A genetic algorithm framework for multilocal optimization

A. Ismael F. Vaz
Lino Costa

Production and Systems Department,
School of engineering,
University of Minho, Gualtar Campus,
4710 - 057 Braga, Portugal;
Email: {lac,aivaz}@dps.uminho.pt

EURO XXIII, Bonn, Germany

5-8 July 2009
1 Introduction

2 Genetic (Evolutionary) algorithm framework

3 Genetic algorithm framework for multi-local optimization

4 Application to SIP

5 Numerical results

6 Conclusions
Outline

1. Introduction
2. Genetic (Evolutionary) algorithm framework
3. Genetic algorithm framework for multi-local optimization
4. Application to SIP
5. Numerical results
6. Conclusions
Outline

1. Introduction
2. Genetic (Evolutionary) algorithm framework
3. Genetic algorithm framework for multi-local optimization
4. Application to SIP
5. Numerical results
6. Conclusions
1. Introduction

2. Genetic (Evolutionary) algorithm framework

3. Genetic algorithm framework for multi-local optimization

4. Application to SIP

5. Numerical results

6. Conclusions
Outline

1. Introduction
2. Genetic (Evolutionary) algorithm framework
3. Genetic algorithm framework for multi-local optimization
4. Application to SIP
5. Numerical results
6. Conclusions
Outline

1. Introduction
2. Genetic (Evolutionary) algorithm framework
3. Genetic algorithm framework for multi-local optimization
4. Application to SIP
5. Numerical results
6. Conclusions
1. Introduction
2. Genetic (Evolutionary) algorithm framework
3. Genetic algorithm framework for multi-local optimization
4. Application to SIP
5. Numerical results
6. Conclusions
Addressed problem

The following problem is under consideration

\[
\min_{x \in \Omega \subset \mathbb{R}^n} f(x)
\]  

(1)

where \( f(x) \) is the objective function, \( x = (x_1, \ldots, x_n)^T \) is an \( n \) dimensional vector and \( \Omega \subset \mathbb{R}^n \) is the feasible set, herein assumed to be a cartesian product of intervals with finite bounds (\( \Omega = [\alpha_1, \beta_1] \times \cdots \times [\alpha_n, \beta_n] \)).

Multi-local

We aim to compute approximations to all the local optima for problem (1).

Assumptions

\( f(x) \) is assumed to be twice continuous differentiable and cheap to evaluate (i.e. the number of objective function evaluations is not a concern).
Addressed problem

The following problem is under consideration

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

(1)

where $f(x)$ is the objective function, $x = (x_1, \ldots, x_n)^T$ is an $n$ dimensional vector and $\Omega \subseteq \mathbb{R}^n$ is the feasible set, herein assumed to be a cartesian product of intervals with finite bounds ($\Omega = [\alpha_1, \beta_1] \times \cdots \times [\alpha_n, \beta_n]$).

Multi-local

We aim to compute approximations to all the local optima for problem (1).

Assumptions

$f(x)$ is assumed to be twice continuous differentiable and cheap to evaluate (i.e. the number of objective function evaluations is not a concern).
Addressed problem

The following problem is under consideration

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

where $f(x)$ is the objective function, $x = (x_1, \ldots, x_n)^T$ is an $n$ dimensional vector and $\Omega \subset \mathbb{R}^n$ is the feasible set, herein assumed to be a cartesian product of intervals with finite bounds ($\Omega = [\alpha_1, \beta_1] \times \cdots \times [\alpha_n, \beta_n]$).

Multi-local

We aim to compute approximations to all the local optima for problem (1).

Assumptions

$f(x)$ is assumed to be twice continuous differentiable and cheap to evaluate (i.e., the number of objective function evaluations is not a concern).
Outline

1. Introduction

2. Genetic (Evolutionary) algorithm framework

3. Genetic algorithm framework for multi-local optimization

4. Application to SIP

5. Numerical results

6. Conclusions
Algorithm framework

Genetic algorithms

A genetic algorithm is a population based algorithm that uses techniques inspired by evolutionary biology such as inheritance, mutation, selection, and crossover.

A GA framework

1. Randomly initialize an initial population (real-parameter representation).
2. Compute a set of Parents from the elite population using a fitness function (tournament selection).
3. Compute a set of Offsprings obtained from the set of Parents using the crossover and mutation operators.
4. Verify the stopping criteria. If not met then goto step 2.
Genetic (Evolutionary) algorithm framework

Algorithm framework

Genetic algorithms

A genetic algorithm is a population based algorithm that uses techniques inspired by evolutionary biology such as inheritance, mutation, selection, and crossover.

A GA framework

1. Randomly initialize an initial population (real-parameter representation).
2. Compute a set of Parents from the elite population using a fitness function (tournament selection).
3. Compute a set of Offsprings obtained from the set of Parents using the crossover and mutation operators.
4. Verify the stopping criteria. If not met then goto step 2.
Genetic algorithms

A genetic algorithm is a population based algorithm that uses techniques inspired by evolutionary biology such as inheritance, mutation, selection, and crossover.

A GA framework

1. Randomly initialize an initial population (real-parameter representation).
2. Compute a set of Parents from the elite population using a fitness function (tournament selection).
3. Compute a set of Offsprings obtained from the set of Parents using the crossover and mutation operators.
4. Verify the stopping criteria. If not met then goto step 2.
Genetic algorithms

A genetic algorithm is a population based algorithm that uses techniques inspired by evolutionary biology such as inheritance, mutation, selection, and crossover.

A GA framework

1. Randomly initialize an initial population (real-parameter representation).
2. Compute a set of Parents from the elite population using a *fitness* function (tournament selection).
3. Compute a set of Offsprings obtained from the set of Parents using the *crossover* and *mutation* operators.
4. Verify the stopping criteria. If not met then goto step 2.
Algorithm framework

Genetic algorithms

A genetic algorithm is a population based algorithm that uses techniques inspired by evolutionary biology such as inheritance, mutation, selection, and crossover.

A GA framework

1. Randomly initialize an initial population (real-parameter representation).

2. Compute a set of Parents from the elite population using a fitness function (tournament selection).

3. Compute a set of Offsprings obtained from the set of Parents using the crossover and mutation operators.

4. Verify the stopping criteria. If not met then goto step 2.
Algorithm framework

Genetic algorithms

A genetic algorithm is a population based algorithm that uses techniques inspired by evolutionary biology such as inheritance, mutation, selection, and crossover.

A GA framework

1. Randomly initialize an initial population (real-parameter representation).
2. Compute a set of Parents from the elite population using a fitness function (tournament selection).
3. Compute a set of Offsprings obtained from the set of Parents using the crossover and mutation operators.
4. Verify the stopping criteria. If not met then goto step 2.
Typical Operators

The *fitness* function

Usually the *fitness* function corresponds to the objective function. The points in the population are sorted by the objective function value.

The Tournament Selection

Tournaments are played between points and the better solution is chosen as a Parent (survival of the fittest). The process is repeated until the set of Parents is fulfilled.
Typical Operators

The fitness function

Usually the fitness function corresponds to the objective function. The points in the population are sorted by the objective function value.

The Tournament Selection

Tournaments are played between points and the better solution is chosen as a Parent (survival of the fittest). The process is repeated until the set of Parents is fulfilled.
Typical Operators

The **crossover operator**

Simulated Binary Crossover (SBX) that simulates the working principle of single-point crossover operator for binary strings.

The **mutation operator**

Polynomial mutation that guarantees that the probability of creating a point closer to the parent is more than the probability of creating one away from it.
Typical Operators

The **crossover** operator

Simulated Binary Crossover (SBX) that simulates the working principle of single-point crossover operator for binary strings.

The **mutation** operator

Polynomial mutation that guarantees that the probability of creating a point closer to the parent is more than the probability of creating one away from it.
Outline

1 Introduction

2 Genetic (Evolutionary) algorithm framework

3 Genetic algorithm framework for multi-local optimization

4 Application to SIP

5 Numerical results

6 Conclusions
The proposed *fitness* function

The *fitness* function

The use of the objective function as the *fitness* function may no longer be appropriate.

The proposed *fitness* function - Version 1

Points in the population that are nearby each other (e.g. using the Euclidean distance) are assigned a huge *fitness* value, since diversity near a local optima is not requested.

The objective function is used as the *fitness* function.
The proposed *fitness* function

**The fitness function**

The use of the objective function as the *fitness* function may no longer be appropriate.

**The proposed fitness function - Version 1**

Points in the population that are nearby each other (e.g. using the Euclidean distance) are assigned a huge *fitness* value, since diversity near a local optima is not requested.

The objective function is used as the *fitness* function.
The proposed \textit{fitness} function

The second version of the proposed \textit{fitness} function uses some concepts from the multiobjective (biobjective) optimization.

\textbf{The dominance concept}

Let $f_1(x) \equiv f(x)$ and $f_2(x) \equiv \|\nabla L(x, \lambda)\|^a$ be two objective functions. A point $x$ is said to dominate a point $y$ ($x \prec y$) if

\begin{equation}
    f_i(x) \leq f_i(y), \quad i = 1, 2 \quad \text{and} \quad \exists j \quad \text{such that} \quad f_j(x) < f_j(y). \quad (2)
\end{equation}

If the conditions in (2) are not verified we simply say that $x$ does not dominate $y$ ($x \not\prec y$).

\footnote{$\nabla L(x, \lambda)$ is the Lagrangian function with Lagrange multipliers $\lambda$.}
The proposed *fitness* function

The second version of the proposed *fitness* function uses some concepts from the multiobjective (biobjective) optimization.

The dominance concept

Let \( f_1(x) \) \( (\equiv f(x)) \) and \( f_2(x) \) \( (\equiv \|\nabla L(x, \lambda)\|^a) \) be two objective functions. A point \( x \) is said to dominate a point \( y \) \( (x \prec y) \) if

\[
  f_i(x) \leq f_i(y), \quad i = 1, 2 \quad \text{and} \quad \exists j \quad \text{such that} \quad f_j(x) < f_j(y). \quad (2)
\]

If the conditions in (2) are not verified we simply say that \( x \) does not dominate \( y \) \( (x \not\prec y) \).

\( ^aL(x, \lambda) \) is the Lagrangian function with Lagrange multipliers \( \lambda \).
The proposed *fitness* function

**Lagrangian computation**

Since we are dealing with a simple bound constrained problem the norm of the Lagrangian gradient can be computed without explicitly knowing the Lagrange multipliers.

**The proposed *fitness* function - Version 2**

Points in the population not dominated by any other point (in the population) is assigned rank 1 (*fitness* equal to 1) and removed from the *fitness* computation. Points thereafter not dominated by any other point (in the remaining point of the population) is assigned rank 2. The procedure is repeated until no point are left to assign a rank.

A point \( x \) only dominates a point \( y \) if it satisfies relations (2) and is within a specified Euclidian distance.
The proposed *fitness* function

**Lagrangian computation**

Since we are dealing with a simple bound constrained problem the norm of the Lagrangian gradient can be computed without explicitly knowing the Lagrange multipliers.

**The proposed *fitness* function - Version 2**

Points in the population not dominated by any other point (in the population) is assigned rank 1 (*fitness* equal to 1) and removed from the *fitness* computation. Points thereafter not dominated by any other point (in the remaining point of the population) is assigned rank 2. The procedure is repeated until no point are left to assign a rank.

A point $x$ only dominates a point $y$ if it satisfies relations (2) and is within a specified Euclidian distance.
Quasi-Newton iterations

A quasi-Newton iteration is performed for each point in the elite population:

- a BFGS update formula to approximate the inverse Lagrangian Hessian is used.
- The identity matrix is used whenever a new point enters the elite population.
- A point kept in the elite population performs a sequence of quasi-Newton iterations (independently of the fitness).
- a line search strategy is used together with an Armijo like rule in order to globalize the algorithm.
- A reset strategy (setting the approximation to the identity matrix) is performed each \( n \) successive iterations and the inverse Lagrangian Hessian approximation is not updated if the denominator of the updating formula is, in absolute value, lower than a specified tolerance.
Quasi-Newton iterations

A quasi-Newton iteration is performed for each point in the elite population:

- A BFGS update formula to approximate the inverse Lagrangian Hessian is used.

- The identity matrix is used whenever a new point enters the elite population.

- A point kept in the elite population performs a sequence of quasi-Newton iterations (independently of the fitness).

- A line search strategy is used together with an Armijo like rule in order to globalize the algorithm.

- A reset strategy (setting the approximation to the identity matrix) is performed each \( n \) successive iterations and the inverse Lagrangian Hessian approximation is not updated if the denominator of the updating formula is, in absolute value, lower than a specified tolerance.
Quasi-Newton iterations

A quasi-Newton iteration is performed for each point in the elite population:

- A BFGS update formula to approximate the inverse Lagrangian Hessian is used.
- The identity matrix is used whenever a new point enters the elite population.
- A point kept in the elite population performs a sequence of quasi-Newton iterations (independently of the fitness).
- A line search strategy is used together with an Armijo-like rule in order to globalize the algorithm.
- A reset strategy (setting the approximation to the identity matrix) is performed each $n$ successive iterations and the inverse Lagrangian Hessian approximation is not updated if the denominator of the updating formula is, in absolute value, lower than a specified tolerance.
Quasi-Newton iterations

A quasi-Newton iteration is performed for each point in the elite population:

- A BFGS update formula to approximate the inverse Lagrangian Hessian is used.
- The identity matrix is used whenever a new point enters the elite population.
- A point kept in the elite population performs a sequence of quasi-Newton iterations (independently of the fitness).
- A line search strategy is used together with an Armijo like rule in order to globalize the algorithm.
- A reset strategy (setting the approximation to the identity matrix) is performed each $n$ successive iterations and the inverse Lagrangian Hessian approximation is not updated if the denominator of the updating formula is, in absolute value, lower than a specified tolerance.
Quasi-Newton iterations

A quasi-Newton iteration is performed for each point in the elite population:

- a BFGS update formula to approximate the inverse Lagrangian Hessian is used.
- The identity matrix is used whenever a new point enters the elite population.
- A point kept in the elite population performs a sequence of quasi-Newton iterations (independently of the fitness).
- a line search strategy is used together with an Armijo like rule in order to globalize the algorithm.
- A reset strategy (setting the approximation to the identity matrix) is performed each $n$ successive iterations and the inverse Lagrangian Hessian approximation is not updated if the denominator of the updating formula is, in absolute value, lower than a specified tolerance.
The algorithm

The GA can be described as follows

MLOCGAMO

1. Randomly initialize the initial population $E^0$. Set $k = 0$ as the iteration counter.
2. While the number of iterations and objective function evaluations is below the given maxima then
   - Compute the set of parents $P^k$ from the elite population $E^k$ using a fitness function (selection).
   - Compute the Offspring $O^k$ from the parent population $P^k$ using the crossover and mutation operators.
   - Apply a quasi-Newton iteration for all the points in the elite set $E^k$.
   - Compute the new elite set $E^{k+1}$ by selecting points from $E^k \cup O^k$ with the best fitness. For points with the same fitness value the ones with lower objective function value are selected.
   - Set $k = k + 1$. 
The algorithm

The GA can be described as follows

1. Randomly initialize the initial population $\mathcal{E}^0$. Set $k = 0$ as the iteration counter.

2. While the number of iterations and objective function evaluations is below the given maxima then
   - Compute the set of parents $P^k$ from the elite population $\mathcal{E}^k$ using a fitness function (selection);
   - Compute the Offspring $O^k$ from the parent population $P^k$ using the crossover and mutation operators;
   - Apply a quasi-Newton iteration for all the points in the elite set $\mathcal{E}^k$.
   - Compute the new elite set $\mathcal{E}^{k+1}$ by selecting points from $\mathcal{E}^k \cup O^k$ with the best fitness. For points with the same fitness value the ones with lower objective function value are selected.
   - Set $k = k + 1$. 

Vaz (UMinho - PT)
The algorithm

The GA can be described as follows

1. Randomly initialize the initial population $E^0$. Set $k = 0$ as the iteration counter.

2. While the number of iterations and objective function evaluations is below the given maxima then

   - Compute the set of parents $P^k$ from the elite population $E^k$ using a fitness function (selection);
   - Compute the Offspring $O^k$ from the parent population $P^k$ using the crossover and mutation operators;
   - Apply a quasi-Newton iteration for all the points in the elite set $E^k$.
   - Compute the new elite set $E^{k+1}$ by selecting points from $E^k \cup O^k$ with the best fitness. For points with the same fitness value the ones with lower objective function value are selected.
   - Set $k = k + 1$. 

MLOCGAMO
The algorithm

The GA can be described as follows

**MLOCGAMO**

1. Randomly initialize the initial population $E^0$. Set $k = 0$ as the iteration counter.
2. While the number of iterations and objective function evaluations is below the given maxima then
   - Compute the set of parents $P^k$ from the elite population $E^k$ using a fitness function (selection);
   - Compute the Offspring $O^k$ from the parent population $P^k$ using the crossover and mutation operators;
   - Apply a quasi-Newton iteration for all the points in the elite set $E^k$.
   - Compute the new elite set $E^{k+1}$ by selecting points from $E^k \cup O^k$ with the best fitness. For points with the same fitness value the ones with lower objective function value are selected.
   - Set $k = k + 1$. 
The algorithm

The GA can be described as follows

**MLOC GAMO**

1. Randomly initialize the initial population $E^0$. Set $k = 0$ as the iteration counter.

2. While the number of iterations and objective function evaluations is below the given maxima then
   - Compute the set of parents $P^k$ from the elite population $E^k$ using a *fitness* function (selection);
   - Compute the Offspring $O^k$ from the parent population $P^k$ using the *crossover* and *mutation* operators;
   - Apply a quasi-Newton iteration for all the points in the elite set $E^k$.
   - Compute the new elite set $E^{k+1}$ by selecting points from $E^k \cup O^k$ with the best *fitness*. For points with the same *fitness* value the ones with lower objective function value are selected.
   - Set $k = k + 1$. 
The algorithm

The GA can be described as follows

**MLOCGAMO**

1. Randomly initialize the initial population $\mathcal{E}^0$. Set $k = 0$ as the iteration counter.

2. While the number of iterations and objective function evaluations is below the given maxima then
   - Compute the set of parents $\mathcal{P}^k$ from the elite population $\mathcal{E}^k$ using a *fitness* function (selection);
   - Compute the Offspring $\mathcal{O}^k$ from the parent population $\mathcal{P}^k$ using the *crossover* and *mutation* operators;
   - Apply a quasi-Newton iteration for all the points in the elite set $\mathcal{E}^k$.
   - Compute the new elite set $\mathcal{E}^{k+1}$ by selecting points from $\mathcal{E}^k \cup \mathcal{O}^k$ with the best *fitness*. For points with the same *fitness* value the ones with lower objective function value are selected.
   - Set $k = k + 1$. 
The algorithm

The GA can be described as follows

\[ \text{MLOCGAMO} \]

1. Randomly initialize the initial population \( E^0 \). Set \( k = 0 \) as the iteration counter.

2. While the number of iterations and objective function evaluations is below the given maxima then
   - Compute the set of parents \( P^k \) from the elite population \( E^k \) using a \textit{fitness} function (selection);
   - Compute the Offspring \( O^k \) from the parent population \( P^k \) using the \textit{crossover} and \textit{mutation} operators;
   - Apply a quasi-Newton iteration for all the points in the elite set \( E^k \).
   - Compute the new elite set \( E^{k+1} \) by selecting points from \( E^k \cup O^k \) with the best \textit{fitness}. For points with the same \textit{fitness} value the ones with lower objective function value are selected.
   - Set \( k = k + 1 \).
The algorithm

The GA can be described as follows:

1. Randomly initialize the initial population $E^0$. Set $k = 0$ as the iteration counter.
2. While the number of iterations and objective function evaluations is below the given maxima then
   a. Compute the set of parents $P^k$ from the elite population $E^k$ using a fitness function (selection);
   b. Compute the Offspring $O^k$ from the parent population $P^k$ using the crossover and mutation operators;
   c. Apply a quasi-Newton iteration for all the points in the elite set $E^k$.
   d. Compute the new elite set $E^{k+1}$ by selecting points from $E^k \cup O^k$ with the best fitness. For points with the same fitness value the ones with lower objective function value are selected.
   e. Set $k = k + 1$. 
Outline

1. Introduction
2. Genetic (Evolutionary) algorithm framework
3. Genetic algorithm framework for multi-local optimization
4. Application to SIP
5. Numerical results
6. Conclusions
A semi-infinite programming problem can be described in the following form:

\[
\begin{align*}
\min_{y \in \mathbb{R}^m} & \quad g(y), \\
\text{s.t.} & \quad h(y, x) \leq 0 \\
& \quad \forall x \in \Omega \subset \mathbb{R}^n.
\end{align*}
\]

**Why semi-infinite?**

These problems are characterized to have a finite number of variables \(m\) to be determined subject to an infinite number of constraints (recall that \(\Omega\) is an infinite set).
**Definition**

A semi-infinite programming problem can be described in the following form:

\[
\begin{align*}
\min_{y \in \mathbb{R}^m} & \quad g(y), \\
\text{s.t.} & \quad h(y, x) \leq 0 \\
& \quad \forall x \in \Omega \subset \mathbb{R}^n.
\end{align*}
\]

**Why semi-infinite?**

These problems are characterized to have a finite number of variables \((m)\) to be determined subject to an infinite number of constraints (recall that \(\Omega\) is an infinite set).
Multi-local

Where multi-local plays a role?

One of the major difficulties is to deal with the infinite number of constraints. A simple check of the $y^*$ KKT conditions for optimality requests the computation of all the global optima (the set of global optima is the set of active constraints) for the problem

$$\min_{x \in \Omega} f(x) \equiv -h(y^*, x).$$  \hspace{1cm} (3)

For $\bar{y}$ to be feasible:

$$h(\bar{y}, x) \leq 0 \quad \forall x \in \Omega$$
Multi-local

Where multi-local plays a role?

One of the major difficulties is to deal with the infinite number of constraints. A simple check of the $y^*$ KKT conditions for optimality requests the computation of all the global optima (the set of global optima is the set of active constraints) for the problem

$$\min_{x \in \Omega} f(x) \equiv -h(y^*, x).$$ (3)

For $\bar{y}$ to be feasible:

$$h(\bar{y}, x) \leq 0$$
$$\forall x \in \Omega$$
Outline

1. Introduction
2. Genetic (Evolutionary) algorithm framework
3. Genetic algorithm framework for multi-local optimization
4. Application to SIP
5. Numerical results
6. Conclusions
Test problems

A set of test SIP problems were obtained from the SIPAMPL problem database where problem (3) was considered.

Multi-local problems considered

<table>
<thead>
<tr>
<th>SIP problem</th>
<th>Test problem</th>
<th>$n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>watson2</td>
<td>sip_wat2</td>
<td>1</td>
</tr>
<tr>
<td>vaz1</td>
<td>sip_vaz1</td>
<td>2</td>
</tr>
<tr>
<td>vaz3</td>
<td>sip_vaz3</td>
<td>2</td>
</tr>
<tr>
<td>priceS6</td>
<td>sip_S6</td>
<td>6</td>
</tr>
<tr>
<td>priceU</td>
<td>sip_U</td>
<td>6</td>
</tr>
</tbody>
</table>

Multi-local problems considered

Since problems were obtained from SIP by replacing $y$ by $y^*$ the global optima is attained at $f^* = 0$. 
Test problems

A set of test SIP problems were obtained from the SIPAMPL problem database where problem (3) was considered.

### Multi-local problems considered

<table>
<thead>
<tr>
<th>SIP problem</th>
<th>Test problem</th>
<th>$n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>watson2</td>
<td>sip_wat2</td>
<td>1</td>
</tr>
<tr>
<td>vaz1</td>
<td>sip_vaz1</td>
<td>2</td>
</tr>
<tr>
<td>vaz3</td>
<td>sip_vaz3</td>
<td>2</td>
</tr>
<tr>
<td>priceS6</td>
<td>sip_S6</td>
<td>6</td>
</tr>
<tr>
<td>priceU</td>
<td>sip_U</td>
<td>6</td>
</tr>
</tbody>
</table>

Since problems were obtained from SIP by replacing $y$ by $y^*$ the global optima is attained at $f^* = 0$. 
Numerical results

Test problems

Additional problems

Other well known problems from global optimization were also considered.

Some details

- 4000 function evaluations allowed (since we have a population algorithm this limit can be slightly exceeded).
- Since we are dealing with a stochastic algorithm we are reporting average values for 10 runs.
- We are considering that a global optima was obtained when the best obtained point is a Lagrangian stationary point.
- A local optima is obtained if it is a stationary point of the Lagrangian and its objective function value is far away from the best obtained point.
Numerical results

Test problems

Additional problems

Other well known problems from global optimization were also considered.

Some details

- 4000 function evaluations allowed (since we have a population algorithm this limit can be slightly exceeded).
- Since we are dealing with a stochastic algorithm we are reporting average values for 10 runs.
- We are considering that a global optima was obtained when the best obtained point is a Lagrangian stationary point.
- A local optima is obtained if it is a stationary point of the Lagrangian and its objective function value is far away from the best obtained point.
Test problems

Additional problems
Other well known problems from global optimization were also considered.

Some details
- 4000 function evaluations allowed (since we have a population algorithm this limit can be slightly exceeded).
- Since we are dealing with a stochastic algorithm we are reporting average values for 10 runs.
- We are considering that a global optima was obtained when the best obtained point is a Lagrangian stationary point.
- A local optima is obtained if it is a stationary point of the Lagrangian and its objective function value is far away from the best obtained point.
Numerical results

Test problems

Additional problems

Other well known problems from global optimization were also considered.

Some details

- 4000 function evaluations allowed (since we have a population algorithm this limit can be slightly exceeded).

- Since we are dealing with a stochastic algorithm we are reporting average values for 10 runs.

- We are considering that a global optima was obtained when the best obtained point is a Lagrangian stationary point.

- A local optima is obtained if it is a stationary point of the Lagrangian and its objective function value is far away from the best obtained point.
Test problems

Additional problems

Other well known problems from global optimization were also considered.

Some details

- 4000 function evaluations allowed (since we have a population algorithm this limit can be slightly exceeded).
- Since we are dealing with a stochastic algorithm we are reporting average values for 10 runs.
- We are considering that a global optima was obtained when the best obtained point is a Lagrangian stationary point.
- A local optima is obtained if it is a stationary point of the Lagrangian and its objective function value is far away from the best obtained point.
Numerical results

Test problems

Additional problems

Other well known problems from global optimization were also considered.

Some details

- 4000 function evaluations allowed (since we have a population algorithm this limit can be slightly exceeded).
- Since we are dealing with a stochastic algorithm we are reporting average values for 10 runs.
- We are considering that a global optima was obtained when the best obtained point is a Lagrangian stationary point.
- A local optima is obtained if it is a stationary point of the Lagrangian and its objective function value is far away from the best obtained point.
## Numerical results - Version 1

<table>
<thead>
<tr>
<th>Test functions</th>
<th>$n$</th>
<th>$N_x^*$</th>
<th>$f^*$</th>
<th>$f_{mean}$ ($f_{sd}$)</th>
<th>$N_{glob}$</th>
<th>$N_{loc}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>b2</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>5.71E-14 (1.81E-13)</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>bohachevsky</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>5.72E-14 (1.81E-13)</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>branin</td>
<td>2</td>
<td>3</td>
<td>3.98E-01</td>
<td>3.98E-01 (5.85E-17)</td>
<td>2.9</td>
<td>2.9</td>
</tr>
<tr>
<td>dejong</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0.00E+00 (0.00E+00)</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>easom</td>
<td>2</td>
<td>1</td>
<td>-1</td>
<td>-1.00E+00 (0.00E+00)</td>
<td>1</td>
<td>58.2</td>
</tr>
<tr>
<td>f1</td>
<td>30</td>
<td>1</td>
<td>-1.26E+04</td>
<td>-1.43E+04 (1.04E+01)</td>
<td>0.6</td>
<td>1</td>
</tr>
<tr>
<td>goldprice</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>3.00E+00 (5.11E-04)</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>griewank</td>
<td>6</td>
<td>1</td>
<td>0</td>
<td>2.39E-02 (8.69E-03)</td>
<td>1.7</td>
<td>23.2</td>
</tr>
<tr>
<td>hartmann3</td>
<td>3</td>
<td>1</td>
<td>-3.86E+00</td>
<td>-3.86E+00 (9.70E-04)</td>
<td>0</td>
<td>1.7</td>
</tr>
<tr>
<td>hartmann6</td>
<td>6</td>
<td>1</td>
<td>-3.32E+00</td>
<td>-3.32E+00 (9.72E-07)</td>
<td>0.4</td>
<td>0.9</td>
</tr>
<tr>
<td>hump</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>4.65E-08 (8.59E-13)</td>
<td>1</td>
<td>1.2</td>
</tr>
<tr>
<td>hump_camel</td>
<td>2</td>
<td>2</td>
<td>-1.0316285</td>
<td>-1.03E+00 (2.34E-16)</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>levy3</td>
<td>2</td>
<td>18</td>
<td>-1.77E+02</td>
<td>-1.77E+02 (0.00E+00)</td>
<td>3.8</td>
<td>11.9</td>
</tr>
<tr>
<td>parsopoulos</td>
<td>2</td>
<td>12</td>
<td>0</td>
<td>2.61E-04 (4.19E-04)</td>
<td>0.6</td>
<td>7</td>
</tr>
<tr>
<td>rosenbrock10</td>
<td>10</td>
<td>1</td>
<td>0</td>
<td>1.96E+02 (2.58E+02)</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>rosenbrock2</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>3.65E-02 (7.33E-02)</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>rosenbrock5</td>
<td>5</td>
<td>1</td>
<td>0</td>
<td>4.74E-01 (1.59E-01)</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>shekel10</td>
<td>4</td>
<td>1</td>
<td>-1.05E+01</td>
<td>-7.85E+00 (2.84E+00)</td>
<td>1</td>
<td>4.9</td>
</tr>
<tr>
<td>shekel5</td>
<td>4</td>
<td>1</td>
<td>-1.02E+01</td>
<td>-9.65E+00 (1.60E+00)</td>
<td>1</td>
<td>3.2</td>
</tr>
</tbody>
</table>
### Numerical results - Version 1 (Cont.)

<table>
<thead>
<tr>
<th>Test functions</th>
<th>$n$</th>
<th>$N_{x^*}$</th>
<th>$f^*$</th>
<th>$f_{mean}$ ($f_{sd}$)</th>
<th>$N_{glob}$</th>
<th>$N_{loc}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>shekel7</td>
<td>4</td>
<td>1</td>
<td>-1.04E+01</td>
<td>-7.76E+00 (2.78E+00)</td>
<td>1</td>
<td>4.2</td>
</tr>
<tr>
<td>shubert</td>
<td>2</td>
<td>18</td>
<td>-1.87E+02</td>
<td>-1.87E+02 (0.00E+00)</td>
<td>8</td>
<td>15.9</td>
</tr>
<tr>
<td>storn1</td>
<td>2</td>
<td>2</td>
<td>-4.08E-01</td>
<td>-4.07E-01 (0.00E+00)</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>storn2</td>
<td>2</td>
<td>2</td>
<td>-1.81E+01</td>
<td>-1.81E+01 (3.74E-15)</td>
<td>2</td>
<td>2.2</td>
</tr>
<tr>
<td>storn3</td>
<td>2</td>
<td>2</td>
<td>-2.28E+02</td>
<td>-2.28E+02 (6.39E-02)</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>storn4</td>
<td>2</td>
<td>2</td>
<td>-2.43E+03</td>
<td>-2.43E+03 (8.68E-03)</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>storn5</td>
<td>2</td>
<td>2</td>
<td>-2.48E+04</td>
<td>-2.48E+04 (1.85E+01)</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>storn6</td>
<td>2</td>
<td>2</td>
<td>-2.49E+05</td>
<td>-2.48E+04 (6.44E+00)</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>zakharov10</td>
<td>10</td>
<td>1</td>
<td>0</td>
<td>5.38E-11 (1.11E-10)</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>zakharov2</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>2.65E-12 (2.47E-12)</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>zakharov20</td>
<td>20</td>
<td>1</td>
<td>0</td>
<td>5.00E-02 (1.19E-01)</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>zakharov4</td>
<td>4</td>
<td>1</td>
<td>0</td>
<td>3.30E-12 (3.99E-12)</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>zakharov5</td>
<td>5</td>
<td>1</td>
<td>0</td>
<td>1.02E-12 (6.34E-13)</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>spherical</td>
<td>6</td>
<td>2</td>
<td>1</td>
<td>-3.00E+05 (0.00E+00)</td>
<td>40</td>
<td>40</td>
</tr>
<tr>
<td>sip_S6</td>
<td>6</td>
<td>1</td>
<td>0</td>
<td>-9.25E+00 (0.00E+00)</td>
<td>1</td>
<td>15.8</td>
</tr>
<tr>
<td>sip_U</td>
<td>6</td>
<td>1</td>
<td>0</td>
<td>-8.00E+00 (9.53E-05)</td>
<td>1.2</td>
<td>4.1</td>
</tr>
<tr>
<td>sip_vaz1</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0.00E+00 (0.00E+00)</td>
<td>74</td>
<td>74</td>
</tr>
<tr>
<td>sip_vaz3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0.00E+00 (0.00E+00)</td>
<td>69</td>
<td>69</td>
</tr>
<tr>
<td>sip_wat2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>-1.11E-01 (1.46E-17)</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>
### Numerical results - Version 2

<table>
<thead>
<tr>
<th>Test functions</th>
<th>$n$</th>
<th>$N_{x^*}$</th>
<th>$f^*$</th>
<th>$f_{mean}(f_{sd})$</th>
<th>$N_{glob}$</th>
<th>$N_{loc}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>b2</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1.55E-14 (4.91E-14)</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>bohachevsky</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>-5.55E-17 (6.50E-33)</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>branin</td>
<td>2</td>
<td>3</td>
<td>3.98E-01</td>
<td>3.98E-01 (5.85E-17)</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>dejoung</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0.00E+00 (0.00E+00)</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>easom</td>
<td>2</td>
<td>1</td>
<td>-1</td>
<td>-1.00E+00 (0.00E+00)</td>
<td>1</td>
<td>58.4</td>
</tr>
<tr>
<td>f1</td>
<td>30</td>
<td>1</td>
<td>-1.26E+04</td>
<td>-1.43E+04 (4.01E+01)</td>
<td>0.4</td>
<td>0.5</td>
</tr>
<tr>
<td>goldprice</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>3.00E+00 (1.58E-04)</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>griewank</td>
<td>6</td>
<td>1</td>
<td>0</td>
<td>1.70E-02 (1.27E-02)</td>
<td>1.3</td>
<td>21.9</td>
</tr>
<tr>
<td>hartmann3</td>
<td>3</td>
<td>1</td>
<td>-3.86E+00</td>
<td>-3.86E+00 (2.42E-03)</td>
<td>0.1</td>
<td>2.1</td>
</tr>
<tr>
<td>hartmann6</td>
<td>6</td>
<td>1</td>
<td>-3.32E+00</td>
<td>-3.32E+00 (1.27E-06)</td>
<td>0.2</td>
<td>1.3</td>
</tr>
<tr>
<td>hump</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>4.65E-08 (1.65E-12)</td>
<td>1</td>
<td>1.1</td>
</tr>
<tr>
<td>hump_camel</td>
<td>2</td>
<td>2</td>
<td>-1.0316285</td>
<td>-1.03E+00 (2.34E-16)</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>levy3</td>
<td>2</td>
<td>18</td>
<td>-1.77E+02</td>
<td>-1.77E+02 (0.00E+00)</td>
<td>2.8</td>
<td>12</td>
</tr>
<tr>
<td>parsopoulos</td>
<td>2</td>
<td>12</td>
<td>0</td>
<td>1.14E-04 (2.01E-04)</td>
<td>0.9</td>
<td>7.3</td>
</tr>
<tr>
<td>rosenbrock10</td>
<td>10</td>
<td>1</td>
<td>0</td>
<td>3.22E+02 (3.95E+02)</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>rosenbrock2</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>8.74E-02 (8.74E-02)</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>rosenbrock5</td>
<td>5</td>
<td>1</td>
<td>0</td>
<td>4.41E-01 (2.75E-01)</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>shekel10</td>
<td>4</td>
<td>1</td>
<td>-1.05E+01</td>
<td>-8.39E+00 (2.77E+00)</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>shekel5</td>
<td>4</td>
<td>1</td>
<td>-1.02E+01</td>
<td>-1.02E+01 (0.00E+00)</td>
<td>1</td>
<td>3.7</td>
</tr>
</tbody>
</table>
## Numerical results

### Test functions

<table>
<thead>
<tr>
<th>Test functions</th>
<th>$n$</th>
<th>$N_{x*}$</th>
<th>$f^*$</th>
<th>$f_{mean}(f_{sd})$</th>
<th>$N_{glob}$</th>
<th>$N_{loc}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>shekel7</td>
<td>4</td>
<td>1</td>
<td>-1.04E+01</td>
<td>-9.35E+00 ( 2.22E+00 )</td>
<td>1</td>
<td>4.2</td>
</tr>
<tr>
<td>shubert</td>
<td>2</td>
<td>18</td>
<td>-1.87E+02</td>
<td>-1.87E+02 ( 0.00E+00 )</td>
<td>8.2</td>
<td>16.7</td>
</tr>
<tr>
<td>storn1</td>
<td>2</td>
<td>2</td>
<td>-4.08E-01</td>
<td>-4.07E-01 ( 0.00E+00 )</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>storn2</td>
<td>2</td>
<td>2</td>
<td>-1.81E+01</td>
<td>-1.81E+01 ( 3.74E-15 )</td>
<td>2</td>
<td>2.4</td>
</tr>
<tr>
<td>storn3</td>
<td>2</td>
<td>2</td>
<td>-2.28E+02</td>
<td>-2.28E+02 ( 1.14E-01 )</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>storn4</td>
<td>2</td>
<td>2</td>
<td>-2.43E+03</td>
<td>-2.43E+03 ( 9.40E-02 )</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>storn5</td>
<td>2</td>
<td>2</td>
<td>-2.48E+04</td>
<td>-2.48E+04 ( 7.63E+00 )</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>storn6</td>
<td>2</td>
<td>2</td>
<td>-2.49E+05</td>
<td>-2.48E+04 ( 8.09E+00 )</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>zakharov10</td>
<td>10</td>
<td>1</td>
<td>0</td>
<td>8.07E-11 ( 1.41E-10 )</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>zakharov2</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1.86E-12 ( 1.78E-12 )</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>zakharov20</td>
<td>20</td>
<td>1</td>
<td>0</td>
<td>4.35E-02 ( 5.63E-02 )</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>zakharov4</td>
<td>4</td>
<td>1</td>
<td>0</td>
<td>3.29E-11 ( 8.68E-11 )</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>zakharov5</td>
<td>5</td>
<td>1</td>
<td>0</td>
<td>9.71E-13 ( 1.67E-12 )</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>spherical</td>
<td>6</td>
<td>2</td>
<td>1</td>
<td>-3.00E+05 ( 0.00E+00 )</td>
<td>40</td>
<td>40</td>
</tr>
<tr>
<td>sip_S6</td>
<td>6</td>
<td>1</td>
<td>0</td>
<td>-9.25E+00 ( 0.00E+00 )</td>
<td>1</td>
<td>15.7</td>
</tr>
<tr>
<td>sip_U</td>
<td>6</td>
<td>1</td>
<td>0</td>
<td>-8.00E+00 ( 3.91E-04 )</td>
<td>1.5</td>
<td>4</td>
</tr>
<tr>
<td>sip_vaz1</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0.00E+00 ( 0.00E+00 )</td>
<td>72.3</td>
<td>72.3</td>
</tr>
<tr>
<td>sip_vaz3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0.00E+00 ( 0.00E+00 )</td>
<td>69.5</td>
<td>69.6</td>
</tr>
<tr>
<td>sip_wat2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>-1.11E-01 ( 1.46E-17 )</td>
<td>1</td>
<td>2.1</td>
</tr>
</tbody>
</table>
Numerical results

Version 1 versus Version 2

Version 1 vs Version 2 – 1/(Average number of global optima)

Vaz (UMinho - PT)
Outline

1. Introduction
2. Genetic (Evolutionary) algorithm framework
3. Genetic algorithm framework for multi-local optimization
4. Application to SIP
5. Numerical results
6. Conclusions
Conclusions

- We propose an algorithm for multi-local optimization (with two versions).
- The proposed algorithm uses a neighborhood concept for the fitness function. One version considered the dominance concept from multi-objective optimization.
- The algorithms use a quasi-Newton step in order to accelerate the convergence to local optima.
- We provide numerical results for the implemented algorithm and a compare between the two versions.
- Version 2 (with the dominance concept from multi-objective) proved to be (slightly) less efficient than version 1.
Conclusions

- We propose an algorithm for multi-local optimization (with two versions).
- The proposed algorithm uses a neighborhood concept for the fitness function. One version considered the dominance concept from multi-objective optimization.
- The algorithms use a quasi-Newton step in order to accelerate the convergence to local optima.
- We provide numerical results for the implemented algorithm and a compare between the two versions.
- Version 2 (with the dominance concept from multi-objective) proved to be (slightly) less efficient than version 1.
We propose an algorithm for multi-local optimization (with two versions).

The proposed algorithm uses a neighborhood concept for the fitness function. One version considered the dominance concept from multi-objective optimization.

The algorithms use a quasi-Newton step in order to accelerate the convergence to local optima.

We provide numerical results for the implemented algorithm and a compare between the two versions.

Version 2 (with the dominance concept from multi-objective) proved to be (slightly) less efficient than version 1.
Conclusions

- We propose an algorithm for multi-local optimization (with two versions).
- The proposed algorithm uses a neighborhood concept for the fitness function. One version considered the dominance concept from multi-objective optimization.
- The algorithms use a quasi-Newton step in order to accelerate the convergence to local optima.
- We provide numerical results for the implemented algorithm and a compare between the two versions.
- Version 2 (with the dominance concept from multi-objective) proved to be (slightly) less efficient than version 1.
Conclusions

- We propose an algorithm for multi-local optimization (with two versions).
- The proposed algorithm uses a neighborhood concept for the fitness function. One version considered the dominance concept from multi-objective optimization.
- The algorithms use a quasi-Newton step in order to accelerate the convergence to local optima.
- We provide numerical results for the implemented algorithm and a compare between the two versions.
- Version 2 (with the dominance concept from multi-objective) proved to be (slightly) less efficient than version 1.
email: aivaz@dps.uminho.pt
Web http://www.norg.uminho.pt/aivaz

email: lac@dps.uminho.pt
Web http://pessoais.dps.uminho.pt/lac

Supported by
FCT Fundação para a Ciência e a Tecnologia
MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E ENSINO SUPERIOR Portugal
EURO XXIV LISBON
24th European Conference on Operational Research

July 11-14 www.euro2010lisbon.org

Welcome!