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

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

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

Outline

General formulation - Nonlinear semi-infinite programming

Problem

$$\begin{aligned} \min_{x \in R^n} \ & f(x) \\ s.t. \ & g(x,t) \leq 0 \\ & \forall t \in T \end{aligned} \qquad \text{(NLSIP)}$$

- $f(x)$ is the objective function
- $g(x,t)$ is the *infinite* constraint function
- $T \subset R^p$ 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

$$\begin{aligned} \min_{x \in R^n} \ & f(x) \\ s.t. \ & g(x,t) \le 0 \\ & \forall t \in T \end{aligned} \qquad \text{(NLSIP)}$$

- $f(x)$ is the objective function
- $g(x,t)$ is the *infinite* constraint function
- $T \subset R^p$ 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

$$\begin{aligned} \min_{x \in R^n} \ & f(x) \\ s.t. \ & g(x,t) \leq 0 \\ & \forall t \in T \end{aligned} \qquad \text{(NLSIP)}$$

- $f(x)$ is the objective function
- $g(x,t)$ is the *infinite* constraint function
- $T \subset R^p$ 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

$$\begin{aligned} \min_{x \in R^n} \quad & f(x) \\ s.t. \quad & g(x,t) \leq 0 \\ & \forall t \in T \end{aligned} \qquad \text{(NLSIP)}$$

- $f(x)$ is the objective function
- $g(x,t)$ is the *infinite* constraint function
- $T \subset R^p$ 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

$$\begin{aligned} \min_{x \in R^n} \quad & f(x) \\ s.t. \quad & g(x,t) \leq 0 \\ & \forall t \in T \end{aligned} \qquad \text{(NLSIP)}$$

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

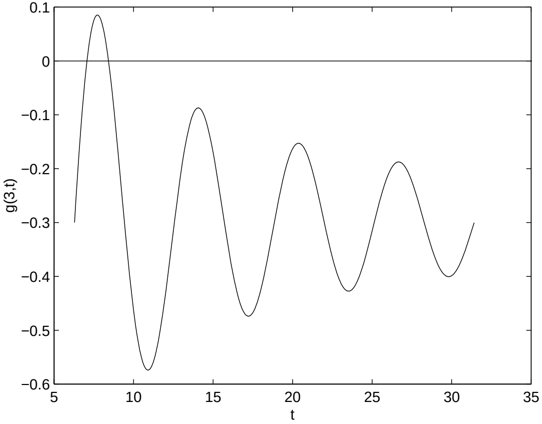
Note

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

An very simple academic example ($n = 1$ and $p = 1$)

Example

$$\min_{x \in R} \quad x^2, \quad s.t. \ \frac{x}{t}\sin(t) - \frac{x}{10} \leq 0, \quad \forall t \in [2\pi, 10\pi]$$



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

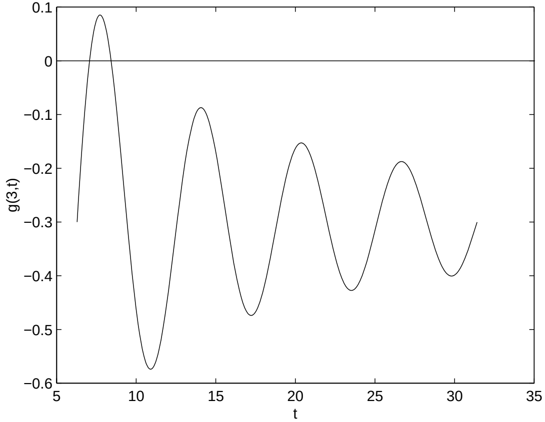
Feasibility

Is $\bar{x} = 3$ feasible?

An very simple academic example ($n = 1$ and $p = 1$)

Example

$$\min_{x \in R} \quad x^2, \quad s.t. \ \frac{x}{t}\sin(t) - \frac{x}{10} \leq 0, \quad \forall t \in [2\pi, 10\pi]$$



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

Feasibility

Is $\bar{x} = 3$ feasible?

Definition of stationary point

Let $x^* \in R^n$ 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 $\lambda_*^0, \lambda_*^1, \lambda_*^2, ..., \lambda_*^{m^*}$ such that

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

with

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

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 complectly 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 complectly 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 complectly 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

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.

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 $\dot{\chi} = h(\chi, u, t),\ \ \chi(t^0) = \chi^0,\ \ t^0 \le t \le t^f$, where χ 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 φ is the performance index of the state variables at final time t^f and ϕ is the integrated performance index during the operation.

- Additional constraints that often reflet 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 χ 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 φ is the performance index of the state variables at final time t^f and ϕ is the integrated performance index during the operation.
- Additional constraints that often reflet 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 χ 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 φ is the performance index of the state variables at final time t^f and ϕ is the integrated performance index during the operation.
- Additional constraints that often reflet some physical limitation of the system can be imposed.

The control problem

The general maximization problem (P) can be posed as

problem (P)

$$\max \quad J(t^f) \tag{1}$$
$$s.t. \quad \dot{\chi} = h(\chi, u, t) \tag{2}$$
$$\underline{\chi} \leq \chi(t) \leq \overline{\chi}, \tag{3}$$
$$\underline{u} \leq u(t) \leq \overline{u}, \tag{4}$$
$$\forall t \in [t^0, t^f] \tag{5}$$

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

problem (P)

$$\max \quad J(t^f) \tag{1}$$
$$s.t. \quad \dot{\chi} = h(\chi, u, t) \tag{2}$$
$$\underline{\chi} \le \chi(t) \le \overline{\chi}, \tag{3}$$
$$\underline{u} \le u(t) \le \overline{u}, \tag{4}$$
$$\forall t \in [t^0, t^f] \tag{5}$$

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

$$\underline{u} \leq w(t^i) \leq \overline{u}, \quad i = 1, \dots, n$$

Where t^i are the spline knots.

The maximization NLP problem is

$$\max_{w(t^i)} \hat{J}(t^f), \quad s.t. \ \underline{u} \leq w(t^i) \leq \overline{u}, \quad i = 1, \dots, n$$

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

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

Where t^i are the spline knots.

The maximization NLP problem is

$$\max_{w(t^i)} \hat{J}(t^f), \quad s.t. \; \underline{u} \leq w(t^i) \leq \overline{u}, \quad i = 1, \ldots, n$$

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

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

Where t^i are the spline knots.

The maximization NLP problem is

$$\max_{w(t^i)} \hat{J}(t^f), \quad s.t. \; \underline{u} \leq w(t^i) \leq \overline{u}, \quad i = 1, \ldots, n$$

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

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

New objective function

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

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

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

New objective function

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

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

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

New objective function

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

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

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

New objective function

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

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.

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 processe. 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 `cprotein` 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 refers to a fed-batch fermentation processe. 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 refers to a fed-batch fermentation processe. 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 refers to a fed-batch fermentation processe. 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 refers to a fed-batch fermentation processe. 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 refers to a fed-batch fermentation processe. 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.

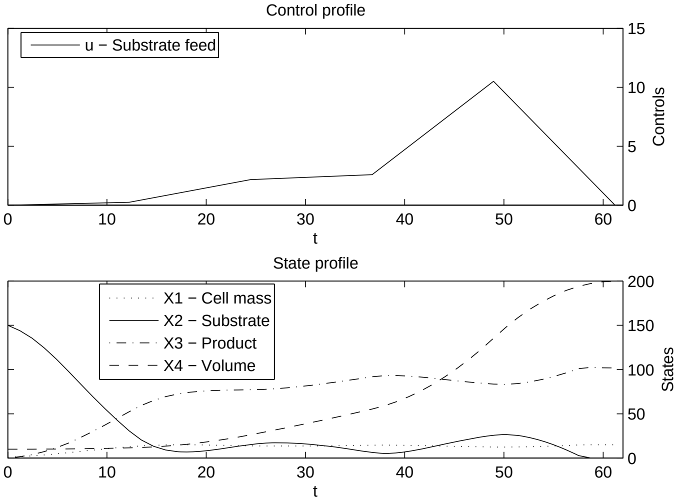
Numerical results

Problema	NT	n	t^f	Cubic $J(t^f)$	Linear $J(t^f)$	Literature $J(t^f)$
penicillin	1	5	132.00	87.70	88.29	87.99
ethanol	1	5	61.20	20550.70	20379.50	20839.00
chemotherapy	1	4	84.00	15.75	16.83	14.48
hprotein	1	5	15.00	38.86	32.73	32.40
rprotein	2	5	10.00	0.13	0.12	0.16

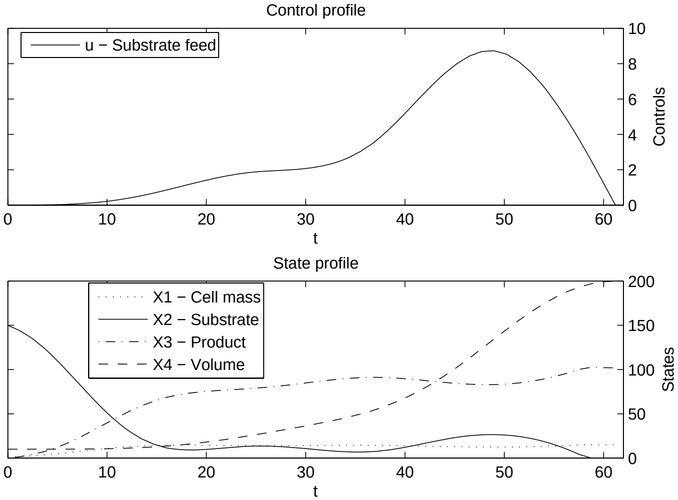
$$J(t^f) = \hat{J}(t^f) = \bar{J}(t^f), \quad \text{for all feasible points - splines}$$

Similar results between approaches. A new solution for the ethanol case.

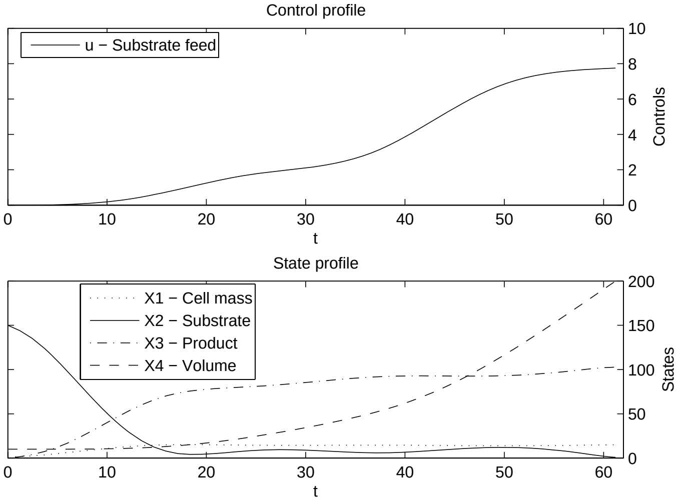
Plots - Linear spline approximation - ethanol case



Plots - Cubic spline approximation - Similar result



Plots - Cubic spline approximation - Best result



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.

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

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 $T \in R^p$.

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 $\wp$, at time instant k, is represented by its current position ($t^{\wp}(k)$), its best ever position ($y^{\wp}(k)$) and its traveling velocity ($v^{\wp}(k)$).

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 $\wp$, at time instant k, is represented by its current position ($t^{\wp}(k)$), its best ever position ($y^{\wp}(k)$) and its traveling velocity ($v^{\wp}(k)$).

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 $\wp$, at time instant k, is represented by its current position ($t^{\wp}(k)$), its best ever position ($y^{\wp}(k)$) and its traveling velocity ($v^{\wp}(k)$).

The new travel position and velocity

The new particle position is updated by

Update position

$$t^{\wp}(k+1) = t^{\wp}(k) + v^{\wp}(k+1),$$

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

Update velocity

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

for $j = 1, \ldots, p$.

- $\iota(k)$ is a weighting factor (inertial)
- μ is the cognition parameter and ν 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

$$t^{\wp}(k+1) = t^{\wp}(k) + v^{\wp}(k+1),$$

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

Update velocity

$$v^{\wp}_j(k+1) = \iota(k)v^{\wp}_j(k) + \mu\omega_{1j}(k)\left(y^{\wp}_j(k) - t^{\wp}_j(k)\right) + \nu\omega_{2j}(k)\left(\hat{y}_j(k) - t^{\wp}_j(k)\right),$$

for $j = 1, \ldots, p$.

- $\iota(k)$ is a weighting factor (inertial)
- μ is the *cognition* parameter and ν 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

$$t^{\wp}(k+1) = t^{\wp}(k) + v^{\wp}(k+1),$$

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

Update velocity

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

for $j = 1, \ldots, p$.

- $\iota(k)$ is a weighting factor (inertial)
- μ is the *cognition* parameter and ν 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

$$t^{\wp}(k+1) = t^{\wp}(k) + v^{\wp}(k+1),$$

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

Update velocity

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

for $j = 1, \ldots, p$.

- $\iota(k)$ is a weighting factor (inertial)
- μ is the *cognition* parameter and ν 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

$$t^{\wp}(k+1) = t^{\wp}(k) + v^{\wp}(k+1),$$

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

Update velocity

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

for $j = 1, \ldots, p$.

- $\iota(k)$ is a weighting factor (inertial)
- μ is the *cognition* parameter and ν is the *social* parameter
- $\omega_{1j}(k)$ and $\omega_{2j}(k)$ 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.*,

Best position

$$\hat{y}(k) \in \arg\min_{a \in \mathcal{A}} \bar{g}(a)$$
$$\mathcal{A} = \left\{ y^1(k), \ldots, y^s(k) \right\} .$$

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.*,

Best position

$$\hat{y}(k) \in \arg\min_{a \in \mathcal{A}} \bar{g}(a)$$
$$\mathcal{A} = \left\{ y^1(k), \ldots, y^s(k) \right\}.$$

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.

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.

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

Multi-local revisited

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

Multi-local optimization problem

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

with $T \in R^n$.

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

Multi-local optimization problem

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

with $T \in R^n$.

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

Multi-local optimization problem

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

with $T \in R^n$.

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

Steepest ascent velocity

$$v_j^{\wp}(k+1) = \iota(k) v_j^{\wp}(k) + \mu\omega_{1j}(k)\left(y_j^{\wp}(k) - t_j^{\wp}(k)\right) + \nu\omega_{2j}(t)\left(\nabla_j \bar{g}(y_j^{\wp}(k))\right),$$

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 ($y^{\wp}(k)$).

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

$$w^{\wp} = \frac{1}{\sum_{j=1}^{m} |\bar{g}(z_j^{\wp}) - \bar{g}(y^{\wp})|} \sum_{j=1}^{m} (\bar{g}(z_j^{\wp}) - \bar{g}(y^{\wp})) \frac{(z_j^{\wp} - y^{\wp})}{\|z_j^{\wp} - y^{\wp}\|}$$

as an ascent direction at $y^{\wp}$, in the velocity equation, to overcome the need to compute the gradient.

Where

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

Under certain conditions $w^{\wp}$ simulates the steepest ascent direction.

Stopping criterion

We propose the stopping criterion

Minimum velocity attained

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

where

Constrained velocity

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

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

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`)

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

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

Test problems

	Problems	p	N_{t^*}	$\bar{g}^*$
17	rosenbrock5	5	1	0.000E+00
18	shekel10	4	1	-1.054E+01
19	shekel5	4	1	-1.015E+01
20	shekel7	4	1	-1.040E+01
21	shubert	2	18	-1.867E+02
22	storn1	2	2	-4.075E-01
23	storn2	2	2	-1.806E+01
24	storn3	2	2	-2.278E+02
25	storn4	2	2	-2.429E+03
26	storn5	2	2	-2.478E+04
27	storn6	2	2	-2.493E+05
28	zakharov10	10	1	0.000E+00
29	zakharov2	2	1	0.000E+00
30	zakharov20	20	1	0.000E+00
31	zakharov4	4	1	0.000E+00
32	zakharov5	5	1	0.000E+00

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 μ and ν 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 μ and ν 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 μ and ν 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 μ and ν 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 μ and ν 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.

Numerical results

	Gradient version					Approximate descent direction version			
	F.O.	N_{afe}	N_{age}	g_a^*	g_{best}	F.O.	N_{afe}	g_a^*	g_{best}
1	100	3444343	873	0,000E+00	0,000E+00	100	3602386	0,000E+00	0,000E+00
2	100	2782058	545	0,000E+00	0,000E+00	100	3600983	0,000E+00	0,000E+00
3	100	1740823	1397	3,979E-01	3,979E-01	100	3601171	3,979E-01	3,979E-01
4	100	1647820	4420	2,618E-23	0,000E+00	100	10003223	0,000E+00	0,000E+00
5	100	283500	70615	-1,000E+00	-1,000E+00	100	3601354	-1,000E+00	-1,000E+00
6	Not differentiable					100	10104250	-1,448E+04	-1,468E+04
7	20	3600000	59	2,431E+01	4,583E+00	100	3600967	3,000E+00	3,000E+00
8	20	10000000	7754	1,084E-02	0,000E+00	0	10004487	2,257E-02	1,503E-02
9	100	10000000	483	-3,850E+00	-3,861E+00	100	10002098	-3,862E+00	-3,863E+00
10	40	10000000	525	-2,937E+00	-3,185E+00	100	10002652	-3,202E+00	-3,242E+00
11	100	963259	1082	-1,032E+00	-1,032E+00	100	3600946	-1,032E+00	-1,032E+00
12	100	1171181	1329	4,651E-08	4,651E-08	100	3601098	2,362E-06	6,756E-07
13	0	3600000	439	-1,276E+02	-1,592E+02	49	3601052	-1,765E+02	-1,765E+02
14	85	2952979	2295	4,922E-23	3,749E-33	75	3600819	2,607E-07	9,685E-08
15	0	10000000	154	8,051E+04	3,387E+04	0	10009292	8,726E+00	7,386E+00
16	0	3600000	91	3,046E+00	1,190E+00	100	3601268	1,437E-06	5,698E-07

Numerical results

	Gradient version					Approximate descent direction version			
	F.O.	N_{afe}	N_{age}	g_a^*	g_{best}	F.O.	N_{afe}	g_a^*	g_{best}
17	0	10000000	177	4,652E+03	2,393E+03	40	10005589	2,203E-01	1,327E-01
18	100	10000000	1850	-9,160E+00	-1,026E+01	100	10004066	-1,052E+01	-1,052E+01
19	100	10000000	2126	-7,801E+00	-8,760E+00	100	10003906	-1,012E+01	-1,014E+01
20	100	10000000	1909	-9,401E+00	-9,997E+00	100	10004069	-1,037E+01	-1,039E+01
21	0	3600000	335	-1,024E+02	-1,648E+02	60	3600999	-1,867E+02	-1,867E+02
22	100	1366222	973	-4,075E-01	-4,075E-01	100	3600804	-4,075E-01	-4,075E-01
23	100	3600000	570	-1,806E+01	-1,806E+01	100	3600902	-1,806E+01	-1,806E+01
24	100	3600000	194	-2,278E+02	-2,278E+02	100	3601003	-2,278E+02	-2,278E+02
25	100	3600000	167	-2,429E+03	-2,429E+03	100	3601160	-2,429E+03	-2,429E+03
26	90	3600000	81	-2,477E+04	-2,478E+04	100	3601278	-2,478E+04	-2,478E+04
27	10	3600000	58	1,607E+05	-2,436E+05	100	3601418	-2,493E+05	-2,493E+05
28	0	10000000	141	4,470E+02	3,102E+01	60	10009759	3,977E-02	2,506E-02
29	0	10000000	135	1,289E+05	7,935E+02	0	10016905	3,633E-01	2,404E-01
30	100	1433664	16314	8,325E-112	0,000E+00	100	3601264	4,987E-07	4,464E-08
31	100	10000000	313	1,997E-13	2,780E-21	100	10005221	2,231E-04	6,612E-05
32	40	10000000	160	8,338E+00	3,031E-04	100	10006065	2,005E-03	1,186E-03

Outline

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.

SIP problem	Test problem	p	Obs
watson2	sip_wat2	1	Unidimensional
vaz3	sip_vaz3	2	Air pollution abatement
priceS6	sip_S6	6	Higher dimension in SIPAMPL
priceU	sip_U	6	Higher dimension in SIPAMPL
random	sip_rand	6	Random generated with known solution

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.

SIP problem	Test problem	p	Obs
`watson2`	`sip_wat2`	1	Unidimensional
`vaz3`	`sip_vaz3`	2	Air pollution abatement
`priceS6`	`sip_S6`	6	Higher dimension in SIPAMPL
`priceU`	`sip_U`	6	Higher dimension in SIPAMPL
`random`	`sip_rand`	6	Random generated with known solution

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.

t	$\bar{g}(t)$	$npar$
$(-0.783012, 2.172526)$	0.000000	1
$(-0.112199, -0.686259)$	0.000000	1
$(-0.278460, 0.095245)$	0.000000	1
$(-0.446057, 1.157275)$	0.000000	1
$(0.443709, 3.811052)$	0.000000	1
$(3.684002, -0.629689)$	0.500007	22
$(1.099826, 0.112477)$	0.500055	13

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.

t	$\bar{g}(t)$	$npar$
$(-0.783012, 2.172526)$	0.000000	1
$(-0.112199, -0.686259)$	0.000000	1
$(-0.278460, 0.095245)$	0.000000	1
$(-0.446057, 1.157275)$	0.000000	1
$(0.443709, 3.811052)$	0.000000	1
$(3.684002, -0.629689)$	0.500007	22
$(1.099826, 0.112477)$	0.500055	13

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.

t	$\bar{g}(t)$	$npar$
$(-0.783012, 2.172526)$	0.000000	1
$(-0.112199, -0.686259)$	0.000000	1
$(-0.278460, 0.095245)$	0.000000	1
$(-0.446057, 1.157275)$	0.000000	1
$(0.443709, 3.811052)$	0.000000	1
$(3.684002, -0.629689)$	0.500007	22
$(1.099826, 0.112477)$	0.500055	13

Numerical results

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

t	$\bar{g}(t)$
...	
$(1.622134, 1.687810, 2.000000, 0.085439, 2.000000, 0.350174)$	0.024811
...	
$(1.634326, 1.671065, 2.000000, 0.054348, 2.000000, 2.000000)$	-1.954538
...	

- sip_U reported two global maximizers and eleven local maximizers

t	$\bar{g}(t)$	$npar$
(-0.665555,-1.000000,1.00,1.00,1.00,1.00)	-0.002587	1
(-0.689138,-0.933410,1.00,1.00,1.00,1.00)	-0.003319	1
(-0.890160,-1.000000,1.00,1.00,1.00,1.00)	-0.000225	1
(-0.894640,-1.000000,1.00,1.00,1.00,1.00)	-0.000103	1
(-0.897369,-1.000000,1.00,1.00,1.000,1.00)	-0.000648	1
(1.000000,1.000000,1.00,1.00,1.00,1.00)	0.239638e-07	35

Numerical results

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

t	$\bar{g}(t)$
...	
$(1.622134, 1.687810, 2.000000, 0.085439, 2.000000, 0.350174)$	0.024811
...	
$(1.634326, 1.671065, 2.000000, 0.054348, 2.000000, 2.000000)$	-1.954538
...	

- sip_U reported two global maximizers and eleven local maximizers

t	$\bar{g}(t)$	$npar$
(-0.665555,-1.000000,1.00,1.00,1.00,1.00)	-0.002587	1
(-0.689138,-0.933410,1.00,1.00,1.00,1.00)	-0.003319	1
(-0.890160,-1.000000,1.00,1.00,1.00,1.00)	-0.000225	1
(-0.894640,-1.000000,1.00,1.00,1.00,1.00)	-0.000103	1
(-0.897369,-1.000000,1.00,1.00,1.000,1.00)	-0.000648	1
(1.000000,1.000000,1.00,1.00,1.00,1.00)	0.239638e-07	35

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.

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

Outline

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

- 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).