Global and multi-local optimization in the semi-infinite programming context

A. Ismael F. Vaz

Production and Systems Department
Engineering School
Minho University - Braga - Portugal
aivaz@dps.uminho.pt

Iberian Conference in Optimization, Coimbra

16-18 November 2006
Outline

1. Notation and motivation for global optimization
2. A motivating example
3. The particle swarm algorithm
4. Modification of PSOA for multi-local optimization
5. Numerical results in semi-infinite programming
6. Conclusions and future work
Outline

1. Notation and motivation for global optimization
2. A motivating example
3. The particle swarm algorithm
4. Modification of PSOA for multi-local optimization
5. Numerical results in semi-infinite programming
6. Conclusions and future work
1. Notation and motivation for global optimization

2. A motivating example

3. The particle swarm algorithm

4. Modification of PSOA for multi-local optimization

5. Numerical results in semi-infinite programming

6. Conclusions and future work
Outline

1. Notation and motivation for global optimization
2. A motivating example
3. The particle swarm algorithm
4. Modification of PSOA for multi-local optimization
5. Numerical results in semi-infinite programming
6. Conclusions and future work
Outline

1. Notation and motivation for global optimization
2. A motivating example
3. The particle swarm algorithm
4. Modification of PSOA for multi-local optimization
5. Numerical results in semi-infinite programming
6. Conclusions and future work
Outline

1. Notation and motivation for global optimization
2. A motivating example
3. The particle swarm algorithm
4. Modification of PSOA for multi-local optimization
5. Numerical results in semi-infinite programming
6. Conclusions and future work
Outline

1. Notation and motivation for global optimization
2. A motivating example
3. The particle swarm algorithm
4. Modification of PSOA for multi-local optimization
5. Numerical results in semi-infinite programming
6. Conclusions and future work
Notation and motivation for global optimization

General formulation - Nonlinear semi-infinite programming

Problem

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

(NLSIP)

- \( f(x) \) is the objective function
- \( g(x, t) \) is the infinite constraint function
- \( T \subset \mathbb{R}^p \) is, usually, a cartesian product of intervals
  \( ([\alpha_1, \beta_1] \times [\alpha_2, \beta_2] \times \ldots \times [\alpha_p, \beta_p]) \)

Note

A more general problem could be defined, but the extension is straightforward.
General formulation - Nonlinear semi-infinite programming

Problem

\[
\min_{x \in \mathbb{R}^n} f(x) \\
\text{s.t. } g(x, t) \leq 0 \\
\forall t \in T
\]

\[(NLSIP)\]

- \(f(x)\) is the objective function
- \(g(x, t)\) is the infinite constraint function
- \(T \subset \mathbb{R}^p\) is, usually, a cartesian product of intervals 
  \([\alpha_1, \beta_1] \times [\alpha_2, \beta_2] \times \ldots \times [\alpha_p, \beta_p]\)

Note

A more general problem could be defined, but the extension is straightforward.
General formulation - Nonlinear semi-infinite programming

Problem

\[
\min_{x \in \mathbb{R}^n} f(x) \quad \text{s.t.} \quad g(x, t) \leq 0 \quad \forall t \in T
\]

(NLSIP)

- \( f(x) \) is the objective function
- \( g(x, t) \) is the infinite constraint function
- \( T \subset \mathbb{R}^p \) is, usually, a cartesian product of intervals

([\alpha_1, \beta_1] \times [\alpha_2, \beta_2] \times \ldots \times [\alpha_p, \beta_p])

Note

A more general problem could be defined, but the extension is straightforward.
General formulation - Nonlinear semi-infinite programming

Problem

\[
\min_{x \in \mathbb{R}^n} f(x) \\
\text{s.t. } g(x, t) \leq 0 \\
\forall t \in T
\]

(NLSIP)

- \( f(x) \) is the objective function
- \( g(x, t) \) is the infinite constraint function
- \( T \subset \mathbb{R}^p \) is, usually, a cartesian product of intervals 
  \( ([\alpha_1, \beta_1] \times [\alpha_2, \beta_2] \times \ldots \times [\alpha_p, \beta_p]) \)

Note

A more general problem could be defined, but the extension is straightforward.
Problem

\[
\min_{x \in \mathbb{R}^n} f(x) \\
\text{s.t. } g(x, t) \leq 0 \\
\forall t \in T
\]

(NLSIP)

- \( f(x) \) is the objective function
- \( g(x, t) \) is the \textit{infinite} constraint function
- \( T \subset \mathbb{R}^p \) is, usually, a cartesian product of intervals 
  \([\alpha_1, \beta_1] \times [\alpha_2, \beta_2] \times \ldots \times [\alpha_p, \beta_p]\)

Note

A more general problem could be defined, but the extension is straightforward.
An very simple academic example \((n = 1\) and \(p = 1\))

Example

\[
\begin{align*}
\min_{x \in \mathbb{R}} & \quad x^2, \\
\text{s.t.} & \quad \frac{x}{t} \sin(t) - \frac{x}{10} \leq 0, \quad \forall t \in [2\pi, 10\pi]
\end{align*}
\]

\[
g(3, t) = \frac{3}{t} \sin(t) - \frac{3}{10}
\]

Feasibility

Is \(\bar{x} = 3\) feasible?
An very simple academic example \((n = 1 \text{ and } p = 1)\)

Example

\[
\begin{align*}
\min_{x \in \mathbb{R}} & \quad x^2, \\
\text{s.t.} & \quad \frac{x}{t} \sin(t) - \frac{x}{10} \leq 0, \quad \forall t \in [2\pi, 10\pi]
\end{align*}
\]

\(g(3, t) = \frac{3}{t} \sin(t) - \frac{3}{10}\)

Feasibility

Is \(\bar{x} = 3\) feasible?
Definition of stationary point

Let $x^* \in \mathbb{R}^n$ be a point such that

$$g(x^*, t) \leq 0, \ \forall t \in T,$$

and there exists $t^1, t^2, \ldots, t^{m^*} (\in T)$ and non negative numbers $\lambda^0_*, \lambda^1_*, \lambda^2_*, \ldots, \lambda^{m^*}_*$ such that

$$\lambda^0_* \nabla_x f(x^*) + \sum_{i=1}^{m^*} \lambda^i_* \nabla_x g(x^*, t^i) = 0.$$

with

$$g(x^*, t^i) = 0, \ i = 1, \ldots, m^*.$$

Then $x^*$ is a stationary point for the (NLSIP).
Where global (multi-local) optimization plays a role?

The $t^i$, $i = 1, \ldots, m^*$, points are global solutions of the problem

Multi-local problem (also called lower level problem)

\[
\max_{t \in T} g(x^*, t)
\]

- The simple check for feasibility requests the computation of the global solutions for the lower level problem (not completely true).
- In order to obtain global convergence for some methods the computation of all the global and local solutions for the lower level problem is necessary.
Where global (multi-local) optimization plays a role?

The \( t^i, i = 1, \ldots, m^* \), points are global solutions of the problem

**Multi-local problem (also called lower level problem)**

\[
\max_{t \in T} g(x^*, t)
\]

- The simple check for feasibility requests the computation of the global solutions for the lower level problem (not completely true).
- In order to obtain global convergence for some methods the computation of all the global and local solutions for the lower level problem is necessary.
Where global (multi-local) optimization plays a role?

The $t^i$, $i = 1, \ldots, m^*$, points are global solutions of the problem

Multi-local problem (also called lower level problem)

$$\max_{t \in T} g(x^*, t)$$

- The simple check for feasibility requests the computation of the global solutions for the lower level problem (not completely true).
- In order to obtain global convergence for some methods the computation of all the global and local solutions for the lower level problem is necessary.
Outline

1. Notation and motivation for global optimization
2. A motivating example
3. The particle swarm algorithm
4. Modification of PSOA for multi-local optimization
5. Numerical results in semi-infinite programming
6. Conclusions and future work
Motivation

- A great number of valuable products are produced using fermentation processes and thus optimizing such processes is of great economic importance.
- Fermentation modeling process involves, in general, highly nonlinear and complex differential equations.
- Often optimizing these processes results in control optimization problems for which an analytical solution is not possible.
A motivating example

Motivation

A great number of valuable products are produced using fermentation processes and thus optimizing such processes is of great economic importance.

Fermentation modeling process involves, in general, highly nonlinear and complex differential equations.

Often optimizing these processes results in control optimization problems for which an analytical solution is not possible.
Motivation

- A great number of valuable products are produced using fermentation processes and thus optimizing such processes is of great economic importance.
- Fermentation modeling process involves, in general, highly nonlinear and complex differential equations.
- Often optimizing these processes results in control optimization problems for which an analytical solution is not possible.
The control problem

The optimal control problem is described by a set of differential equations
\[ \dot{\chi} = h(\chi, u, t), \quad \chi(t_0) = \chi^0, \quad t_0 \leq t \leq t_f, \]
where \( \chi \) represent the state variables and \( u \) the control variables.

The performance index \( J \) can be generally stated as
\[
J(t_f) = \varphi(\chi(t_f), t_f) + \int_{t_0}^{t_f} \phi(\chi, u, t) \, dt,
\]
where \( \varphi \) is the performance index of the state variables at final time \( t_f \)
and \( \phi \) is the integrated performance index during the operation.

Additional constraints that often reflect some physical limitation of the system can be imposed.
The optimal control problem is described by a set of differential equations \( \dot{\chi} = h(\chi, u, t) \), \( \chi(t^0) = \chi^0 \), \( t^0 \leq t \leq t^f \), where \( \chi \) represent the state variables and \( u \) the control variables.

The performance index \( J \) can be generally stated as

\[
J(t^f) = \varphi(\chi(t^f), t^f) + \int_{t^0}^{t^f} \phi(\chi, u, t) dt,
\]

where \( \varphi \) is the performance index of the state variables at final time \( t^f \) and \( \phi \) is the integrated performance index during the operation.

Additional constraints that often reflect some physical limitation of the system can be imposed.
The control problem

- The optimal control problem is described by a set of differential equations \( \dot{\chi} = h(\chi, u, t) \), \( \chi(t^0) = \chi^0 \), \( t^0 \leq t \leq t^f \), where \( \chi \) represents the state variables and \( u \) the control variables.

- The performance index \( J \) can be generally stated as

\[
J(t^f) = \varphi(\chi(t^f), t^f) + \int_{t^0}^{t^f} \phi(\chi, u, t) \, dt,
\]

where \( \varphi \) is the performance index of the state variables at final time \( t^f \) and \( \phi \) is the integrated performance index during the operation.

- Additional constraints that often reflect some physical limitation of the system can be imposed.
The control problem

The general maximization problem \((P)\) can be posed as

\[
\begin{align*}
\text{max} & \quad J(t_f) \\
\text{s.t.} & \quad \dot{\chi} = h(\chi, u, t) \\
& \quad \underline{\chi} \leq \chi(t) \leq \bar{\chi}, \\
& \quad \underline{u} \leq u(t) \leq \bar{u}, \\
& \quad \forall t \in [t_0, t_f]
\end{align*}
\tag{1-5}
\]

Where the state constraints (3) and control constraints (4) are to be understood as componentwise inequalities.

How we addressed problem \((P)\)?

The control problem

The general maximization problem \((P)\) can be posed as

\[
\begin{align*}
\text{max} & \quad J(t^f) \\
\text{s.t.} & \quad \dot{x} = h(x, u, t) \\
& \quad x \leq x(t) \leq \bar{x}, \\
& \quad u \leq u(t) \leq \bar{u}, \\
& \quad \forall t \in [t^0, t^f]
\end{align*}
\]  

Where the state constraints \((3)\) and control constraints \((4)\) are to be understood as componentwise inequalities.

How we addressed problem \((P)\)?
Approaches - Fed trajectory $u(t)$ approximated by a Linear spline $w(t)$.

- Penalty function for state constraints
- The multi-local (getting all local optima) problem is easy to solve

**Objective function**

$$\hat{J}(t^f) = \begin{cases} 
J(t^f) & \text{if } \underline{\chi} \leq \chi(t) \leq \overline{\chi}, \\
\forall t \in [t^0, t^f] \\
-\infty & \text{otherwise}
\end{cases}$$

**State constraints**

$$u \leq w(t^i) \leq \bar{u}, \ i = 1, \ldots, n$$

Where $t^i$ are the spline knots.

The maximization NLP problem is

$$\max_{w(t^i)} \hat{J}(t^f), \ s.t. \ u \leq w(t^i) \leq \bar{u}, \ i = 1, \ldots, n$$
A motivating example

**Used approaches**

**Appropaches** - Fed trajectory $u(t)$ approximated by a Linear spline $w(t)$.

- Penalty function for state constraints
- The multi-local (getting all local optima) problem is easy to solve

**Objective function**

\[
\hat{J}(t^f) = \begin{cases} 
J(t^f) & \text{if } \chi \leq \chi(t) \leq \bar{\chi}, \\
-\infty & \text{otherwise}
\end{cases}
\] 

Where $t_i$ are the spline knots.

**State constraints**

\[u \leq w(t^i) \leq \bar{u}, \quad i = 1, \ldots, n\]

The maximization NLP problem is

\[
\max_{w(t^i)} \hat{J}(t^f), \quad \text{s.t.} \quad u \leq w(t^i) \leq \bar{u}, \quad i = 1, \ldots, n
\]
A motivating example

**Approaches** - Fed trajectory $u(t)$ approximated by a Linear spline $w(t)$.

- Penalty function for state constraints
- The multi-local (getting all local optima) problem is easy to solve

**Objective function**

$$\hat{J}(t^f) = \begin{cases} J(t^f) & \text{if } \chi \leq \chi(t) \leq \bar{\chi}, \\
\forall t \in [t^0, t^f] \\
-\infty & \text{otherwise} \end{cases}$$

**State constraints**

$$u \leq w(t^i) \leq \bar{u}, \quad i = 1, \ldots, n$$

Where $t^i$ are the spline knots.

The maximization NLP problem is

$$\max_{w(t^i)} \hat{J}(t^f), \quad s.t. \quad u \leq w(t^i) \leq \bar{u}, \quad i = 1, \ldots, n$$
A motivating example

Approaches - Fed trajectory $u(t)$ approximated by a Cubic spline $s(t)$.

- Penalty function for state constraints
- The multi-local (getting all local optima) problem is hard to solve
- No of-the-shelf software to address this problem
- A new penalty function defined for control constraints

**Objective function**

\[
\hat{J}(t^f) = \begin{cases} 
J(t^f) & \text{if } \chi \leq \chi(t) \leq \overline{\chi}, \\
\forall t \in [t^0, t^f] \\
-\infty & \text{otherwise}
\end{cases}
\]

**New objective function**

\[
\bar{J}(t^f) = \begin{cases} 
\hat{J}(t^f) & \text{if } u \leq w(t) \leq \overline{u}, \\
\forall t \in [t^0, t^f] \\
-\infty & \text{otherwise}
\end{cases}
\]
A motivating example

Approaches - Fed trajectory $u(t)$ approximated by a Cubic spline $s(t)$.

- Penalty function for state constraints
- The multi-local (getting all local optima) problem is hard to solve
- No off-the-shelf software to address this problem
- A new penalty function defined for control constraints

**Objective function**

$$
\hat{J}(t^f) = \begin{cases} 
J(t^f) & \text{if } \underline{\chi} \leq \chi(t) \leq \bar{\chi}, \\
\forall t \in [t^0, t^f] & \\
-\infty & \text{otherwise}
\end{cases}
$$

**New objective function**

$$
\bar{J}(t^f) = \begin{cases} 
\hat{J}(t^f) & \text{if } u \leq w(t) \leq \bar{u}, \\
\forall t \in [t^0, t^f] & \\
-\infty & \text{otherwise}
\end{cases}
$$
Approaches - Fed trajectory $u(t)$ approximated by a Cubic spline $s(t)$.

- Penalty function for state constraints
- The multi-local (getting all local optima) problem is hard to solve
- No of-the-shelf software to address this problem
- A new penalty function defined for control constraints

**Objective function**

\[
\hat{J}(t^f) = \begin{cases} 
J(t^f) & \text{if } \underline{x} \leq x(t) \leq \overline{x}, \\
& \forall t \in [t^0, t^f] \\
-\infty & \text{otherwise}
\end{cases}
\]

**New objective function**

\[
\bar{J}(t^f) = \begin{cases} 
\hat{J}(t^f) & \text{if } u \leq w(t) \leq \overline{u}, \\
& \forall t \in [t^0, t^f] \\
-\infty & \text{otherwise}
\end{cases}
\]
A motivating example

**Approaches** - Fed trajectory $u(t)$ approximated by a Cubic spline $s(t)$.

- Penalty function for state constraints
- The multi-local (getting all local optima) problem is hard to solve
- No off-the-shelf software to address this problem
- A new penalty function defined for control constraints

**Objective function**

$$
\hat{J}(t^f) = \begin{cases} 
  J(t^f) & \text{if } \chi \leq \chi(t) \leq \overline{\chi}, \\
  -\infty & \text{otherwise}
\end{cases}, \quad \forall t \in [t^0, t^f]
$$

**New objective function**

$$
\bar{J}(t^f) = \begin{cases} 
  \hat{J}(t^f) & \text{if } u \leq w(t) \leq \overline{u}, \\
  -\infty & \text{otherwise}
\end{cases}, \quad \forall t \in [t^0, t^f]
$$
Implementation details

The AMPL modeling language:
- was used to model five optimal control problems
- dynamic external library facility was used to solve the ordinary differentiable equations

AMPL - A Modeling Programming Language
www.ampl.com

The ordinary differentiable equations were solved using the CVODE software package.
http://www.llnl.gov/casc/sundials/

A stochastic algorithm based on particle swarm was used to solve the non-differentiable optimization problem. We address this algorithm later on.
The AMPL modeling language:
- was used to model five optimal control problems
- dynamic external library facility was used to solve the ordinary differentiable equations

AMPL - A Modeling Programming Language
www.ampl.com

The ordinary differentiable equations were solved using the CVODE software package.
http://www.llnl.gov/casc/sundials/

A stochastic algorithm based on particle swarm was used to solve the non-differentiable optimization problem. We address this algorithm later on.
Implementation details

- The AMPL modeling language:
  - was used to model five optimal control problems
  - dynamic external library facility was used to solve the ordinary differentiable equations

  AMPL - A Modeling Programming Language
  www.ampl.com

- The ordinary differentiable equations were solved using the CVODE software package.

  http://www.llnl.gov/casc/sundials/

- A stochastic algorithm based on particle swarm was used to solve the non-differentiable optimization problem. We address this algorithm later on.
Implementation details

- The AMPL modeling language:
  - was used to model five optimal control problems
  - dynamic external library facility was used to solve the ordinary differentiable equations

  AMPL - A Modeling Programming Language
  www.ampl.com

- The ordinary differentiable equations were solved using the CVODE software package.

  http://www.llnl.gov/casc/sundials/

- A stochastic algorithm based on particle swarm was used to solve the non-differentiable optimization problem. We address this algorithm later on.
Implementation details

- The AMPL modeling language:
  - was used to model five optimal control problems
  - dynamic external library facility was used to solve the ordinary differentiable equations

  AMPL - A Modeling Programming Language
  www.ampl.com

- The ordinary differentiable equations were solved using the CVODE software package.

  http://www.llnl.gov/casc/sundials/

- A stochastic algorithm based on particle swarm was used to solve the non-differentiable optimization problem. We address this algorithm later on.
The problems set

- We obtained numerical results for five case studies.

- Problem

  - penicillin refers to a problem of fed-batch fermentation process where the optimal feed trajectory is to be computed while the penicillin production is to be maximized.
  
  - ethanol refers to a similar optimal control problem where the ethanol production is to be maximized.
  
  - chemotherapy is the only optimal control problem that does not refers to a fed-batch fermentation process. It is a problem of drug administration in chemotherapy. The optimal trajectory to be computed is the quantity of drug that must be present in order to achieve a specified tumor reduction.
  
  - hprotein optimal control problem is to compute a unique trajectory (substrate to be fed) problem. hprotein includes also a trajectory for an inducer. Both problems refer to a maximization for protein production.
We obtained numerical results for five case studies.

**Problem**

- **Penicillin** refers to a problem of fed-batch fermentation process where the optimal feed trajectory is to be computed while the penicillin production is to be maximized.

- **Ethanol** refers to a similar optimal control problem where the ethanol production is to be maximized.

- **Chemotherapy** is the only optimal control problem that does not refer to a fed-batch fermentation process. It is a problem of drug administration in chemotherapy. The optimal trajectory to be computed is the quantity of drug that must be present in order to achieve a specified tumor reduction.

- **Hprotein** optimal control problem is to compute a unique trajectory (substrate to be fed) problem. **Rprotein** includes also a trajectory for an inducer. Both problems refer to a maximization for protein production.
The problems set

- We obtained numerical results for five case studies.

**Problem**

- **penicillin** refers to a problem of fed-batch fermentation process where the optimal feed trajectory is to be computed while the penicillin production is to be maximized.

- **ethanol** refers to a similar optimal control problem where the ethanol production is to be maximized.

- **chemotherapy** is the only optimal control problem that does not refer to a fed-batch fermentation process. It is a problem of drug administration in chemotherapy. The optimal trajectory to be computed is the quantity of drug that must be present in order to achieve a specified tumor reduction.

- **hprotein** optimal control problem is to compute a unique trajectory (substrate to be fed) problem. **rprotein** includes also a trajectory for an inducer. Both problems refer to a maximization for protein production.
The problems set

- We obtained numerical results for five case studies.

- Problem
  - **penicillin** refers to a problem of fed-batch fermentation process where the optimal feed trajectory is to be computed while the penicillin production is to be maximized.
  - **ethanol** refers to a similar optimal control problem where the ethanol production is to be maximized.
  - **chemotherapy** is the only optimal control problem that does not refer to a fed-batch fermentation process. It is a problem of drug administration in chemotherapy. The optimal trajectory to be computed is the quantity of drug that must be present in order to achieve a specified tumor reduction.
  - **hprotein** optimal control problem is to compute a unique trajectory (substrate to be fed) problem. **rprotein** includes also a trajectory for an inducer. Both problems refer to a maximization for protein production.
We obtained numerical results for five case studies.

Problem

- **penicillin** refers to a problem of fed-batch fermentation process where the optimal feed trajectory is to be computed while the penicillin production is to be maximized.
- **ethanol** refers to a similar optimal control problem where the ethanol production is to be maximized.
- **chemotherapy** is the only optimal control problem that does not refer to a fed-batch fermentation process. It is a problem of drug administration in chemotherapy. The optimal trajectory to be computed is the quantity of drug that must be present in order to achieve a specified tumor reduction.
- **hprotein** optimal control problem is to compute a unique trajectory (substrate to be fed) problem. **rprotein** includes also a trajectory for an inducer. Both problems refer to a maximization for protein production.
The problems set

- We obtained numerical results for five case studies.
- Problem
  - *penicillin* refers to a problem of fed-batch fermentation process where the optimal feed trajectory is to be computed while the penicillin production is to be maximized.
  - *ethanol* refers to a similar optimal control problem where the ethanol production is to be maximized.
  - *chemotherapy* is the only optimal control problem that does not refer to a fed-batch fermentation process. It is a problem of drug administration in chemotherapy. The optimal trajectory to be computed is the quantity of drug that must be present in order to achieve a specified tumor reduction.
  - *hprotein* optimal control problem is to compute a unique trajectory (substrate to be fed) problem. *rprotein* includes also a trajectory for an inducer. Both problems refer to a maximization for protein production.
Characteristics and parameters

- The time displacement ($h_i$) are fixed while the optimal trajectory values are to be approximated.
- Particle swarm is a population based optimization algorithm and a population size of 60 was used with a maximum of 1000 iterations.
- Since a stochastic algorithm was used we performed 10 runs of the solver and the best solution is reported.
Characteristics and parameters

- The time displacement \((h_i)\) are fixed while the optimal trajectory values are to be approximated.
- Particle swarm is a population based optimization algorithm and a population size of 60 was used with a maximum of 1000 iterations.
- Since a stochastic algorithm was used we performed 10 runs of the solver and the best solution is reported.
Characteristics and parameters

- The time displacement \((h_i)\) are fixed while the optimal trajectory values are to be approximated.
- Particle swarm is a population based optimization algorithm and a population size of 60 was used with a maximum of 1000 iterations.
- Since a stochastic algorithm was used we performed 10 runs of the solver and the best solution is reported.
## Numerical results

<table>
<thead>
<tr>
<th>Problema</th>
<th>NT</th>
<th>n</th>
<th>$t_f$</th>
<th>Cubic $J(t_f)$</th>
<th>Linear $\hat{J}(t_f)$</th>
<th>Literature $\bar{J}(t_f)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>penicillin</td>
<td>1</td>
<td>5</td>
<td>132.00</td>
<td>87.70</td>
<td>88.29</td>
<td>87.99</td>
</tr>
<tr>
<td>ethanol</td>
<td>1</td>
<td>5</td>
<td>61.20</td>
<td>20550.70</td>
<td>20379.50</td>
<td>20839.00</td>
</tr>
<tr>
<td>chemotherapy</td>
<td>1</td>
<td>4</td>
<td>84.00</td>
<td>15.75</td>
<td>16.83</td>
<td>14.48</td>
</tr>
<tr>
<td>hprotein</td>
<td>1</td>
<td>5</td>
<td>15.00</td>
<td>38.86</td>
<td>32.73</td>
<td>32.40</td>
</tr>
<tr>
<td>rprotein</td>
<td>2</td>
<td>5</td>
<td>10.00</td>
<td>0.13</td>
<td>0.12</td>
<td>0.16</td>
</tr>
</tbody>
</table>

$J(t_f) = \hat{J}(t_f) = \bar{J}(t_f)$, for all feasible points - splines

Similar results between approaches. A new solution for the ethanol case.
Plots - Linear spline approximation - ethanol case

State profile

- $X_1$ - Cell mass
- $X_2$ - Substrate
- $X_3$ - Product
- $X_4$ - Volume

Control profile

- $u$ - Substrate feed
Plots - Cubic spline approximation - Similar result

Control profile

State profile

- $X_1$ – Cell mass
- $X_2$ – Substrate
- $X_3$ – Product
- $X_4$ – Volume

Ismael Vaz (UMinho - PT)
Plots - Cubic spline approximation - Best result

Control profile

State profile

- X1 − Cell mass
- X2 − Substrate
- X3 − Product
- X4 − Volume
Some intermediate conclusions and future work

Conclusions

- Viability of the cubic spline approach on fed-batch optimal control.
- Shown numerical results with particle swarm
- Similar numerical results with the two approaches

Future work

- Numerical experiments with the E. coli bacteria
- Laboratory confirmation of the obtained results (a lab bioreactor will be available)
- Laboratory confirmation of the two approaches and we expect the cubic approach to obtain a lower gap between simulated and real performance.
Conclusions

- Viability of the cubic spline approach on fed-batch optimal control.
- Shown numerical results with particle swarm
- Similar numerical results with the two approaches

Future work

- Numerical experiments with the E. coli bacteria
- Laboratory confirmation of the obtained results (a lab bioreactor will be available)
- Laboratory confirmation of the two approaches and we expect the cubic approach to obtain a lower gap between simulated and real performance.
Some intermediate conclusions and future work

**Conclusions**

- Viability of the cubic spline approach on fed-batch optimal control.
- Shown numerical results with particle swarm
- Similar numerical results with the two approaches

**Future work**

- Numerical experiments with the E. coli bacteria
- Laboratory confirmation of the obtained results (a lab bioreactor will be available)
- Laboratory confirmation of the two approaches and we expect the cubic approach to obtain a lower gap between simulated and real performance
Some intermediate conclusions and future work

Conclusions

- Viability of the cubic spline approach on fed-batch optimal control.
- Shown numerical results with particle swarm
- Similar numerical results with the two approaches

Future work

- Numerical experiments with the E. coli bacteria
- Laboratory confirmation of the obtained results (a lab bioreactor will be available)
- Laboratory confirmation of the two approaches and we expect the cubic approach to obtain a lower gap between simulated and real performance.
Some intermediate conclusions and future work

**Conclusions**

- Viability of the cubic spline approach on fed-batch optimal control.
- Shown numerical results with particle swarm
- Similar numerical results with the two approaches

**Future work**

- Numerical experiments with the *E. coli* bacteria
- Laboratory confirmation of the obtained results (a lab bioreactor will be available)
- Laboratory confirmation of the two approaches and we expect the cubic approach to obtain a lower gap between simulated and real performance.
Some intermediate conclusions and future work

Conclusions

- Viability of the cubic spline approach on fed-batch optimal control.
- Shown numerical results with particle swarm
- Similar numerical results with the two approaches

Future work

- Numerical experiments with the *E. coli* bacteria
- Laboratory confirmation of the obtained results (a lab bioreactor will be available)
- Laboratory confirmation of the two approaches and we expect the cubic approach to obtain a lower gap between simulated and real performance.
Some intermediate conclusions and future work

Conclusions

- Viability of the cubic spline approach on fed-batch optimal control.
- Shown numerical results with particle swarm.
- Similar numerical results with the two approaches.

Future work

- Numerical experiments with the *E. coli* bacteria.
- Laboratory confirmation of the obtained results (a lab bioreactor will be available).
- Laboratory confirmation of the two approaches and we expect the cubic approach to obtain a lower gap between simulated and real performance.
Some intermediate conclusions and future work

Conclusions

- Viability of the cubic spline approach on fed-batch optimal control.
- Shown numerical results with particle swarm
- Similar numerical results with the two approaches

Future work

- Numerical experiments with the *E. coli* bacteria
- Laboratory confirmation of the obtained results (a lab bioreactor will be available)
- Laboratory confirmation of the two approaches and we expect the cubic approach to obtain a lower gap between simulated and real performance.
Outline

1. Notation and motivation for global optimization
2. A motivating example
3. The particle swarm algorithm
4. Modification of PSOA for multi-local optimization
5. Numerical results in semi-infinite programming
6. Conclusions and future work
We intended to solve the following global optimization problem with a particle swarm algorithm.

**Global optimization problem**

\[
\max_{t \in T} \bar{g}(t) \equiv g(\bar{x}, t)
\]

with \( T \in \mathbb{R}^p \).
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 $\varphi$, at time instant $k$, is represented by its current position ($t^\varphi(k)$), its best ever position ($y^\varphi(k)$) and its traveling velocity ($v^\varphi(k)$).
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 $\varphi$, at time instant $k$, is represented by its current position ($t^\varphi(k)$), its best ever position ($y^\varphi(k)$) and its traveling velocity ($v^\varphi(k)$).
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 $\phi$, at time instant $k$, is represented by its current position ($t^\phi(k)$), its best ever position ($y^\phi(k)$) and its traveling velocity ($v^\phi(k)$).
The new travel position and velocity

The new particle position is updated by

**Update position**

\[ t^p(k + 1) = t^p(k) + v^p(k + 1), \]

where \( v^p(k + 1) \) is the new velocity given by

**Update velocity**

\[ v^p_j(k + 1) = \iota(k)v^p_j(k) + \mu_1j(k)\left( y^p_j(k) - t^p_j(k) \right) + \nu_2j(k)\left( \hat{y}_j(k) - t^p_j(k) \right), \]

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

- \( \iota(k) \) is a weighting factor (inertial)
- \( \mu \) is the cognition parameter and \( \nu \) is the social parameter
- \( \omega_1(k) \) and \( \omega_2(k) \) are random numbers drawn from the uniform \((0, 1)\) distribution.
The new travel position and velocity

The new particle position is updated by

**Update position**

\[ \mathbf{t}^\varphi(k+1) = \mathbf{t}^\varphi(k) + \mathbf{v}^\varphi(k+1), \]

where \( \mathbf{v}^\varphi(k+1) \) is the new velocity given by

**Update velocity**

\[ v_j^\varphi(k+1) = \iota(k)v_j^\varphi(k) + \mu \omega_{1j}(k) \left( y_j^\varphi(k) - t_j^\varphi(k) \right) + \nu \omega_{2j}(k) \left( \hat{y}_j(k) - t_j^\varphi(k) \right), \]

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

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

The new particle position is updated by

**Update position**

\[ t^p(k + 1) = t^p(k) + v^p(k + 1), \]

where \( v^p(k + 1) \) is the new velocity given by

**Update velocity**

\[ v^p_j(k + 1) = \iota(k)v^p_j(k) + \mu \omega_1 j(k) (y^p_j(k) - t^p_j(k)) + \nu \omega_2 j(k) (\hat{y}_j(k) - t^p_j(k)), \]

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

- \( \iota(k) \) is a weighting factor (inertial)
- \( \mu \) is the cognition parameter and \( \nu \) is the social parameter
- \( \omega_1 j(k) \) and \( \omega_2 j(k) \) are random numbers drawn from the uniform \((0, 1)\) distribution.
The new travel position and velocity

The new particle position is updated by

**Update position**

\[ t^{\theta}(k + 1) = t^{\theta}(k) + v^{\theta}(k + 1), \]

where \( v^{\theta}(k + 1) \) is the new velocity given by

**Update velocity**

\[ v_j^{\theta}(k+1) = \iota(k)v_j^{\theta}(k) + \mu \omega_{1j}(k) \left( y_j^{\theta}(k) - t_j^{\theta}(k) \right) + \nu \omega_{2j}(k) \left( \hat{y}_j(k) - t_j^{\theta}(k) \right), \]

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

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

The new particle position is updated by

**Update position**

\[
\tau^p(k + 1) = \tau^p(k) + v^p(k + 1),
\]

where \(v^p(k + 1)\) is the new velocity given by

**Update velocity**

\[
v^p_j(k + 1) = \nu(k)v^p_j(k) + \mu \omega_1 j(k) \left( y^p_j(k) - \tau^p_j(k) \right) + \nu \omega_2 j(k) \left( \hat{y}_j(k) - \tau^p_j(k) \right),
\]

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

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

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

\[
\hat{y}(k) \in \arg\min_{a \in A} \bar{g}(a)
\]

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

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

Note

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

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

\[
\hat{y}(k) \in \arg \min_{a \in A} \bar{g}(a)
\]

\( A = \{ y^1(k), \ldots, y^s(k) \} \).

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

Note

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.

🌟 Good
- Easy to implement.
- Easy to parallelize.
- Easy to handle discrete variables.
- Only uses objective function evaluations.

🌟 Not so good
- Slow rate of convergence near an optimum.
- Quite large number of function evaluations.
- In the presence of several global optima the algorithm may not converge.
Features

Population based algorithm.

- **Good**
  - Easy to implement.
  - Easy to parallelize.
  - Easy to handle discrete variables.
  - Only uses objective function evaluations.

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

Population based algorithm.

**Good**
- Easy to implement.
- Easy to parallelize.
- Easy to handle discrete variables.
- Only uses objective function evaluations.

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

Population based algorithm.

⭐️ Good
- Easy to implement.
- Easy to parallelize.
- Easy to handle discrete variables.
- Only uses objective function evaluations.

⭐️ Not so good
- Slow rate of convergence near an optimum.
- Quite large number of function evaluations.
- In the presence of several global optima the algorithm may not converge.
Features

Population based algorithm.

- **Good**
  - Easy to implement.
  - Easy to parallelize.
  - Easy to handle discrete variables.
  - Only uses objective function evaluations.

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

Population based algorithm.

- **Good**
  - Easy to implement.
  - Easy to parallelize.
  - Easy to handle discrete variables.
  - Only uses objective function evaluations.

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

Population based algorithm.

* Good
  - Easy to implement.
  - Easy to parallelize.
  - Easy to handle discrete variables.
  - Only uses objective function evaluations.

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

Population based algorithm.

🌟 Good
- Easy to implement.
- Easy to parallelize.
- Easy to handle discrete variables.
- Only uses objective function evaluations.

🌟 Not so good
- Slow rate of convergence near an optimum.
- Quite large number of function evaluations.
- In the presence of several global optima the algorithm may not converge.
Population based algorithm.

**Good**
- Easy to implement.
- Easy to parallelize.
- Easy to handle discrete variables.
- Only uses objective function evaluations.

**Not so good**
- Slow rate of convergence near an optimum.
- Quite large number of function evaluations.
- In the presence of several global optima the algorithm may not converge.
Properties

- With a proper selection of the algorithm parameters finite termination of the algorithm can be established, in a probabilistic sense.

- Convergence for a global optimum is not guaranteed by this simple version of the particle swarm algorithm, but some adaption can be introduce to guarantee it.
Properties

- With a proper selection of the algorithm parameters finite termination of the algorithm can be established, in a probabilistic sense.
- Convergence for a global optimum is not guaranteed by this simple version of the particle swarm algorithm, but some adaption can be introduce to guarantee it.
Outline

1. Notation and motivation for global optimization
2. A motivating example
3. The particle swarm algorithm
4. Modification of PSOA for multi-local optimization
5. Numerical results in semi-infinite programming
6. Conclusions and future work
Multi-local revisited

Given $\bar{x}$ the multi-local optimization problem is defined as

$$\max_{t \in T} g(\bar{x}, t) \equiv \bar{g}(t)$$

with $T \in \mathbb{R}^n$.

The multi-local concept

All the global and local optima are to be computed.

Some characteristics

These problems are mostly differentiable and the objective function computation is costless.
Multi-local revisited

Given $\bar{x}$ the multi-local optimization problem is defined as

\[
\max_{t \in T} g(\bar{x}, t) \equiv \bar{g}(t)
\]

with $T \in \mathbb{R}^n$.

The multi-local concept

All the global and local optima are to be computed.

Some characteristics

These problems are mostly differentiable and the objective function computation is costless.
Multi-local revisited

Given \( \bar{x} \) the multi-local optimization problem is defined as

\[
\max_{t \in T} g(\bar{x}, t) \equiv \bar{g}(t)
\]

with \( T \in \mathbb{R}^n \).

The multi-local concept

All the global and local optima are to be computed.

Some characteristics

These problems are mostly differentiable and the objective function computation is costless.
PSP with the steepest ascent direction

The new particle position update equation is kept while the new velocity equation is given by

\[ v_j^o(k+1) = \nu(k)v_j^o(k) + \mu \omega_1 j(k) \left( y_j^o(k) - t_j^o(k) \right) + \nu \omega_2(t) \left( \nabla_j \bar{g}(y_j^o(k)) \right), \]

for \( j = 1, \ldots, p \), where \( \nabla \bar{g}(t) \) is the gradient of the objective function.

Each particle uses the steepest ascent direction computed at each particle best position \( (y^o(k)) \).

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

Other approach is to use

Ascent velocity formula

\[ w^\varphi = \frac{1}{\sum_{j=1}^{m} |\bar{g}(z_j^\varphi) - \bar{g}(y^\varphi)|} \sum_{j=1}^{m} (\bar{g}(z_j^\varphi) - \bar{g}(y^\varphi)) \frac{(z_j^\varphi - y^\varphi)}{\|z_j^\varphi - y^\varphi\|} \]

as an ascent direction at \( y^\varphi \), in the velocity equation, to overcome the need to compute the gradient.

Where

- \( y^\varphi \) is the best position of particle \( \varphi \)
- \( \{z_j^\varphi\}_{j=1}^{m} \) is a set of \( m \) (random) points close to \( y^p \),

Under certain conditions \( w^\varphi \) simulates the steepest ascent direction.
Stopping criterion

We propose the stopping criterion

**Minimum velocity attained**

\[
\max_{\varphi}[v^\varphi(k)]_{opt} \leq \epsilon_{\varphi}
\]

where

**Constrained velocity**

\[
[v^\varphi(k)]_{opt} = \left( \sum_{j=1}^{p} \begin{cases} 
0 & \text{if } t_j^\varphi(k) = \beta_j \text{ and } v_j^\varphi(k) \geq 0 \\
0 & \text{if } t_j^\varphi(k) = \alpha_j \text{ and } v_j^\varphi(k) \leq 0 \\
\left(v_j^\varphi(k)\right)^2 & \text{otherwise}
\end{cases} \right)^{1/2}
\]

The stopping criterion is based on the optimality conditions for the multi-local optimization problem.
We have coined the solver as MLOCPSOA (Multi-Local Optimization Particle Swarm Algorithm)

- Implemented in the C programming language
- Interfaced with AMPL (www.ampl.com)
Environment

- We have coined the solver as MLOCPSOA (Multi-Local Optimization Particle Swarm Algorithm)
- Implemented in the C programming language
- Interfaced with AMPL (www.ampl.com)
Environment

- We have coined the solver as MLOCPSOA (Multi-Local Optimization Particle Swarm Algorithm)
- Implemented in the C programming language
- Interfaced with AMPL (www.ampl.com)
## Test problems set

<table>
<thead>
<tr>
<th>Problems</th>
<th>$p$</th>
<th>$N_{t^*}$</th>
<th>$\bar{g}^*$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1    b2</td>
<td>2</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>
</tr>
<tr>
<td>3    branin</td>
<td>2</td>
<td>3</td>
<td>3.979E-01</td>
</tr>
<tr>
<td>4    dejoung</td>
<td>3</td>
<td>1</td>
<td>0.000E+00</td>
</tr>
<tr>
<td>5    easom</td>
<td>2</td>
<td>1</td>
<td>-1.000E+00</td>
</tr>
<tr>
<td>6    f1</td>
<td>30</td>
<td>1</td>
<td>-1.257E+04</td>
</tr>
<tr>
<td>7    goldprice</td>
<td>2</td>
<td>1</td>
<td>3.000E+00</td>
</tr>
<tr>
<td>8    griewank</td>
<td>6</td>
<td>1</td>
<td>0.000E+00</td>
</tr>
<tr>
<td>9    hartmann3</td>
<td>3</td>
<td>1</td>
<td>-3.863E+00</td>
</tr>
<tr>
<td>10   hartmann6</td>
<td>6</td>
<td>1</td>
<td>-3.322E+00</td>
</tr>
<tr>
<td>11   hump</td>
<td>2</td>
<td>2</td>
<td>0.000E+00</td>
</tr>
<tr>
<td>12   hump_camel</td>
<td>2</td>
<td>2</td>
<td>-1.032E+00</td>
</tr>
<tr>
<td>13   levy3</td>
<td>2</td>
<td>18</td>
<td>-1.765E+02</td>
</tr>
<tr>
<td>14   parsopulos</td>
<td>2</td>
<td>12</td>
<td>0.000E+00</td>
</tr>
<tr>
<td>15   rosenbrock10</td>
<td>10</td>
<td>1</td>
<td>0.000E+00</td>
</tr>
<tr>
<td>16   rosenbrock2</td>
<td>2</td>
<td>1</td>
<td>0.000E+00</td>
</tr>
</tbody>
</table>
## Test problems

<table>
<thead>
<tr>
<th>Problems</th>
<th>p</th>
<th>$N_t^*$</th>
<th>$g^*$</th>
</tr>
</thead>
<tbody>
<tr>
<td>rosenbrock5</td>
<td>5</td>
<td>1</td>
<td>0.000E+00</td>
</tr>
<tr>
<td>shekel10</td>
<td>4</td>
<td>1</td>
<td>-1.054E+01</td>
</tr>
<tr>
<td>shekel5</td>
<td>4</td>
<td>1</td>
<td>-1.015E+01</td>
</tr>
<tr>
<td>shekel7</td>
<td>4</td>
<td>1</td>
<td>-1.040E+01</td>
</tr>
<tr>
<td>shubert</td>
<td>2</td>
<td>18</td>
<td>-1.867E+02</td>
</tr>
<tr>
<td>storn1</td>
<td>2</td>
<td>2</td>
<td>-4.075E-01</td>
</tr>
<tr>
<td>storn2</td>
<td>2</td>
<td>2</td>
<td>-1.806E+01</td>
</tr>
<tr>
<td>storn3</td>
<td>2</td>
<td>2</td>
<td>-2.278E+02</td>
</tr>
<tr>
<td>storn4</td>
<td>2</td>
<td>2</td>
<td>-2.429E+03</td>
</tr>
<tr>
<td>storn5</td>
<td>2</td>
<td>2</td>
<td>-2.478E+04</td>
</tr>
<tr>
<td>storn6</td>
<td>2</td>
<td>2</td>
<td>-2.493E+05</td>
</tr>
<tr>
<td>zakharov10</td>
<td>10</td>
<td>1</td>
<td>0.000E+00</td>
</tr>
<tr>
<td>zakharov2</td>
<td>2</td>
<td>1</td>
<td>0.000E+00</td>
</tr>
<tr>
<td>zakharov20</td>
<td>20</td>
<td>1</td>
<td>0.000E+00</td>
</tr>
<tr>
<td>zakharov4</td>
<td>4</td>
<td>1</td>
<td>0.000E+00</td>
</tr>
<tr>
<td>zakharov5</td>
<td>5</td>
<td>1</td>
<td>0.000E+00</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 $K_{max} = 100000$.
- 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 $K_{max}$ iterations.
- The swarm size is given by $\min(6^p, 100)$, where $p$ 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 $K^{max} = 100000$
- 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 $K^{max}$ iterations
- The swarm size is given by $\min(6^p, 100)$, where $p$ 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 $K_{max} = 100000$
- 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 $K_{max}$ iterations
- The swarm size is given by $\min(6^p, 100)$, where $p$ 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 $K^{max} = 100000$
- Coefficients $\mu$ and $\nu$ were both set to 1.2
- The inertial parameter $\nu(t)$ was linearly scaled from 0.7 to 0.2 over a maximum of $K^{max}$ iterations
- The swarm size is given by $\min(6^p, 100)$, where $p$ 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 $K^{max} = 100000$.
- Coefficients $\mu$ and $\nu$ were both set to 1.2.
- The inertial parameter $\nu(t)$ was linearly scaled from 0.7 to 0.2 over a maximum of $K^{max}$ iterations.
- The swarm size is given by $\min(6^p, 100)$, where $p$ is the problem dimension.
## Numerical results

### Gradient version

<table>
<thead>
<tr>
<th>F.O.</th>
<th>$N_{afe}$</th>
<th>$N_{age}$</th>
<th>$g_a^*$</th>
<th>$g_{best}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>100</td>
<td>3444343</td>
<td>873</td>
<td>0,000E+00</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>2782058</td>
<td>545</td>
<td>0,000E+00</td>
</tr>
<tr>
<td>3</td>
<td>100</td>
<td>1740823</td>
<td>1397</td>
<td>3,979E-01</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>1647820</td>
<td>4420</td>
<td>2,618E-23</td>
</tr>
<tr>
<td>5</td>
<td>100</td>
<td>283500</td>
<td>70615</td>
<td>-1,000E+00</td>
</tr>
</tbody>
</table>

### Not differentiable

<table>
<thead>
<tr>
<th>F.O.</th>
<th>$N_{afe}$</th>
<th>$N_{age}$</th>
<th>$g_a^*$</th>
<th>$g_{best}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>Not differentiable</td>
<td>3600000</td>
<td>59</td>
<td>2,431E+01</td>
</tr>
<tr>
<td>7</td>
<td>20</td>
<td>3600000</td>
<td>7754</td>
<td>1,084E-02</td>
</tr>
<tr>
<td>8</td>
<td>20</td>
<td>1000000</td>
<td>483</td>
<td>-3,850E+00</td>
</tr>
<tr>
<td>9</td>
<td>100</td>
<td>10000000</td>
<td>525</td>
<td>-2,937E+00</td>
</tr>
<tr>
<td>10</td>
<td>40</td>
<td>10000000</td>
<td>1082</td>
<td>-1,032E+00</td>
</tr>
</tbody>
</table>

### Approximate descent direction version

<table>
<thead>
<tr>
<th>F.O.</th>
<th>$N_{afe}$</th>
<th>$g_a^*$</th>
<th>$g_{best}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>100</td>
<td>3602386</td>
<td>0,000E+00</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>3600983</td>
<td>0,000E+00</td>
</tr>
<tr>
<td>3</td>
<td>100</td>
<td>3601171</td>
<td>3,979E-01</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>10003223</td>
<td>0,000E+00</td>
</tr>
<tr>
<td>5</td>
<td>100</td>
<td>3601354</td>
<td>-1,000E+00</td>
</tr>
<tr>
<td>6</td>
<td>100</td>
<td>10104250</td>
<td>-1,448E+04</td>
</tr>
<tr>
<td>7</td>
<td>100</td>
<td>3600967</td>
<td>3,000E+00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>F.O.</th>
<th>$N_{afe}$</th>
<th>$g_a^*$</th>
<th>$g_{best}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>0</td>
<td>1004487</td>
<td>2,257E-02</td>
</tr>
<tr>
<td>9</td>
<td>100</td>
<td>10002098</td>
<td>-3,862E+00</td>
</tr>
<tr>
<td>10</td>
<td>100</td>
<td>10002652</td>
<td>-3,202E+00</td>
</tr>
<tr>
<td>11</td>
<td>100</td>
<td>3600946</td>
<td>-1,032E+00</td>
</tr>
<tr>
<td>12</td>
<td>100</td>
<td>3601098</td>
<td>2,362E-06</td>
</tr>
<tr>
<td>13</td>
<td>0</td>
<td>3601052</td>
<td>-1,765E+02</td>
</tr>
<tr>
<td>14</td>
<td>85</td>
<td>3600819</td>
<td>2,607E-07</td>
</tr>
<tr>
<td>15</td>
<td>0</td>
<td>10009292</td>
<td>8,726E+00</td>
</tr>
<tr>
<td>16</td>
<td>0</td>
<td>3601268</td>
<td>1,437E-06</td>
</tr>
</tbody>
</table>
### Numerical results

**Gradient version**

<table>
<thead>
<tr>
<th>F.O.</th>
<th>$N_{afe}$</th>
<th>$N_{age}$</th>
<th>$g_a^*$</th>
<th>$g_{best}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>100000000</td>
<td>177</td>
<td>4.652E+03</td>
<td>2.393E+03</td>
</tr>
<tr>
<td>18</td>
<td>100000000</td>
<td>1850</td>
<td>-9.160E+00</td>
<td>-1.026E+01</td>
</tr>
<tr>
<td>19</td>
<td>100000000</td>
<td>2126</td>
<td>-7.801E+00</td>
<td>-8.760E+00</td>
</tr>
<tr>
<td>20</td>
<td>100000000</td>
<td>1909</td>
<td>-9.401E+00</td>
<td>-9.997E+00</td>
</tr>
<tr>
<td>21</td>
<td>0</td>
<td>3600000</td>
<td>-1.024E+02</td>
<td>-1.648E+02</td>
</tr>
<tr>
<td>22</td>
<td>100</td>
<td>1366222</td>
<td>-4.075E-01</td>
<td>-4.075E-01</td>
</tr>
<tr>
<td>23</td>
<td>100</td>
<td>3600000</td>
<td>-1.806E+01</td>
<td>-1.806E+01</td>
</tr>
<tr>
<td>24</td>
<td>100</td>
<td>3600000</td>
<td>-2.278E+02</td>
<td>-2.278E+02</td>
</tr>
<tr>
<td>25</td>
<td>100</td>
<td>3600000</td>
<td>-2.429E+03</td>
<td>-2.429E+03</td>
</tr>
<tr>
<td>26</td>
<td>100</td>
<td>3600000</td>
<td>-2.477E+04</td>
<td>-2.478E+04</td>
</tr>
<tr>
<td>27</td>
<td>10</td>
<td>3600000</td>
<td>1.607E+05</td>
<td>-2.436E+05</td>
</tr>
<tr>
<td>28</td>
<td>0</td>
<td>100000000</td>
<td>141</td>
<td>3.102E+01</td>
</tr>
<tr>
<td>29</td>
<td>0</td>
<td>100000000</td>
<td>135</td>
<td>7.935E+02</td>
</tr>
<tr>
<td>30</td>
<td>100</td>
<td>1433664</td>
<td>8,325E-112</td>
<td>0.000E+00</td>
</tr>
<tr>
<td>31</td>
<td>100</td>
<td>100000000</td>
<td>313</td>
<td>2.780E-21</td>
</tr>
<tr>
<td>32</td>
<td>40</td>
<td>100000000</td>
<td>160</td>
<td>3.031E-04</td>
</tr>
</tbody>
</table>

**Approximate descent direction version**

<table>
<thead>
<tr>
<th>F.O.</th>
<th>$N_{afe}$</th>
<th>$g_a^*$</th>
<th>$g_{best}$</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>10004006</td>
<td>-1.052E+01</td>
<td>-1.052E+01</td>
</tr>
<tr>
<td>100</td>
<td>10003906</td>
<td>-1.014E+01</td>
<td>-1.014E+01</td>
</tr>
<tr>
<td>100</td>
<td>10004006</td>
<td>-1.039E+01</td>
<td>-1.039E+01</td>
</tr>
<tr>
<td>100</td>
<td>3600000</td>
<td>-1.867E+02</td>
<td>-1.867E+02</td>
</tr>
<tr>
<td>100</td>
<td>3600000</td>
<td>-1.806E+01</td>
<td>-1.806E+01</td>
</tr>
<tr>
<td>100</td>
<td>3600000</td>
<td>-2.278E+02</td>
<td>-2.278E+02</td>
</tr>
<tr>
<td>100</td>
<td>3600000</td>
<td>-2.429E+03</td>
<td>-2.429E+03</td>
</tr>
<tr>
<td>100</td>
<td>3600000</td>
<td>-2.478E+04</td>
<td>-2.478E+04</td>
</tr>
<tr>
<td>100</td>
<td>3600000</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>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>
Outline

1. Notation and motivation for global optimization
2. A motivating example
3. The particle swarm algorithm
4. Modification of PSOA for multi-local optimization
5. Numerical results in semi-infinite programming
6. Conclusions and future work
The test set

The test problems were obtained from SIP where \( \bar{x} \) was replaced by \( x^* \), where \( x^* \) is the SIP solution included in the SIPAMPL database. SIPAMPL stands for SIP with AMPL and is a software package that provides, among other features, a database of SIP coded problems.

All SIP problems considered have only one infinite constraint.

<table>
<thead>
<tr>
<th>SIP problem</th>
<th>Test problem</th>
<th>( p )</th>
<th>Obs</th>
</tr>
</thead>
<tbody>
<tr>
<td>watson2</td>
<td>sip_wat2</td>
<td>1</td>
<td>Unidimensional</td>
</tr>
<tr>
<td>vaz3</td>
<td>sip_vaz3</td>
<td>2</td>
<td>Air pollution abatement</td>
</tr>
<tr>
<td>priceS6</td>
<td>sip_S6</td>
<td>6</td>
<td>Higher dimension in SIPAMPL</td>
</tr>
<tr>
<td>priceU</td>
<td>sip_U</td>
<td>6</td>
<td>Higher dimension in SIPAMPL</td>
</tr>
<tr>
<td>random</td>
<td>sip_rand</td>
<td>6</td>
<td>Random generated with known solution</td>
</tr>
</tbody>
</table>
The test set

- The test problems were obtained from SIP where $\bar{x}$ was replaced by $x^*$, where $x^*$ is the SIP solution included in the SIPAMPL database. SIPAMPL stands for SIP with AMPL and is a software package that provides, among other features, a database of SIP coded problems.

- All SIP problems considered have only one infinite constraint.

<table>
<thead>
<tr>
<th>SIP problem</th>
<th>Test problem</th>
<th>$p$</th>
<th>Obs</th>
</tr>
</thead>
<tbody>
<tr>
<td>watsont2</td>
<td>sip_watsont2</td>
<td>1</td>
<td>Unidimensional</td>
</tr>
<tr>
<td>vaz3</td>
<td>sip_vaz3</td>
<td>2</td>
<td>Air pollution abatement</td>
</tr>
<tr>
<td>priceS6</td>
<td>sip_S6</td>
<td>6</td>
<td>Higher dimension in SIPAMPL</td>
</tr>
<tr>
<td>priceU</td>
<td>sip_U</td>
<td>6</td>
<td>Higher dimension in SIPAMPL</td>
</tr>
<tr>
<td>random</td>
<td>sip_rand</td>
<td>6</td>
<td>Random generated with known solution</td>
</tr>
</tbody>
</table>
Numerical results

- A population of 40 particles and a maximum of 2000 iterations was used, with the steepest ascent direction version.

- sip_wat2 a global and a local maxima were found. 10 particles converged to the local maxima $t = 1$ with $\bar{g}(1) = -0.058594$ and the remaining 30 to the global one ($t = 0$) with $\bar{g}(0) = -2.5156e - 08$

- In sip_vaz3 the objective function is flat (equal to zero) in a region.

<table>
<thead>
<tr>
<th>$t$</th>
<th>$\bar{g}(t)$</th>
<th>npar</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(-0.783012, 2.172526)$</td>
<td>0.000000</td>
<td>1</td>
</tr>
<tr>
<td>$(-0.112199, -0.686259)$</td>
<td>0.000000</td>
<td>1</td>
</tr>
<tr>
<td>$(-0.278460, 0.095245)$</td>
<td>0.000000</td>
<td>1</td>
</tr>
<tr>
<td>$(-0.446057, 1.157275)$</td>
<td>0.000000</td>
<td>1</td>
</tr>
<tr>
<td>$(0.443709, 3.811052)$</td>
<td>0.000000</td>
<td>1</td>
</tr>
<tr>
<td>$(3.684002, -0.629689)$</td>
<td>0.500007</td>
<td>22</td>
</tr>
<tr>
<td>$(1.099826, 0.112477)$</td>
<td>0.500055</td>
<td>13</td>
</tr>
</tbody>
</table>
Numerical results

- A population of 40 particles and a maximum of 2000 iterations was used, with the steepest ascent direction version.

- `sip_wat2` a global and a local maxima were found. 10 particles converged to the local maxima $t = 1$ with $\bar{g}(1) = -0.058594$ and the remaining 30 to the global one $(t = 0)$ with $\bar{g}(0) = -2.5156e - 08$

- In `sip_vaz3` the objective function is flat (equal to zero) in a region.

<table>
<thead>
<tr>
<th>$t$</th>
<th>$\bar{g}(t)$</th>
<th>npar</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(-0.783012, 2.172526)$</td>
<td>0.00000000</td>
<td>1</td>
</tr>
<tr>
<td>$(-0.112199, -0.686259)$</td>
<td>0.00000000</td>
<td>1</td>
</tr>
<tr>
<td>$(-0.278460, 0.095245)$</td>
<td>0.00000000</td>
<td>1</td>
</tr>
<tr>
<td>$(-0.446057, 1.157275)$</td>
<td>0.00000000</td>
<td>1</td>
</tr>
<tr>
<td>$(0.443709, 3.811052)$</td>
<td>0.00000000</td>
<td>1</td>
</tr>
<tr>
<td>$(3.684002, -0.629689)$</td>
<td>0.500007</td>
<td>22</td>
</tr>
<tr>
<td>$(1.099826, 0.112477)$</td>
<td>0.500055</td>
<td>13</td>
</tr>
</tbody>
</table>
Numerical results

- A population of 40 particles and a maximum of 2000 iterations was used, with the steepest ascent direction version.

- `sip_wat2` a global and a local maxima were found. 10 particles converged to the local maxima $t = 1$ with $g(1) = -0.058594$ and the remaining 30 to the global one ($t = 0$) with $g(0) = -2.5156e-08$

- In `sip_vaz3` the objective function is flat (equal to zero) in a region.

<table>
<thead>
<tr>
<th>$t$</th>
<th>$g(t)$</th>
<th>npar</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(-0.783012, 2.172526)$</td>
<td>0.000000</td>
<td>1</td>
</tr>
<tr>
<td>$(-0.112199, -0.686259)$</td>
<td>0.000000</td>
<td>1</td>
</tr>
<tr>
<td>$(-0.278460, 0.095245)$</td>
<td>0.000000</td>
<td>1</td>
</tr>
<tr>
<td>$(-0.446057, 1.157275)$</td>
<td>0.000000</td>
<td>1</td>
</tr>
<tr>
<td>$(0.443709, 3.811052)$</td>
<td>0.000000</td>
<td>1</td>
</tr>
<tr>
<td>$(3.684002, -0.629689)$</td>
<td>0.500007</td>
<td>22</td>
</tr>
<tr>
<td>$(1.099826, 0.112477)$</td>
<td>0.500055</td>
<td>13</td>
</tr>
</tbody>
</table>
Numerical results

- *sip_S6* a reported global maximizer and two local with objective function values of 0.027092, -3.69008 and -1.95425 respectively.

<table>
<thead>
<tr>
<th>( t )</th>
<th>( \bar{g}(t) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>((1.622134, 1.687810, 2.000000, 0.085439, 2.000000, 0.350174))</td>
<td>0.024811</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>((1.634326, 1.671065, 2.000000, 0.054348, 2.000000, 2.000000))</td>
<td>(-1.954538)</td>
</tr>
</tbody>
</table>

- *sip_U* reported two global maximizers and eleven local maximizers

<table>
<thead>
<tr>
<th>( t )</th>
<th>( \bar{g}(t) )</th>
<th>( npar )</th>
</tr>
</thead>
<tbody>
<tr>
<td>((-0.665555, -1.000000, 1.00, 1.00, 1.00, 1.00))</td>
<td>(-0.002587)</td>
<td>1</td>
</tr>
<tr>
<td>((-0.689138, -0.933410, 1.00, 1.00, 1.00, 1.00))</td>
<td>(-0.003319)</td>
<td>1</td>
</tr>
<tr>
<td>((-0.890160, -1.000000, 1.00, 1.00, 1.00, 1.00))</td>
<td>(-0.000225)</td>
<td>1</td>
</tr>
<tr>
<td>((-0.894640, -1.000000, 1.00, 1.00, 1.00, 1.00))</td>
<td>(-0.000103)</td>
<td>1</td>
</tr>
<tr>
<td>((-0.897369, -1.000000, 1.00, 1.00, 1.00, 1.00))</td>
<td>(-0.000648)</td>
<td>1</td>
</tr>
<tr>
<td>((1.000000, 1.000000, 1.00, 1.00, 1.00, 1.00))</td>
<td>(0.239638e-07)</td>
<td>35</td>
</tr>
</tbody>
</table>
Numerical results

sip_S6 a reported global maximizer and two local with objective function values of 0.027092, -3.69008 and -1.95425 respectively.

<table>
<thead>
<tr>
<th>t</th>
<th>(\bar{g}(t))</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>(1.622134, 1.687810, 2.000000, 0.085439, 2.000000, 0.350174)</td>
<td>0.024811</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>(1.634326, 1.671065, 2.000000, 0.054348, 2.000000, 2.000000)</td>
<td>-1.954538</td>
</tr>
</tbody>
</table>

sip_U reported two global maximizers and eleven local maximizers

<table>
<thead>
<tr>
<th>t</th>
<th>(\bar{g}(t))</th>
<th>npar</th>
</tr>
</thead>
<tbody>
<tr>
<td>(-0.665555,-1.000000,1.00,1.00,1.00,1.00)</td>
<td>-0.002587</td>
<td>1</td>
</tr>
<tr>
<td>(-0.689138,-0.933410,1.00,1.00,1.00,1.00)</td>
<td>-0.003319</td>
<td>1</td>
</tr>
<tr>
<td>(-0.890160,-1.000000,1.00,1.00,1.00,1.00)</td>
<td>-0.000225</td>
<td>1</td>
</tr>
<tr>
<td>(-0.894640,-1.000000,1.00,1.00,1.00,1.00)</td>
<td>-0.000103</td>
<td>1</td>
</tr>
<tr>
<td>(-0.897369,-1.000000,1.00,1.00,1.00,1.00)</td>
<td>-0.000648</td>
<td>1</td>
</tr>
<tr>
<td>(1.000000,1.000000,1.00,1.00,1.00,1.00)</td>
<td>0.239638e-07</td>
<td>35</td>
</tr>
</tbody>
</table>
Numerical results in semi-infinite programming

Numerical results

sip_rand are known to be any combination of

\[ x_1 = 0.204475, 0.613425, \]
\[ x_2 = 0.286248, 0.858745, \]
\[ x_3 = 0.358527, \]
\[ x_4 = 0.420428, x_5 = 0.112190, 0.336571, 0.560951, 0.785332 \]
and
\[ x_6 = 1. \]
When \( x_i = 1, i = 1, \ldots, 5 \) we may be in the presence of a local maximizer.

<table>
<thead>
<tr>
<th>( x )</th>
<th>( f(x) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( (1.000000, 0.844527, 1.000000, 0.439280, 1.000000, 1.000000) )</td>
<td>0.099529</td>
</tr>
<tr>
<td>( (0.605034, 0.875442, 0.322422, 0.464760, 1.000000, 0.882592) )</td>
<td>0.103833</td>
</tr>
<tr>
<td>( (1.000000, 0.290493, 0.358070, 0.391673, 1.000000, 1.000000) )</td>
<td>0.111176</td>
</tr>
<tr>
<td>( (1.000000, 0.282674, 1.000000, 0.423782, 1.000000, 1.000000) )</td>
<td>0.100581</td>
</tr>
<tr>
<td>( (1.000000, 0.831978, 0.303846, 0.384511, 1.000000, 0.946638) )</td>
<td>0.100823</td>
</tr>
<tr>
<td>( (1.000000, 0.832307, 0.374898, 0.431689, 1.000000, 1.000000) )</td>
<td>0.109419</td>
</tr>
<tr>
<td>( (1.000000, 0.301931, 1.000000, 0.430023, 1.000000, 1.000000) )</td>
<td>0.099764</td>
</tr>
<tr>
<td>( (0.213111, 0.299029, 0.366111, 1.000000, 1.000000, 0.997780) )</td>
<td>0.035990</td>
</tr>
<tr>
<td>( (1.000000, 0.882755, 0.336751, 0.459651, 1.000000, 1.000000) )</td>
<td>0.107248</td>
</tr>
<tr>
<td>( (1.000000, 0.861285, 0.380012, 0.383345, 1.000000, 0.977121) )</td>
<td>0.108998</td>
</tr>
<tr>
<td>( (1.000000, 0.306139, 0.325222, 0.398052, 1.000000, 1.000000) )</td>
<td>0.108086</td>
</tr>
</tbody>
</table>
Outline

1. Notation and motivation for global optimization
2. A motivating example
3. The particle swarm algorithm
4. Modification of PSOA for multi-local optimization
5. Numerical results in semi-infinite programming
6. Conclusions and future work
Conclusions and future work

- We have presented a new multi-local optimization algorithm that evaluates multiple optimal solutions for multi-modal optimization problems.
- The MLOCPSO algorithm adapts the unimodal particle swarm optimizer using ascent directions information to maintain diversity and to drive the particles to neighbor local maxima.
- Ascent directions are obtained through the gradient vector or an heuristic method to produce an approximate ascent 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.
Conclusions and future work

- We have presented a new multi-local optimization algorithm that evaluates multiple optimal solutions for multi-modal optimization problems.
- The MLOCPSEO algorithm adapts the unimodal particle swarm optimizer using ascent directions information to maintain diversity and to drive the particles to neighbor local maxima.
- Ascent directions are obtained through the gradient vector or an heuristic method to produce an approximate ascent 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.
Conclusions and future work

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

- The MLOCPSO algorithm adapts the unimodal particle swarm optimizer using ascent directions information to maintain diversity and to drive the particles to neighbor local maxima.

- Ascent directions are obtained through the gradient vector or an heuristic method to produce an approximate ascent 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.
Conclusions and future work

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

- The MLOCPSO algorithm adapts the unimodal particle swarm optimizer using ascent directions information to maintain diversity and to drive the particles to neighbor local maxima.

- Ascent directions are obtained through the gradient vector or an heuristic method to produce an approximate ascent 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.
Conclusions and future work

- We have presented a new multi-local optimization algorithm that evaluates multiple optimal solutions for multi-modal optimization problems.
- The MLOCPSO algorithm adapts the unimodal particle swarm optimizer using ascent directions information to maintain diversity and to drive the particles to neighbor local maxima.
- Ascent directions are obtained through the gradient vector or an heuristic method to produce an approximate ascent 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.
Conclusions and future work

- Numerical results with the *E. coli* bacteria and laboratory confirmation of the used approaches.
- Theoretical study of the velocity equation when an ascent direction is used. Inclusion of other ascent directions.
- A reduction method for semi-infinite programming using the multi-local particle swarm (a warm start can be used).
Conclusions and future work

- Numerical results with the *E. coli* bacteria and laboratory confirmation of the used approaches.
- Theoretical study of the velocity equation when an ascent direction is used. Inclusion of other ascent directions.
- A reduction method for semi-infinite programming using the multi-local particle swarm (a warm start can be used).
Conclusions and future work

- Numerical results with the *E. coli* bacteria and laboratory confirmation of the used approaches.
- Theoretical study of the velocity equation when an ascent direction is used. Inclusion of other ascent directions.
- A reduction method for semi-infinite programming using the multi-local particle swarm (a warm start can be used).
THE END

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

With special thanks to Eugénio Ferreira (DEB) and Edite Fernandes (DPS).