s}$ is within a factor $\tau \in \mathbb{R}$ of the best possible ratio. The value of $\rho_s(1)$ gives the probability that the solver $s$ will win over the others in the set. However, for large values of $\tau$, the $\rho_s(\tau)$ measures the solver robustness. Overall the highest the $\rho$ values the better the solver is.

3.3. Hybridization schemes

We propose and test four different hybridization schemes. They differ in the number of points of the population that are selected to be improved in the local search. The tested alternatives are: just one point, the best point of the population; the 10% best points; the 25% best points; and the 50% best points. For simplicity, we denote the hybridization schemes as follows:
• version 1 that improves the best population point found by GA with HJ;
• version 2 that improves the best 10\% population points found by GA with HJ;
• version 3 that improves the best 25\% population points found by GA with HJ;
• version 4 that improves the best 50\% population points found by GA with HJ.

3.4. Algorithm parameters

All parameters of the HGPSAL algorithm were kept constant for all problems. No effort was made in finding the best parameter setting for each problem. Table 2 shows the parameters of the augmented Lagrangian used in all experiments. In Table 3 the genetic algorithm parameters are listed. The maximum number of iterations $k_{\text{max}}$ for Hooke and Jeeves pattern search was set on 200 iterations.

Table 2: Augmented Lagrangian parameters.

<table>
<thead>
<tr>
<th>$\lambda_{\text{min}}$</th>
<th>$\lambda_{\text{max}}$</th>
<th>$\delta_{\text{max}}$</th>
<th>$\mu^0$</th>
<th>$\mu_{\text{min}}$</th>
<th>$\nu^*$</th>
<th>$\nu^*$</th>
<th>$\lambda_0$</th>
<th>$\psi_i$</th>
<th>$\delta_{i, \psi_i}^0$</th>
<th>$\eta^0$</th>
<th>$j_{\text{max}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$-10^{-3}$</td>
<td>$10^{12}$</td>
<td>$10^{12}$</td>
<td>1</td>
<td>$10^{-12}$</td>
<td>0.5</td>
<td>$10^{-12}$</td>
<td>$10^{-8}$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>$300$</td>
</tr>
</tbody>
</table>

Table 3: Genetic Algorithm parameters.

<table>
<thead>
<tr>
<th>$k_{\text{max}}$</th>
<th>$s$</th>
<th>$\epsilon$</th>
<th>$p_c$</th>
<th>$p_m$</th>
<th>$\eta_m$</th>
<th>$\epsilon$</th>
<th>$\Delta_g$</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>20</td>
<td>2</td>
<td>0.9</td>
<td>20</td>
<td>1/n</td>
<td>20</td>
<td>10^{-8}</td>
</tr>
</tbody>
</table>

3.5. Discussion

Figure 1 allows to analyze the effect of the population size $s$ on the algorithm performance. The profiles in Figure 1a, based on the metric $f_{\text{avg}}$, show the central tendency of the best solutions found over the 10 runs, displaying that $s = \min(200, 20n)$ outperforms the other executions with different (and smaller) population sizes in comparison. The execution with this size of population gives the best average solution in about 84\% of the tested problems, and it is followed by the execution with $s = \min(200, 10n)$ which attains the best average solution in 75\% of the problems. The execution with the smallest population $s = 20$ is able to reach the least average function value in about 62\% of the problems. The conclusion that we may draw from Figure 1a is that the larger the population size the better the accuracy of the average solution is. However, as expected, the size of the population affects the computational effort of the algorithm for solving the problem. Figure 1b displays the performance profiles of the average number of function evaluations $n_{\text{eval}_{\text{avg}}}$ computed over the 10 runs. A large population size $s$ requires consequently a large number of function evaluations. The difference here in the percentage of solved problems with least number of function evaluations between the executions $s = 20$ (with 70\% of better values) and $s = \min(200, 20n)$ (with 5\% of better values) is around 65\%. Thus, a compromise seems crucial. A population size of $s = \min(200, 10n)$ seems appropriate for the remaining numerical experiments.

In order to test the four proposed hybridization schemes, we draw performance profiles on the central tendency of the best solutions, $f_{\text{avg}}$, and the average number of function evaluations, $n_{\text{eval}_{\text{avg}}}$, over 10 runs.

Attending on Figure 2a, we can observe that version 3 outperforms the other versions, attaining the best average solution in more than 90\% of the problems. We remark that this version improves the best 25\% best points of the population obtained by the GA algorithm with the HJ local search. The worst performance is obtained by version 4, meaning that it is self defeating to use an excessive number of points of the population to improve by local search.

As to the $n_{\text{eval}_{\text{avg}}}$ (Figure 2b), version 3 obtains the lower number of average function evaluations in about 33\% of the problems. Despite version 2 and version 4 are less expensive in this item (42\% of efficiency), it seems a good compromise to choose version 3 as the one that gives the best results.
Figure 1: Performance profile on favg and nfevavg for version 1

Figure 2: Performance profile on favg and nfevavg for s = min(200, 10n)

Figure 3 shows the boxplot representations of the comparative performance of the HGPSAL versions with different population sizes for the problem g02. It should be noted that this problem is difficult and, in general, the solvers fail to achieve an accurate approximation to the global optimum. The figure indicates the distribution of the approximations obtained in 10 runs (the worst, upper quartile, median, lower quartile, and best approximations are depicted). In Figures 3a and 3b, we can observe that the best performance is obtained with the version 4, which presents the best median. However, for larger population sizes (see Figures 3c and 3d), version 3 outperforms version 4 on this particular problem.

4. Conclusions

In this paper, we proposed four hybridization schemes for constrained global optimization that combines the augmented Lagrangian technique for handling constraints with a genetic algorithm as global optimizer and a pattern search as local optimizer. We performed a comparative analysis on the population size effect, as well as on the population-handling on the hybridization schemes.

We show that a population size of $s = \min(200, 10n)$ seems to be the more appropriate for the generality of the problems.

In general, using an excessive number of points of the population to improve by HJ leads to poor results.
Figure 3: Boxplots for different populations sizes (problem g02)

As future work, we intend to perform comparisons with other stochastic approaches and solve other benchmark problems, to improve the integration of global and local search, as well as tuning the parameters of the algorithm. We also intend to test other global and local strategies.

References