Particle swarm algorithms for multi-local optimization

A. Ismael F. Vaz
Edite M.G.P. Fernandes
Production and System Department
Engineering School
Minho University
{aivaz,emgpf}@dps.uminho.pt

Work partially supported by FCT grant POCI/MAT/58957/2004 and Algoritmi research center
Outline

- **Motivation**
- The multi-local optimization problem
- The particle swarm paradigm for global optimization
- Particle swarm variants for multi-local optimization
- Implementation
- Numerical results
- Conclusions
Outline

- Motivation
- The multi-local optimization problem
- The particle swarm paradigm for global optimization
- Particle swarm variants for multi-local optimization
- Implementation
- Numerical results
- Conclusions
Outline

- Motivation
- The multi-local optimization problem
- The particle swarm paradigm for global optimization
- Particle swarm variants for multi-local optimization
- Implementation
- Numerical results
- Conclusions
Outline

- Motivation
- The multi-local optimization problem
- The particle swarm paradigm for global optimization
- Particle swarm variants for multi-local optimization
- Implementation
- Numerical results
- Conclusions
Outline

- Motivation
- The multi-local optimization problem
- The particle swarm paradigm for global optimization
- Particle swarm variants for multi-local optimization

Implementation
- Numerical results
- Conclusions
Outline

- Motivation
- The multi-local optimization problem
- The particle swarm paradigm for global optimization
- Particle swarm variants for multi-local optimization
- Implementation
- **Numerical results**
- Conclusions
Outline

- Motivation
- The multi-local optimization problem
- The particle swarm paradigm for global optimization
- Particle swarm variants for multi-local optimization
- Implementation
- Numerical results
- Conclusions
Motivation

One of the (many) applications of multi-local optimization is in reduction type methods for semi-infinite programming (SIP) problems.
**Motivation**

One of the (many) applications of multi-local optimization is in reduction type methods for semi-infinite programming (SIP) problems.

A SIP problems can be posed as:

$$\min_{y \in \mathbb{R}^q} o(y)$$

$$s.t. \quad f_i(y, x) \geq 0, \quad i = 1, \ldots, m$$

$$\forall x \in T \subset \mathbb{R}^n,$$

where $o(y)$ is the objective function and $f_i(y, x), \ i = 1, \ldots, m,$ are the infinite constraint functions.
Motivation

One of the (many) applications of multi-local optimization is in reduction type methods for semi-infinite programming (SIP) problems.

A SIP problems can be posed as:

\[
\begin{align*}
\min_{y \in \mathbb{R}^q} \quad & o(y) \\
\text{s.t.} \quad & f_i(y, x) \geq 0, \quad i = 1, \ldots, m \\
& \forall x \in T \subset \mathbb{R}^n,
\end{align*}
\]

where \( o(y) \) is the objective function and \( f_i(y, x), i = 1, \ldots, m, \) are the infinite constraint functions.

A feasible point must satisfy:

\[
f_i(y, x) \geq 0, \quad i = 1, \ldots, m, \quad \forall x \in T
\]

meaning that the global minima of \( f_i \) must be upper than or equal to zero.
Multi-local optimization

Assume, for the sake of simplicity, that \( m = 1 \). Then we want to address the following optimization problem

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

\[
s.t. \quad a \leq x \leq b
\]

where \( f : \mathbb{R}^n \rightarrow \mathbb{R} \) is the objective function and \( a, b \) are the simple bounds on the variables \( x \) (defining the set \( T \)).
Multi-local optimization

Assume, for the sake of simplicity, that $m = 1$. Then we want to address the following optimization problem

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

s.t. $a \leq x \leq b$

where $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is the objective function and $a, b$ are the simple bounds on the variables $x$ (defining the set $T$).

In each iteration of a reduction type method for SIP we need to obtain all the feasible global and local optima for function $f(x)$. 
The Particle Swarm Paradigm (PSP)

The PSP is a population (swarm) based algorithm that mimics the social behavior of a set of individuals (particles).

An individual behavior is a combination of its past experience (cognition influence) and the society experience (social influence).

In the optimization context a particle $p$, at time instant $t$, is represented by its current position ($x^p(t)$), its best ever position ($y^p(t)$) and its travelling velocity ($v^p(t)$).
The new travel position and velocity

The new particle position is updated by

\[ x^P(t + 1) = x^P(t) + v^P(t + 1), \]

where \( v^P(t + 1) \) is the new velocity given by

\[ v_j^P(t + 1) = \iota(t)v_j^P(t) + \mu \omega_{1j}(t) (y_j^P(t) - x_j^P(t)) + \nu \omega_{2j}(t) (\hat{y}_j(t) - x_j^P(t)), \]

for \( j = 1, \ldots, n. \)

- \( \iota(t) \) is a weighting factor (inertial)
- \( \mu \) is the cognition parameter and \( \nu \) is the social parameter
- \( \omega_{1j}(t) \) and \( \omega_{2j}(t) \) are random numbers drawn from the uniform \((0, 1)\) distribution.
The new travel position and velocity

The new particle position is updated by

\[ x^P(t + 1) = x^P(t) + v^P(t + 1), \]

where \(v^P(t + 1)\) is the new velocity given by

\[ v^P_j(t+1) = \nu(t) v^P_j(t) + \mu \omega^P_1 j(t) \left( y^P_j(t) - x^P_j(t) \right) + \nu \omega^P_2 j(t) \left( \hat{y}^P_j(t) - x^P_j(t) \right), \]

for \( j = 1, \ldots, n \).

- \( \nu(t) \) is a weighting factor (inertial)
- \( \mu \) is the cognition parameter and \( \nu \) is the social parameter
- \( \omega^P_1 j(t) \) and \( \omega^P_2 j(t) \) are random numbers drawn from the uniform \((0, 1)\) distribution.
The new travel position and velocity

The new particle position is updated by

\[ x^P(t + 1) = x^P(t) + v^P(t + 1), \]

where \( v^P(t + 1) \) is the new velocity given by

\[ v^P_j(t+1) = \iota(t)v^P_j(t) + \mu \omega_1j(t) (y_j^P(t) - x_j^P(t)) + \nu \omega_2j(t) (\hat{y}_j(t) - x_j^P(t)), \]

for \( j = 1, \ldots, n \).

- \( \iota(t) \) is a weighting factor (inertial)
- \( \mu \) is the **cognition** parameter and \( \nu \) is the **social** parameter
- \( \omega_{1j}(t) \) and \( \omega_{2j}(t) \) are random numbers drawn from the uniform \((0, 1)\) distribution.
The new travel position and velocity

The new particle position is updated by

\[ x^P(t + 1) = x^P(t) + v^P(t + 1), \]

where \( v^P(t + 1) \) is the new velocity given by

\[ v^P_j(t+1) = \iota(t)v^P_j(t) + \mu_1 \omega_1^j(t) (y^P_j(t) - x^P_j(t)) + \nu_2 \omega_2^j(t) (\hat{y}_j(t) - x^P_j(t)), \]

for \( j = 1, \ldots, n. \)

- \( \iota(t) \) is a weighting factor (inertial)
- \( \mu \) is the cognition parameter and \( \nu \) is the social parameter
- \( \omega_1^j(t) \) and \( \omega_2^j(t) \) are random numbers drawn from the uniform \((0, 1)\) distribution.
The best ever particle

\( \hat{y}(t) \) is a particle position with global best function value so far, i.e.,

\[
\hat{y}(t) = \arg \min_{a \in A} f(a)
\]

\[ A = \{ y^1(t), \ldots, y^s(t) \} . \]

where \( s \) is the number of particles in the swarm.
The best ever particle

\( \hat{y}(t) \) is a particle position with global best function value so far, i.e.,

\[
\hat{y}(t) = \arg \min_{a \in \mathcal{A}} f(a)
\]

\[\mathcal{A} = \{y^1(t), \ldots, y^s(t)\} \]

where \( s \) is the number of particles in the swarm.

In an algorithmic point of view we just have to keep track of the particle with the best ever function value.
Features

Population based algorithm.
1. Good
   (a) Easy to implement.
   (b) Easy to parallelize.
   (c) Easy to handle discrete variables.
   (d) Only uses objective function evaluations.

2. Not so good
   (a) Slow rate of convergence near an optimum.
   (b) Quite large number of function evaluations.
   (c) In the presence of several global optima the algorithm may not converge.
Features

Population based algorithm.

1. Good
   (a) Easy to implement.
   (b) Easy to parallelize.
   (c) Easy to handle discrete variables.
   (d) Only uses objective function evaluations.

2. Not so good
   (a) Slow rate of convergence near an optimum.
   (b) Quite large number of function evaluations.
   (c) In the presence of several global optima the algorithm may not converge.
Features

Population based algorithm.

1. Good
   (a) Easy to implement.
   (b) Easy to parallelize.
   (c) Easy to handle discrete variables.
   (d) Only uses objective function evaluations.

2. Not so good
   (a) Slow rate of convergence near an optimum.
   (b) Quite large number of function evaluations.
   (c) In the presence of several global optima the algorithm may not converge.
Features

Population based algorithm.

1. Good
   (a) Easy to implement.
   (b) Easy to parallelize.
   (c) Easy to handle discrete variables.
   (d) Only uses objective function evaluations.

2. Not so good
   (a) Slow rate of convergence near an optimum.
   (b) Quite large number of function evaluations.
   (c) In the presence of several global optima the algorithm may not converge.
Features

Population based algorithm.

1. Good
   (a) Easy to implement.
   (b) Easy to parallelize.
   (c) Easy to handle discrete variables.
   (d) Only uses objective function evaluations.

2. Not so good
   (a) Slow rate of convergence near an optimum.
   (b) Quite large number of function evaluations.
   (c) In the presence of several global optima the algorithm may not converge.
**Features**

Population based algorithm.

1. Good
   (a) Easy to implement.
   (b) Easy to parallelize.
   (c) Easy to handle discrete variables.
   (d) Only uses objective function evaluations.

2. Not so good
   (a) Slow rate of convergence near an optimum.
   (b) Quite large number of function evaluations.
   (c) In the presence of several global optima the algorithm may not converge.
**Features**

Population based algorithm.

1. Good
   (a) Easy to implement.
   (b) Easy to parallelize.
   (c) Easy to handle discrete variables.
   (d) Only uses objective function evaluations.

2. Not so good
   (a) Slow rate of convergence near an optimum.
   (b) Quite large number of function evaluations.
   (c) In the presence of several global optima the algorithm may not converge.
PSP with the steepest descent direction

The new particle position is updated by

\[ x^P(t + 1) = x^P(t) + v^P(t + 1), \]

where \( v^P(t + 1) \) is the new velocity given by

\[ v^P_j(t+1) = \iota(t)v^P_j(t)+\mu_1\omega_j(t) (y^P_j(t) - x^P_j(t)) + \nu_2\omega_j(t) (-\nabla_j f(y^P_j(t))), \]

for \( j = 1, \ldots, n \), where \( \nabla f(x) \) is the gradient of the objective function.

Each particle uses the steepest descent direction computed at each particle best position \( (y^P_j(t)) \).

The inclusion of the steepest descent direction in the velocity equation aims to drive each particle to a neighbor local minimum and since we have a population of particles, each one will be driven to a local minimum.
PSP with a descent direction

Other approach is to use

\[ w^p = \frac{-1}{\sum_{k=1}^{m} |f(z^p_k) - f(y^p)|} \sum_{k=1}^{m} (f(z^p_k) - f(y^p)) \frac{(z^p_k - y^p)}{\|z^p_k - y^p\|} \]

as a descent direction at \( y^p \), in the velocity equation, to overcome the need to compute the gradient.

Where

- \( y^p \) is the best position of particle \( p \)
- \( \{z^p_k\}_{k=1}^m \) is a set of \( m \) (random) points close to \( y^p \),
**PSP with a descent direction**

Other approach is to use

$$w^p = \frac{-1}{\sum_{k=1}^{m} |f(z_k^p) - f(y^p)| \sum_{k=1}^{m} (f(z_k^p) - f(y^p)) \frac{(z_k^p - y^p)}{\|z_k^p - y^p\|}}$$

as a descent direction at $y^p$, in the velocity equation, to overcome the need to compute the gradient.

**Where**

- $y^p$ is the best position of particle $p$

- $\{z_k^p\}_{k=1}^{m}$ is a set of $m$ (random) points close to $y^p$,
**PSP with a descent direction**

Other approach is to use

\[
    w^p = \frac{-1}{\sum_{k=1}^{m} |f(z^p_k) - f(y^p)|} \sum_{k=1}^{m} (f(z^p_k) - f(y^p)) \frac{(z^p_k - y^p)}{\|z^p_k - y^p\|}
\]

as a descent direction at \( y^p \), in the velocity equation, to overcome the need to compute the gradient.

**Where**

- \( y^p \) is the best position of particle \( p \)
- \( \{z^p_k\}_{k=1}^{m} \) is a set of \( m \) (random) points close to \( y^p \),
PSP with a descent direction

Other approach is to use

\[ w^p = \frac{-1}{\sum_{k=1}^{m} |f(z^p_k) - f(y^p)|} \sum_{k=1}^{m} (f(z^p_k) - f(y^p)) \frac{(z^p_k - y^p)}{\|z^p_k - y^p\|} \]

as a descent direction at \( y^p \), in the velocity equation, to overcome the need to compute the gradient.

Where
- \( y^p \) is the best position of particle \( p \)
- \( \{z^p_k\}_{k=1}^{m} \) is a set of \( m \) (random) points close to \( y^p \),

Under certain conditions \( w^p \) simulates the steepest descent direction.
**Stopping criterion**

We propose the stopping criterion

\[
\max_p [v^p(t)]_{opt} \leq \epsilon_p
\]

where

\[
[v^p(t)]_{opt} = \left( \sum_{j=1}^{n} \begin{cases} 
0 & \text{if } x_j^p(t) = b_j \text{ and } v_j^p(t) \geq 0 \\
0 & \text{if } x_j^p(t) = a_j \text{ and } v_j^p(t) \leq 0 \\
(v_j^p(t))^2 & \text{otherwise}
\end{cases} \right)^{1/2}
\]
**Stopping criterion**

We propose the stopping criterion

$$\max_p [v^p(t)]_{opt} \leq \epsilon_p$$

where

$$[v^p(t)]_{opt} = \left( \sum_{j=1}^{n} \begin{cases} 0 & \text{if } x_j^p(t) = b_j \text{ and } v_j^p(t) \geq 0 \\ 0 & \text{if } x_j^p(t) = a_j \text{ and } v_j^p(t) \leq 0 \\ (v_j^p(t))^2 & \text{otherwise} \end{cases} \right)^{1/2}$$

The stopping criterion is based on the optimality conditions for the multi-local optimization problem.
Environment

- Implemented in the C programming language
- Interfaced with AMPL (www.ampl.com)
- Both methods soon available in the NEOS server
Environment

- Implemented in the C programming language
- **Interfaced with AMPL** ([www.ampl.com](http://www.ampl.com))
- Both methods soon available in the NEOS server
Environment

- Implemented in the C programming language
- Interfaced with AMPL (www.ampl.com)
- Both methods soon available in the NEOS server
## Test problems

<table>
<thead>
<tr>
<th>Problems</th>
<th>$n$</th>
<th>$N_{x^*}$</th>
<th>$f^*$</th>
<th>Problems</th>
<th>$n$</th>
<th>$N_{x^*}$</th>
<th>$f^*$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 b2</td>
<td>2</td>
<td>1</td>
<td>0.000E+00</td>
<td>17 rosenbrock5</td>
<td>5</td>
<td>1</td>
<td>0.000E+00</td>
</tr>
<tr>
<td>2 bohachevsky</td>
<td>2</td>
<td>1</td>
<td>0.000E+00</td>
<td>18 shekel10</td>
<td>4</td>
<td>1</td>
<td>-1.054E+01</td>
</tr>
</tbody>
</table>
Parameters

- For each problem, the optimizer was run 5 times with different initial particle positions and velocities (randomly chosen from the search domain)
- The algorithm terminates if the stopping criterion is met with $\epsilon_p = 0.01$ or the number of iterations exceeds $N_{t max} = 100000$
- Coefficients $\mu$ and $\nu$ were both set to 1.2
- The inertial parameter $\iota(t)$ was linearly scaled from 0.7 to 0.2 over a maximum of $N_{t max}$ iterations
- The swarm size is given by $\min(6^n, 100)$, where $n$ is the problem dimension.
Parameters

- For each problem, the optimizer was run 5 times with different initial particle positions and velocities (randomly chosen from the search domain)
- The algorithm terminates if the stopping criterion is met with $\epsilon_p = 0.01$ or the number of iterations exceeds $N_t^{max} = 100000$
- Coefficients $\mu$ and $\nu$ were both set to 1.2
- The inertial parameter $\iota(t)$ was linearly scaled from 0.7 to 0.2 over a maximum of $N_t^{max}$ iterations
- The swarm size is given by $\min(6^n, 100)$, where $n$ is the problem dimension.
Parameters

- For each problem, the optimizer was run 5 times with different initial particle positions and velocities (randomly chosen from the search domain).
- The algorithm terminates if the stopping criterion is met with \( \epsilon_p = 0.01 \) or the number of iterations exceeds \( N_{t}^{max} = 100000 \).
- **Coefficients** \( \mu \) and \( \nu \) were both set to 1.2.
- The inertial parameter \( \iota(t) \) was linearly scaled from 0.7 to 0.2 over a maximum of \( N_{t}^{max} \) iterations.
- The swarm size is given by \( \min(6^n, 100) \), where \( n \) is the problem dimension.
Parameters

- For each problem, the optimizer was run 5 times with different initial particle positions and velocities (randomly chosen from the search domain).
- The algorithm terminates if the stopping criterion is met with $\epsilon_p = 0.01$ or the number of iterations exceeds $N_t^{max} = 100000$.
- Coefficients $\mu$ and $\nu$ were both set to 1.2.
- The inertial parameter $\iota(t)$ was linearly scaled from 0.7 to 0.2 over a maximum of $N_t^{max}$ iterations.
- The swarm size is given by $\min(6^n, 100)$, where $n$ is the problem dimension.
Parameters

- For each problem, the optimizer was run 5 times with different initial particle positions and velocities (randomly chosen from the search domain)
- The algorithm terminates if the stopping criterion is met with $\epsilon_p = 0.01$ or the number of iterations exceeds $N_{t}{\text{max}} = 100000$
- Coefficients $\mu$ and $\nu$ were both set to 1.2
- The inertial parameter $\iota(t)$ was linearly scaled from 0.7 to 0.2 over a maximum of $N_{t}{\text{max}}$ iterations
- The swarm size is given by $\min(6^n, 100)$, where $n$ is the problem dimension.
### Numerical results

<table>
<thead>
<tr>
<th></th>
<th>Gradient version</th>
<th>Approximate descent direction version</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$F.,O.$</td>
<td>$N_{afe}$</td>
</tr>
<tr>
<td>1</td>
<td>100</td>
<td>3444343</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>2782058</td>
</tr>
<tr>
<td>3</td>
<td>100</td>
<td>1740823</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>1647820</td>
</tr>
<tr>
<td>5</td>
<td>100</td>
<td>283500</td>
</tr>
<tr>
<td>6</td>
<td>Not differentiable</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>20</td>
<td>3600000</td>
</tr>
<tr>
<td>8</td>
<td>20</td>
<td>10000000</td>
</tr>
<tr>
<td>9</td>
<td>100</td>
<td>10000000</td>
</tr>
<tr>
<td>10</td>
<td>40</td>
<td>10000000</td>
</tr>
<tr>
<td>11</td>
<td>100</td>
<td>963259</td>
</tr>
<tr>
<td>12</td>
<td>100</td>
<td>1171181</td>
</tr>
<tr>
<td>13</td>
<td>0</td>
<td>3600000</td>
</tr>
<tr>
<td>14</td>
<td>85</td>
<td>2952979</td>
</tr>
<tr>
<td>15</td>
<td>0</td>
<td>10000000</td>
</tr>
<tr>
<td>16</td>
<td>0</td>
<td>3600000</td>
</tr>
</tbody>
</table>
### Numerical results

<table>
<thead>
<tr>
<th>$F.O.$</th>
<th>$N_{afe}$</th>
<th>$N_{age}$</th>
<th>$f^*$</th>
<th>$f_{best}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>10000000</td>
<td>177</td>
<td>4.652E+03</td>
<td>2.393E+03</td>
</tr>
<tr>
<td>18</td>
<td>10000000</td>
<td>1850</td>
<td>-9.160E+00</td>
<td>-1.026E+01</td>
</tr>
<tr>
<td>19</td>
<td>10000000</td>
<td>2126</td>
<td>-7.801E+00</td>
<td>-8.760E+00</td>
</tr>
<tr>
<td>20</td>
<td>10000000</td>
<td>1909</td>
<td>-9.401E+00</td>
<td>-9.997E+00</td>
</tr>
<tr>
<td>21</td>
<td>3600000</td>
<td>335</td>
<td>-1.024E+02</td>
<td>-1.648E+02</td>
</tr>
<tr>
<td>22</td>
<td>1366222</td>
<td>973</td>
<td>-4.075E-01</td>
<td>-4.075E-01</td>
</tr>
<tr>
<td>23</td>
<td>3600000</td>
<td>570</td>
<td>-1.806E+01</td>
<td>-1.806E+01</td>
</tr>
<tr>
<td>24</td>
<td>3600000</td>
<td>194</td>
<td>-2.278E+02</td>
<td>-2.278E+02</td>
</tr>
<tr>
<td>25</td>
<td>3600000</td>
<td>167</td>
<td>-2.429E+03</td>
<td>-2.429E+03</td>
</tr>
<tr>
<td>26</td>
<td>3600000</td>
<td>81</td>
<td>-2.477E+04</td>
<td>-2.478E+04</td>
</tr>
<tr>
<td>27</td>
<td>3600000</td>
<td>58</td>
<td>1.607E+05</td>
<td>-2.436E+05</td>
</tr>
<tr>
<td>28</td>
<td>10000000</td>
<td>141</td>
<td>4.470E+02</td>
<td>3.102E+01</td>
</tr>
<tr>
<td>29</td>
<td>10000000</td>
<td>135</td>
<td>1.289E+05</td>
<td>7.935E+02</td>
</tr>
<tr>
<td>30</td>
<td>1433664</td>
<td>16314</td>
<td>8.325E-112</td>
<td>0.000E+00</td>
</tr>
<tr>
<td>31</td>
<td>10000000</td>
<td>313</td>
<td>1.997E-13</td>
<td>2.780E-21</td>
</tr>
<tr>
<td>32</td>
<td>10000000</td>
<td>160</td>
<td>8.338E+00</td>
<td>3.031E-04</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$F.O.$</th>
<th>$N_{afe}$</th>
<th>$f^*$</th>
<th>$f_{best}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>10005589</td>
<td>2.203E-01</td>
<td>1.327E-01</td>
</tr>
<tr>
<td>100</td>
<td>10004066</td>
<td>-1.052E+01</td>
<td>-1.052E+01</td>
</tr>
<tr>
<td>100</td>
<td>10004069</td>
<td>-1.037E+01</td>
<td>-1.039E+01</td>
</tr>
<tr>
<td>60</td>
<td>3600999</td>
<td>-1.867E+02</td>
<td>-1.867E+02</td>
</tr>
<tr>
<td>100</td>
<td>3601003</td>
<td>-2.278E+02</td>
<td>-2.278E+02</td>
</tr>
<tr>
<td>100</td>
<td>3601160</td>
<td>-2.429E+03</td>
<td>-2.429E+03</td>
</tr>
<tr>
<td>100</td>
<td>3601278</td>
<td>-2.478E+04</td>
<td>-2.478E+04</td>
</tr>
<tr>
<td>100</td>
<td>3601418</td>
<td>-2.493E+05</td>
<td>-2.493E+05</td>
</tr>
<tr>
<td>60</td>
<td>10009759</td>
<td>3.977E-02</td>
<td>2.506E-02</td>
</tr>
<tr>
<td>0</td>
<td>10016905</td>
<td>3.633E-01</td>
<td>2.404E-01</td>
</tr>
<tr>
<td>100</td>
<td>3601264</td>
<td>4.987E-07</td>
<td>4.464E-08</td>
</tr>
<tr>
<td>100</td>
<td>10005221</td>
<td>2.231E-04</td>
<td>6.612E-05</td>
</tr>
<tr>
<td>100</td>
<td>10006065</td>
<td>2.005E-03</td>
<td>1.186E-03</td>
</tr>
</tbody>
</table>
Conclusions

- We have presented a new multi-local optimization algorithm that evaluates multiple optimal solutions for multi-modal optimization problems

- Our MLPSO algorithm adapts the unimodal particle swarm optimizer using descent directions information to maintain diversity and to drive the particles to neighbor local minima

- Descent directions are obtained through the gradient vector or an heuristic method to produce an approximate descent direction.

- Experimental results indicate that the proposed algorithm is able to evaluate multiple optimal solutions with reasonable success rates.

- The use of a properly scaled gradient vector and the optimizer performance analysis on high-dimensional problems are issues under investigation.

- A inclusion of the proposed algorithm is planned to help a reduction type method for semi-infinite programming.
Conclusions

- We have presented a new multi-local optimization algorithm that evaluates multiple optimal solutions for multi-modal optimization problems.

- **Our MLPSO algorithm adapts the unimodal particle swarm optimizer using descent directions information to maintain diversity and to drive the particles to neighbor local minima.**

- Descent directions are obtained through the gradient vector or an heuristic method to produce an approximate descent direction.

- Experimental results indicate that the proposed algorithm is able to evaluate multiple optimal solutions with reasonable success rates.

- The use of a properly scaled gradient vector and the optimizer performance analysis on high-dimensional problems are issues under investigation.

- A inclusion of the proposed algorithm is planned to help a reduction type method for semi-infinite programming.
Conclusions

- We have presented a new multi-local optimization algorithm that evaluates multiple optimal solutions for multi-modal optimization problems.
- Our MLPSO algorithm adapts the unimodal particle swarm optimizer using descent directions information to maintain diversity and to drive the particles to neighbor local minima.
- Descent directions are obtained through the gradient vector or an heuristic method to produce an approximate descent direction.
- Experimental results indicate that the proposed algorithm is able to evaluate multiple optimal solutions with reasonable success rates.
- The use of a properly scaled gradient vector and the optimizer performance analysis on high-dimensional problems are issues under investigation.
- A inclusion of the proposed algorithm is planned to help a reduction type method for semi-infinite programming.
Conclusions

- We have presented a new multi-local optimization algorithm that evaluates multiple optimal solutions for multi-modal optimization problems.

- Our MLPSO algorithm adapts the unimodal particle swarm optimizer using descent directions information to maintain diversity and to drive the particles to neighbor local minima.

- Descent directions are obtained through the gradient vector or an heuristic method to produce an approximate descent direction.

- Experimental results indicate that the proposed algorithm is able to evaluate multiple optimal solutions with reasonable success rates.

- The use of a properly scaled gradient vector and the optimizer performance analysis on high-dimensional problems are issues under investigation.

- A inclusion of the proposed algorithm is planned to help a reduction type method for semi-infinite programming.
Conclusions

- We have presented a new multi-local optimization algorithm that evaluates multiple optimal solutions for multi-modal optimization problems.

- Our MLPSO algorithm adapts the unimodal particle swarm optimizer using descent directions information to maintain diversity and to drive the particles to neighbor local minima.

- Descent directions are obtained through the gradient vector or an heuristic method to produce an approximate descent direction.

- Experimental results indicate that the proposed algorithm is able to evaluate multiple optimal solutions with reasonable success rates.

- The use of a properly scaled gradient vector and the optimizer performance analysis on high-dimensional problems are issues under investigation.

- A inclusion of the proposed algorithm is planned to help a reduction type method for semi-infinite programming.
Conclusions

• We have presented a new multi-local optimization algorithm that evaluates multiple optimal solutions for multi-modal optimization problems.

• Our MLPSO algorithm adapts the unimodal particle swarm optimizer using descent directions information to maintain diversity and to drive the particles to neighbor local minima.

• Descent directions are obtained through the gradient vector or an heuristic method to produce an approximate descent direction.

• Experimental results indicate that the proposed algorithm is able to evaluate multiple optimal solutions with reasonable success rates.

• The use of a properly scaled gradient vector and the optimizer performance analysis on high-dimensional problems are issues under investigation.

• A inclusion of the proposed algorithm is planned to help a reduction type method for semi-infinite programming.
The end

email: aivaz@dps.uminho.pt
emgpf@dps.uminho.pt
Web http://www.norg.uminho.pt/