Direct multisearch for multiobjective optimization

A. Ismael F. Vaz

University of Minho - Portugal
aivaz@dps.uminho.pt

Joint work with A. L. Custódio, J. F. A. Madeira, and L. N. Vicente

Southampton

October 21, 2010

Outline

Outline

Outline

Outline

Outline

Outline

Outline

Why Derivative-Free Optimization?

Some of the reasons to apply derivative-free optimization are the following:

- Nowadays computer hardware and mathematical algorithms allows increasingly large simulations.
- Functions are noisy (one cannot trust derivatives or approximate them by finite differences).
- Binary codes (source code not available) and random simulations — making automatic differentiation impossible to apply.
- Legacy codes (written in the past and not maintained by the original authors).
- Lack of sophistication of the user (users need improvement but want to use something simple).

Why Derivative-Free Optimization?

Some of the reasons to apply derivative-free optimization are the following:

- Nowadays computer hardware and mathematical algorithms allows increasingly large simulations.
- Functions are noisy (one cannot trust derivatives or approximate them by finite differences).
- Binary codes (source code not available) and random simulations — making automatic differentiation impossible to apply.
- Legacy codes (written in the past and not maintained by the original authors).
- Lack of sophistication of the user (users need improvement but want to use something simple).

Why Derivative-Free Optimization?

Some of the reasons to apply derivative-free optimization are the following:

- Nowadays computer hardware and mathematical algorithms allows increasingly large simulations.
- Functions are noisy (one cannot trust derivatives or approximate them by finite differences).
- Binary codes (source code not available) and random simulations — making automatic differentiation impossible to apply.
- Legacy codes (written in the past and not maintained by the original authors).
- Lack of sophistication of the user (users need improvement but want to use something simple).

Why Derivative-Free Optimization?

Some of the reasons to apply derivative-free optimization are the following:

- Nowadays computer hardware and mathematical algorithms allows increasingly large simulations.
- Functions are noisy (one cannot trust derivatives or approximate them by finite differences).
- Binary codes (source code not available) and random simulations — making automatic differentiation impossible to apply.
- Legacy codes (written in the past and not maintained by the original authors).
- Lack of sophistication of the user (users need improvement but want to use something simple).

Why Derivative-Free Optimization?

Some of the reasons to apply derivative-free optimization are the following:

- Nowadays computer hardware and mathematical algorithms allows increasingly large simulations.
- Functions are noisy (one cannot trust derivatives or approximate them by finite differences).
- Binary codes (source code not available) and random simulations — making automatic differentiation impossible to apply.
- Legacy codes (written in the past and not maintained by the original authors).
- Lack of sophistication of the user (users need improvement but want to use something simple).

Outline

Direct-search methods

Definition

- Sample the objective function at a finite number of points at each iteration.
- Base actions on those function values.
- Do not depend on derivative approximation or model building.

- Direct search of directional type: Achieve descent by using positive spanning sets and moving in the directions of the best points.

Direct-search methods

Definition

- Sample the objective function at a finite number of points at each iteration.
- Base actions on those function values.
- Do not depend on derivative approximation or model building.

- Direct search of directional type: Achieve descent by using positive spanning sets and moving in the directions of the best points.

Direct-search methods

Definition

- Sample the objective function at a finite number of points at each iteration.
- Base actions on those function values.
- Do not depend on derivative approximation or model building.

- Direct search of directional type: Achieve descent by using positive spanning sets and moving in the directions of the best points.

Direct-search methods

Definition

- Sample the objective function at a finite number of points at each iteration.
- Base actions on those function values.
- Do not depend on derivative approximation or model building.

- Direct search of directional type: Achieve descent by using positive spanning sets and moving in the directions of the best points.

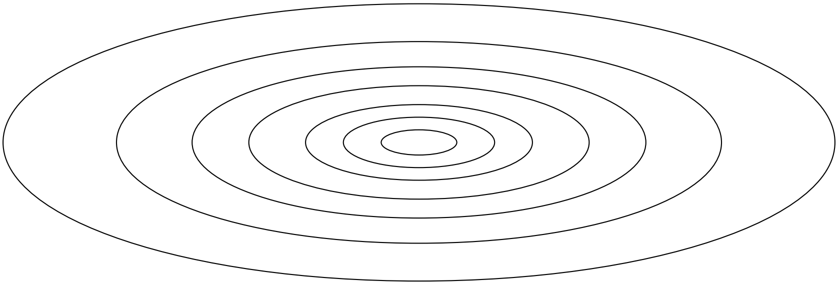
Direct-search methods

Definition

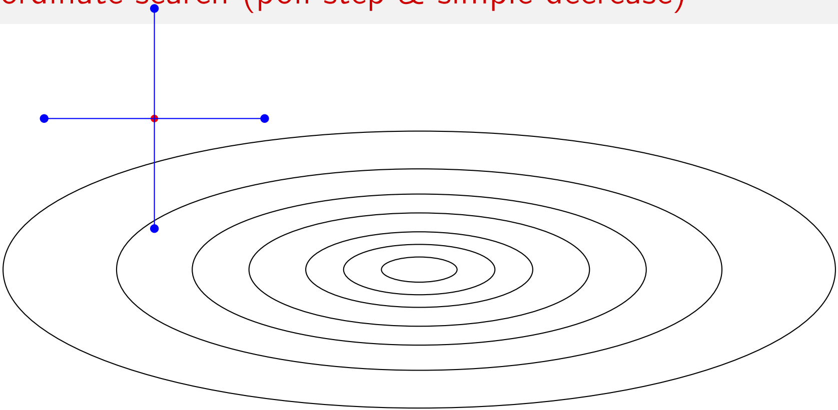
- Sample the objective function at a finite number of points at each iteration.
- Base actions on those function values.
- Do not depend on derivative approximation or model building.

- Direct search of directional type: Achieve descent by using positive spanning sets and moving in the directions of the best points.

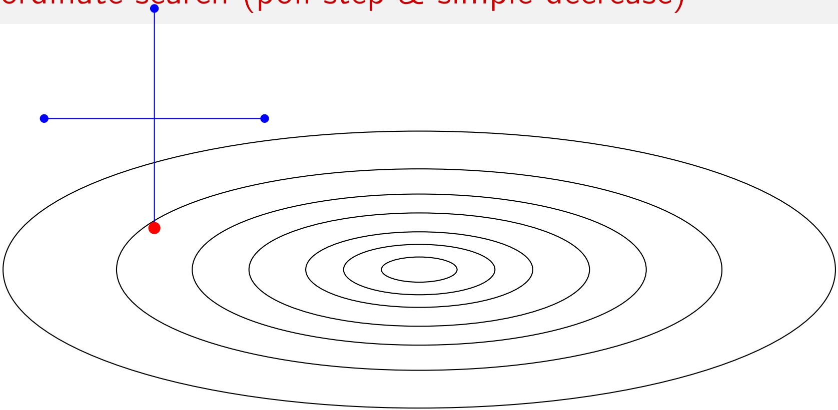
Coordinate search (poll step & simple decrease)



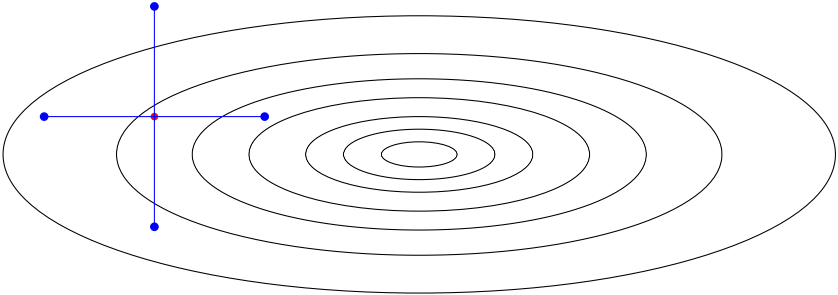
Coordinate search (poll step & simple decrease)



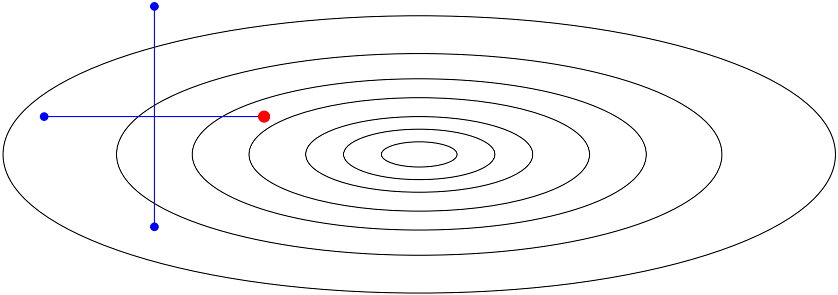
Coordinate search (poll step & simple decrease)



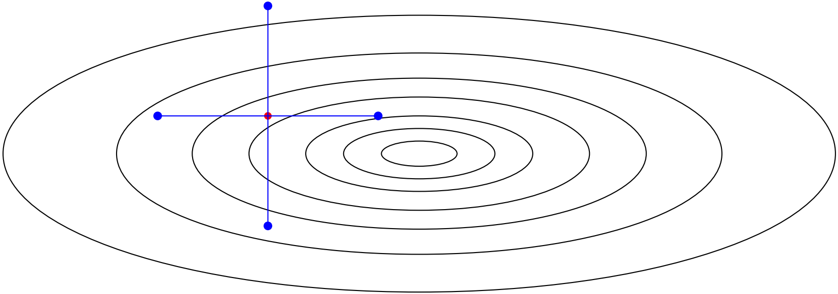
Coordinate search (poll step & simple decrease)



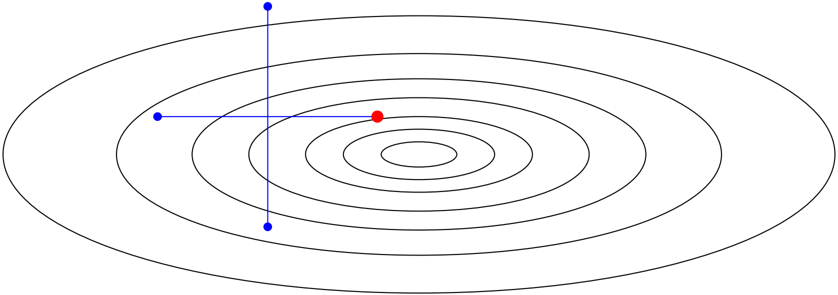
Coordinate search (poll step & simple decrease)



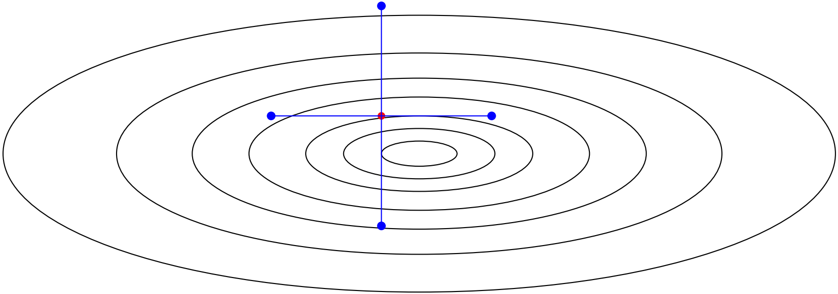
Coordinate search (poll step & simple decrease)



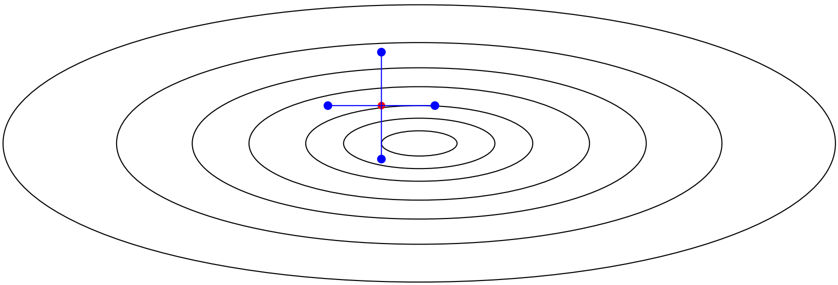
Coordinate search (poll step & simple decrease)



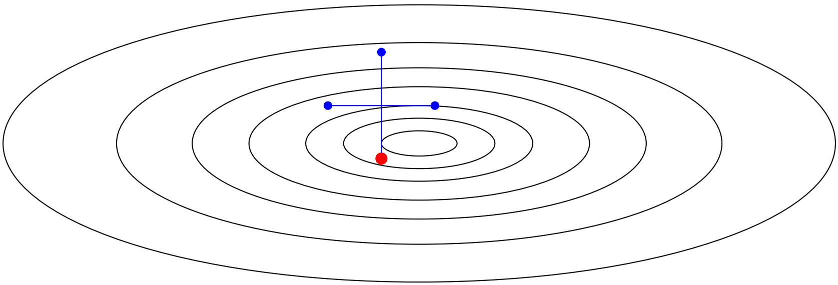
Coordinate search (poll step & simple decrease)



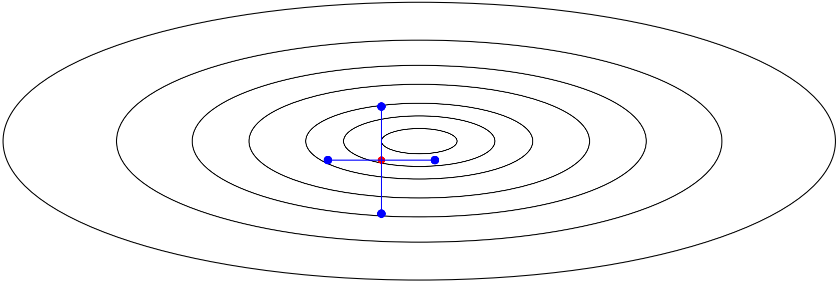
Coordinate search (poll step & simple decrease)



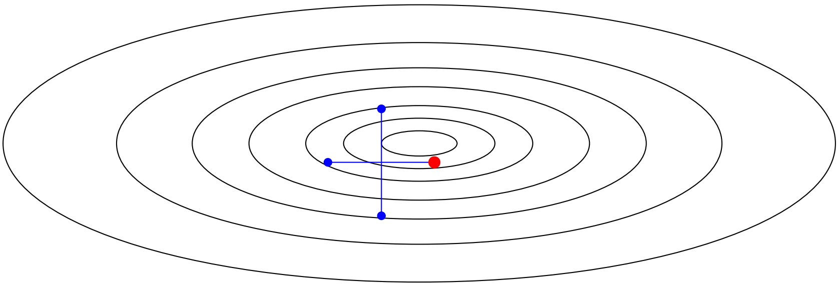
Coordinate search (poll step & simple decrease)



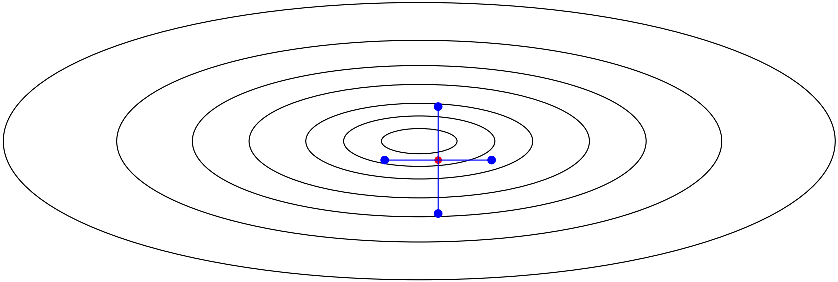
Coordinate search (poll step & simple decrease)



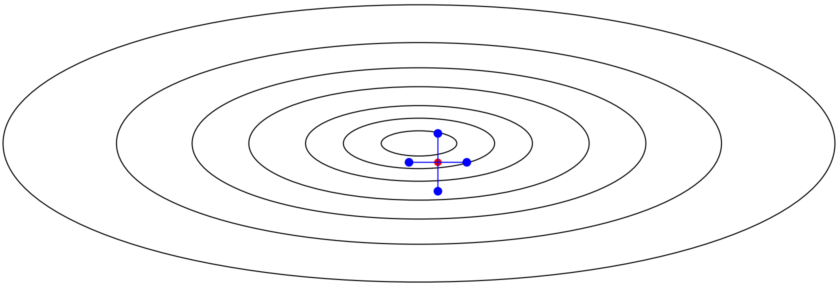
Coordinate search (poll step & simple decrease)



Coordinate search (poll step & simple decrease)



Coordinate search (poll step & simple decrease)



Outline

Derivative-free optimization

Problem formulation (single objective)

$$\min_{x \in \Omega} \; f(x)$$

where

$$\Omega = \{x \in \mathbb{R}^n : \quad \ell \;\leq\; x \;\leq u\}$$

$f : \mathbb{R}^n \rightarrow \mathbb{R} \cup \{+\infty\}$, $\ell \in (\mathbb{R} \cup \{-\infty\})^n$ and $u \in (\mathbb{R} \cup \{+\infty\})^n$

We aim at solving this problem without using derivatives of f.

Some definitions

Positive spanning set

Is a set of vectors that spans $\mathbb{R}^n$ with nonnegative coefficients.

Examples

$$D_{\oplus} = \{e_1, \ldots, e_n, -e_1, \ldots, -e_n\}$$

$$D_{\otimes} = \{e_1, \ldots, e_n, -e_1, \ldots, -e_n, e, -e\}$$

Extreme barrier function

$$f_\Omega(x) = \begin{cases} f(x) & \text{if } x \in \Omega, \\ +\infty & \text{otherwise.} \end{cases}$$

Some definitions

Positive spanning set

Is a set of vectors that spans $\mathbb{R}^n$ with nonnegative coefficients.

Examples

$$D_{\oplus} = \{e_1, \ldots, e_n, -e_1, \ldots, -e_n\}$$

$$D_{\otimes} = \{e_1, \ldots, e_n, -e_1, \ldots, -e_n, e, -e\}$$

Extreme barrier function

$$f_\Omega(x) = \begin{cases} f(x) & \text{if } x \in \Omega, \\ +\infty & \text{otherwise.} \end{cases}$$

Some definitions

Positive spanning set

Is a set of vectors that spans $\mathbb{R}^n$ with nonnegative coefficients.

Examples

$$D_{\oplus} = \{e_1, \ldots, e_n, -e_1, \ldots, -e_n\}$$

$$D_{\otimes} = \{e_1, \ldots, e_n, -e_1, \ldots, -e_n, e, -e\}$$

Extreme barrier function

$$f_{\Omega}(x) = \begin{cases} f(x) & \text{if } x \in \Omega, \\ +\infty & \text{otherwise.} \end{cases}$$

A direct-search method

(0) Initialization
Choose $x_0 \in \Omega$, $\alpha_0 > 0$.

Let $\mathcal{D}$ be a (possibly infinite) set of positive spanning sets.

For $k = 0, 1, 2, \ldots$

(1) Search step (Optional)
Try to compute a point x, using a finite number of trial points, in the grid

$$M_k = \left\{ x_k + \alpha_k D_k z, \; z \in \mathbb{N}_0^{|D_k|} \right\}$$

with $D_k \subseteq \mathcal{D}$ and $f_\Omega(x) < f(x_k)$.

If $f_\Omega(x) < f(x_k)$ then set $x_{k+1} = x$, declare the iteration and the search step successful, and skip the poll step.

A direct-search method

(0) Initialization
Choose $x_0 \in \Omega$, $\alpha_0 > 0$.

Let $\mathcal{D}$ be a (possibly infinite) set of positive spanning sets.

For $k = 0, 1, 2, \ldots$

(1) Search step (Optional)
Try to compute a point x, using a finite number of trial points, in the grid

$$M_k = \left\{ x_k + \alpha_k D_k z, \ z \in \mathbb{N}_0^{|D_k|} \right\}$$

with $D_k \subseteq \mathcal{D}$ and $f_\Omega(x) < f(x_k)$.

If $f_\Omega(x) < f(x_k)$ then set $x_{k+1} = x$, declare the iteration and the search step successful, and skip the poll step.

A direct-search method

(0) Initialization
Choose $x_0 \in \Omega$, $\alpha_0 > 0$.

Let $\mathcal{D}$ be a (possibly infinite) set of positive spanning sets.

For $k = 0, 1, 2, \ldots$

(1) Search step (Optional)
Try to compute a point x, using a finite number of trial points, in the grid

$$M_k = \left\{ x_k + \alpha_k D_k z, \; z \in \mathbb{N}_0^{|D_k|} \right\}$$

with $D_k \subseteq \mathcal{D}$ and $f_\Omega(x) < f(x_k)$.

If $f_\Omega(x) < f(x_k)$ then set $x_{k+1} = x$, declare the iteration and the search step successful, and skip the poll step.

A direct-search method

(0) Initialization
Choose $x_0 \in \Omega$, $\alpha_0 > 0$.

Let $\mathcal{D}$ be a (possibly infinite) set of positive spanning sets.

For $k = 0, 1, 2, \ldots$

(1) Search step (Optional)
Try to compute a point x, using a finite number of trial points, in the grid

$$M_k = \left\{ x_k + \alpha_k D_k z, \; z \in \mathbb{N}_0^{|D_k|} \right\}$$

with $D_k \subseteq \mathcal{D}$ and $f_\Omega(x) < f(x_k)$.

If $f_\Omega(x) < f(x_k)$ then set $x_{k+1} = x$, declare the iteration and the search step successful, and skip the poll step.

A direct-search method

(0) Initialization
Choose $x_0 \in \Omega$, $\alpha_0 > 0$.

Let $\mathcal{D}$ be a (possibly infinite) set of positive spanning sets.

For $k = 0, 1, 2, \ldots$

(1) Search step (Optional)
Try to compute a point x, using a finite number of trial points, in the grid

$$M_k = \left\{ x_k + \alpha_k D_k z, \; z \in \mathbb{N}_0^{|D_k|} \right\}$$

with $D_k \subseteq \mathcal{D}$ and $f_\Omega(x) < f(x_k)$.

If $f_\Omega(x) < f(x_k)$ then set $x_{k+1} = x$, declare the iteration and the search step successful, and skip the poll step.

A direct-search method

(2) Poll step

Optionally order the poll set $P_k = \{x_k + \alpha_k d, \ d \in D_k\}$ with $D_k \subseteq \mathcal{D}$.

If a poll point $x_k + \alpha_k d_k$ is found such that $f_\Omega(x_k + \alpha_k d_k) < f(x_k)$ then stop polling, set $x_{k+1} = x_k + \alpha_k d_k$, and declare the iteration and the poll step successful.

Otherwise declare the iteration (and the poll step) unsuccessful and set $x_{k+1} = x_k$.

A direct-search method

(2) Poll step

Optionally order the poll set $P_k = \{x_k + \alpha_k d,\ d \in D_k\}$ with $D_k \subseteq \mathcal{D}$.

If a poll point $x_k + \alpha_k d_k$ is found such that $f_\Omega(x_k + \alpha_k d_k) < f(x_k)$ then stop polling, set $x_{k+1} = x_k + \alpha_k d_k$, and declare the iteration and the poll step successful.

Otherwise declare the iteration (and the poll step) unsuccessful and set $x_{k+1} = x_k$.

A direct-search method

(2) Poll step

Optionally order the poll set $P_k = \{x_k + \alpha_k d,\ d \in D_k\}$ with $D_k \subseteq \mathcal{D}$.

If a poll point $x_k + \alpha_k d_k$ is found such that $f_\Omega(x_k + \alpha_k d_k) < f(x_k)$ then stop polling, set $x_{k+1} = x_k + \alpha_k d_k$, and declare the iteration and the poll step successful.

Otherwise declare the iteration (and the poll step) unsuccessful and set $x_{k+1} = x_k$.

A direct-search method

(3) Step size update: If the iteration was successful then maintain the step size parameter ($\alpha_{k+1} = \alpha_k$) or double it ($\alpha_{k+1} = 2\alpha_k$) after two consecutive poll successes along the same direction.

If the iteration was unsuccessful, halve the step size parameter ($\alpha_{k+1} = \alpha_k/2$).

A direct-search method

(3) Step size update: If the iteration was successful then maintain the step size parameter $(\alpha_{k+1} = \alpha_k)$ or double it $(\alpha_{k+1} = 2\alpha_k)$ after two consecutive poll successes along the same direction.

If the iteration was unsuccessful, halve the step size parameter $(\alpha_{k+1} = \alpha_k/2)$.

Some comments

We could present the previous algorithm in a different form, namely by

- fixing the set D_k ($D_k = D$, $\forall k$) not to change with the iteration number (problem with only bound constraints).
- allowing the set D_k to be computed in a way to conform with possible linear constraints.
- to use a forcing function $\rho(\cdot)$ (*e.g.*, $\rho(t) = t^2$) instead of a integer lattice (the mesh M_k). A forcing function $\rho(\cdot)$ is continuous, positive, and satisfies $\lim_{t \longrightarrow 0^+} \rho(t)/t = 0$ and $\rho(t_1) \leq \rho(t_2)$ if $t_1 < t_2$.

A point x is accepted (successful) in the search step if

$$f_\Omega(x) < f(x_k) - \rho(\alpha_k).$$

Some comments

We could present the previous algorithm in a different form, namely by

- fixing the set D_k ($D_k = D$, $\forall k$) not to change with the iteration number (problem with only bound constraints).
- allowing the set D_k to be computed in a way to conform with possible linear constraints.
- to use a forcing function $\rho(\cdot)$ (*e.g.*, $\rho(t) = t^2$) instead of a integer lattice (the mesh M_k). A forcing function $\rho(\cdot)$ is continuous, positive, and satisfies $\lim_{t \longrightarrow 0^+} \rho(t)/t = 0$ and $\rho(t_1) \leq \rho(t_2)$ if $t_1 < t_2$.

  A point x is accepted (successful) in the search step if

$$f_\Omega(x) < f(x_k) - \rho(\alpha_k).$$

Some comments

We could present the previous algorithm in a different form, namely by

- fixing the set D_k ($D_k = D,\ \forall k$) not to change with the iteration number (problem with only bound constraints).
- allowing the set D_k to be computed in a way to conform with possible linear constraints.
- to use a forcing function $\rho(\cdot)$ (*e.g.*, $\rho(t) = t^2$) instead of a integer lattice (the mesh M_k). A forcing function $\rho(\cdot)$ is continuous, positive, and satisfies $\lim_{t \longrightarrow 0^+} \rho(t)/t = 0$ and $\rho(t_1) \leq \rho(t_2)$ if $t_1 < t_2$.

  A point x is accepted (successful) in the search step if

$$f_\Omega(x) < f(x_k) - \rho(\alpha_k).$$

Some comments

We could present the previous algorithm in a different form, namely by

- fixing the set D_k ($D_k = D$, $\forall k$) not to change with the iteration number (problem with only bound constraints).
- allowing the set D_k to be computed in a way to conform with possible linear constraints.
- to use a forcing function $\rho(\cdot)$ (*e.g.*, $\rho(t) = t^2$) instead of a integer lattice (the mesh M_k). A forcing function $\rho(\cdot)$ is continuous, positive, and satisfies $\lim_{t \longrightarrow 0^+} \rho(t)/t = 0$ and $\rho(t_1) \leq \rho(t_2)$ if $t_1 < t_2$.

  A point x is accepted (successful) in the search step if

$$f_\Omega(x) < f(x_k) - \rho(\alpha_k).$$

The presented algorithm just suits for the multiobjective version to be described.

Outline

Derivative-free multiobjective optimization

MOO problem

$$\min_{x\in\Omega} F(x) \equiv (f_1(x), f_2(x), \ldots, f_m(x))^\top$$

where

$$\Omega = \{x \in \mathbb{R}^n : \quad \ell \ \leq \ x \ \leq u\}$$

$f_j : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}, \ j = 1, \ldots, m,$
$\ell \in (\mathbb{R} \cup \{-\infty\})^n$ and $u \in (\mathbb{R} \cup \{+\infty\})^n$

- Several objectives, often conflicting.
- Functions with unknown derivatives.
- Expensive function evaluations, possibly subject to noise.
- Impractical to compute approximations to derivatives.

Derivative-free multiobjective optimization

MOO problem

$$\min_{x\in\Omega} F(x) \equiv (f_1(x), f_2(x), \ldots, f_m(x))^\top$$

where

$$\Omega = \{x \in \mathbb{R}^n : \quad \ell \ \leq \ x \ \leq u\}$$

$f_j : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$, $j = 1, \ldots, m$,
$\ell \in (\mathbb{R} \cup \{-\infty\})^n$ and $u \in (\mathbb{R} \cup \{+\infty\})^n$

- Several objectives, often conflicting.
- Functions with unknown derivatives.
- Expensive function evaluations, possibly subject to noise.
- Impractical to compute approximations to derivatives.

Derivative-free multiobjective optimization

MOO problem

$$\min_{x \in \Omega} F(x) \equiv (f_1(x), f_2(x), \ldots, f_m(x))^\top$$

where

$$\Omega = \{x \in \mathbb{R}^n : \quad \ell \leq x \leq u\}$$

$f_j : \mathbb{R}^n \rightarrow \mathbb{R} \cup \{+\infty\}, \; j = 1, \ldots, m,$
$\ell \in (\mathbb{R} \cup \{-\infty\})^n$ and $u \in (\mathbb{R} \cup \{+\infty\})^n$

- Several objectives, often conflicting.
- Functions with unknown derivatives.
- Expensive function evaluations, possibly subject to noise.
- Impractical to compute approximations to derivatives.

Derivative-free multiobjective optimization

MOO problem

$$\min_{x \in \Omega} F(x) \equiv (f_1(x), f_2(x), \ldots, f_m(x))^\top$$

where

$$\Omega = \{x \in \mathbb{R}^n : \quad \ell \leq x \leq u\}$$

$f_j : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\},\ j = 1, \ldots, m,$
$\ell \in (\mathbb{R} \cup \{-\infty\})^n$ and $u \in (\mathbb{R} \cup \{+\infty\})^n$

- Several objectives, often conflicting.
- Functions with unknown derivatives.
- Expensive function evaluations, possibly subject to noise.
- Impractical to compute approximations to derivatives.

DMS algorithmic main lines

- Does not aggregate any of the objective functions.
- Generalizes ALL direct-search methods of directional type to MOO.
- Makes use of search/poll paradigm.
- Implements an optional search step (only to disseminate the search).
- Tries to capture the whole Pareto front from the polling procedure.

DMS algorithmic main lines

- Does not aggregate any of the objective functions.
- Generalizes ALL direct-search methods of directional type to MOO.
- Makes use of search/poll paradigm.
- Implements an optional search step (only to disseminate the search).
- Tries to capture the whole Pareto front from the polling procedure.

DMS algorithmic main lines

- Does not aggregate any of the objective functions.
- Generalizes ALL direct-search methods of directional type to MOO.
- Makes use of search/poll paradigm.
- Implements an optional search step (only to disseminate the search).
- Tries to capture the whole Pareto front from the polling procedure.

DMS algorithmic main lines

- Does not aggregate any of the objective functions.
- Generalizes ALL direct-search methods of directional type to MOO.
- Makes use of search/poll paradigm.
- Implements an optional search step (only to disseminate the search).
- Tries to capture the whole Pareto front from the polling procedure.

DMS algorithmic main lines

- Does not aggregate any of the objective functions.
- Generalizes ALL direct-search methods of directional type to MOO.
- Makes use of search/poll paradigm.
- Implements an optional search step (only to disseminate the search).
- Tries to capture the whole Pareto front from the polling procedure.

DMS algorithmic main lines

- Keeps a list of feasible nondominated points.
- Poll centers are chosen from the list.
- Successful iterations correspond to list changes.

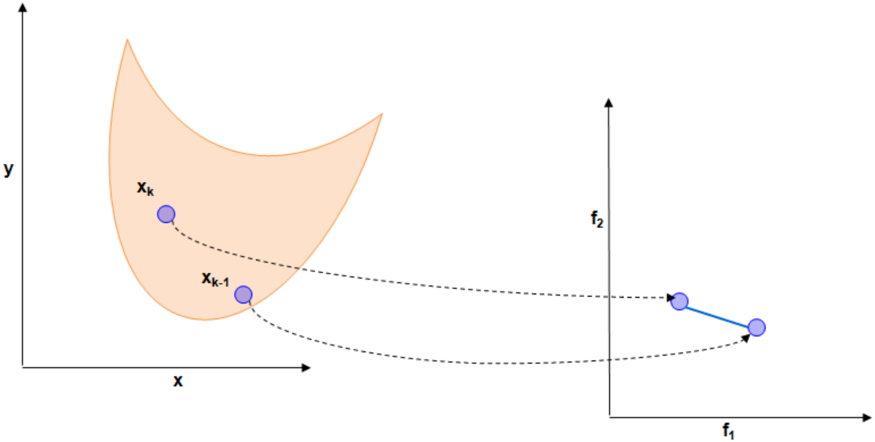
DMS algorithmic main lines

- Keeps a list of feasible nondominated points.
- Poll centers are chosen from the list.
- Successful iterations correspond to list changes.

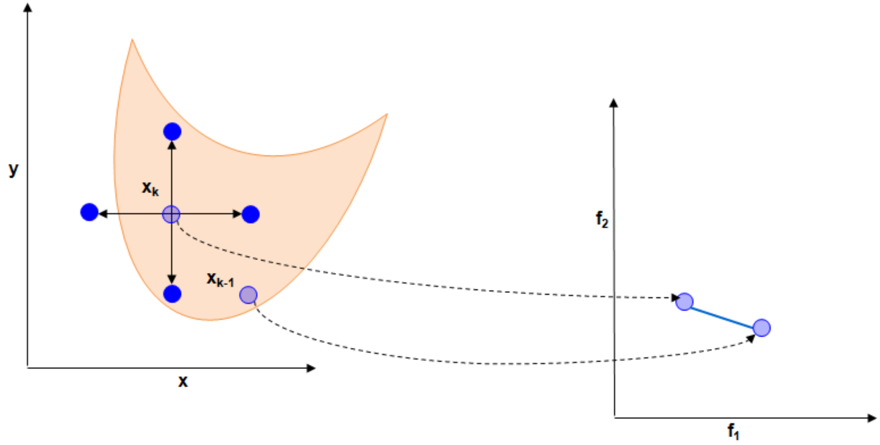
DMS algorithmic main lines

- Keeps a list of feasible nondominated points.
- Poll centers are chosen from the list.
- Successful iterations correspond to list changes.

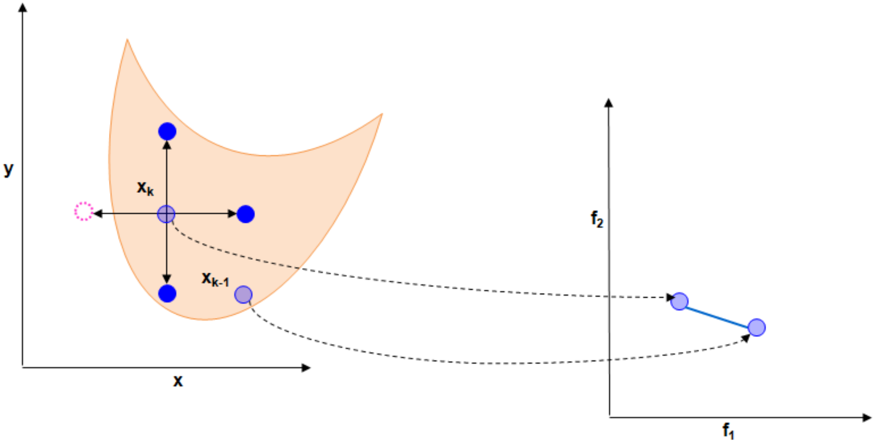
DMS example



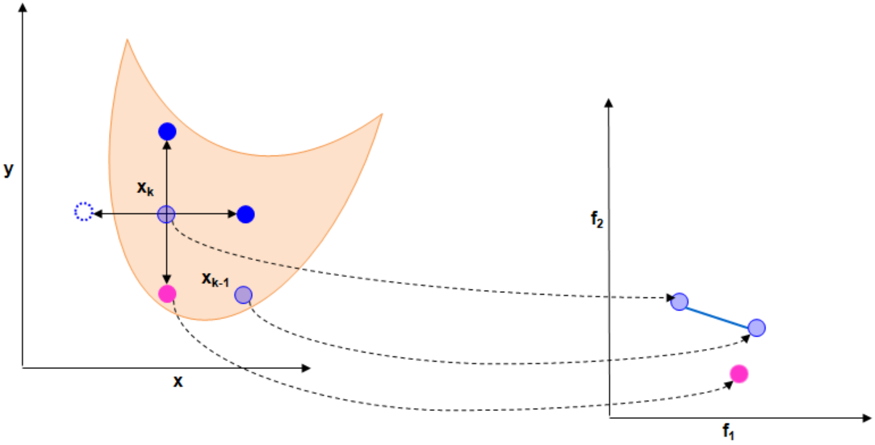
DMS example



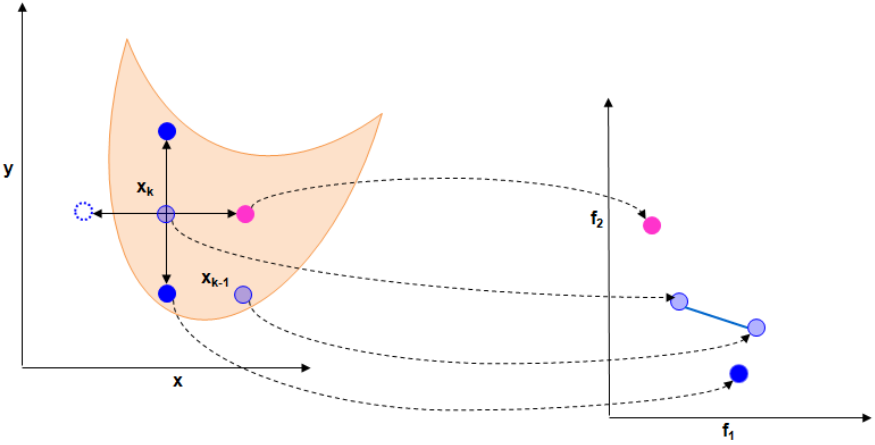
DMS example



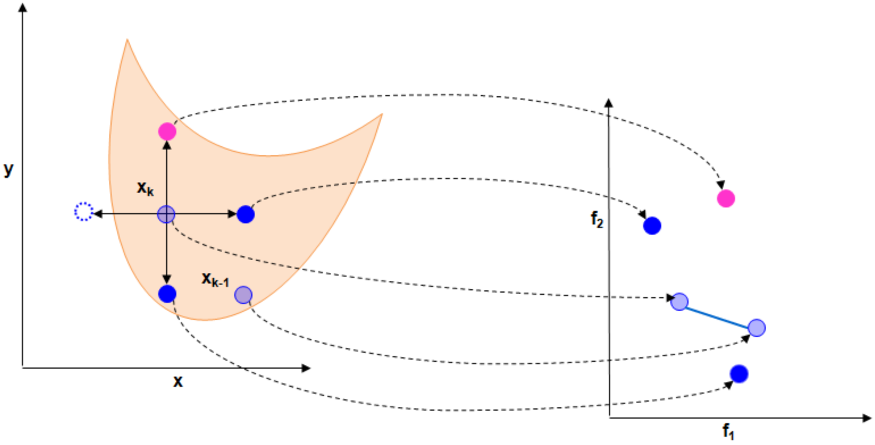
DMS example



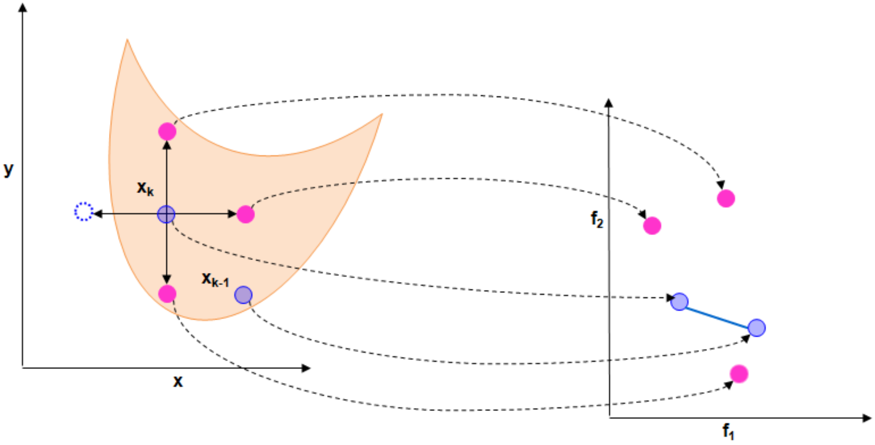
DMS example



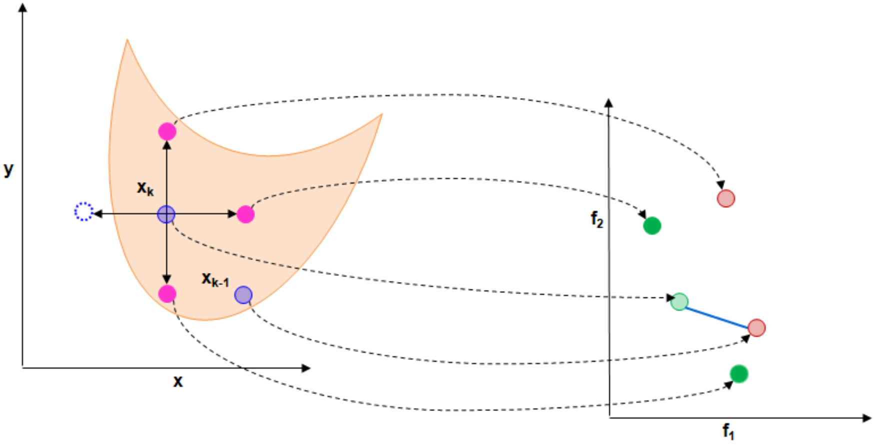
DMS example



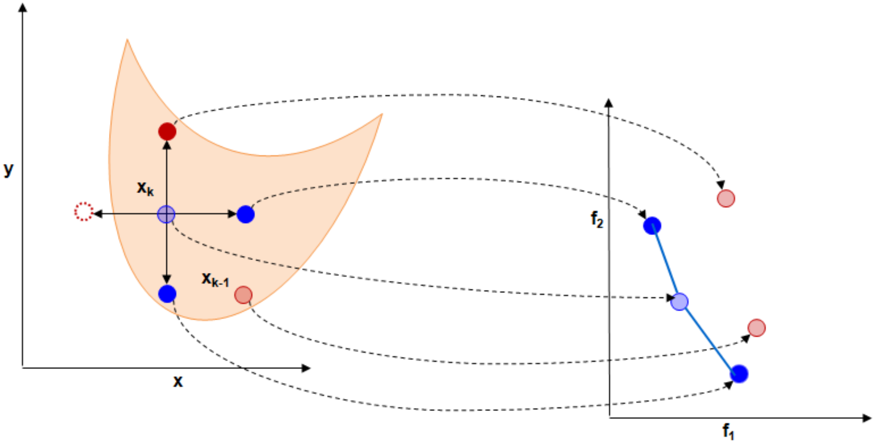
DMS example



DMS example



DMS example



DMS search & poll steps

- Evaluate a finite set of feasible points $\hookrightarrow L_{add}$.
- Remove dominated points from $L_k \cup L_{add} \hookrightarrow L_{filtered}$.
- Select list of feasible nondominated points $\hookrightarrow L_{trial}$.
- Compare L_{trial} to L_k (success if $L_{trial} \neq L_k$, unsuccess otherwise).

DMS search & poll steps

- Evaluate a finite set of feasible points $\hookrightarrow L_{add}$.
- Remove dominated points from $L_k \cup L_{add} \hookrightarrow L_{filtered}$.
- Select list of feasible nondominated points $\hookrightarrow L_{trial}$.
- Compare L_{trial} to L_k (success if $L_{trial} \neq L_k$, unsuccess otherwise).

DMS search & poll steps

- Evaluate a finite set of feasible points $\hookrightarrow L_{add}$.
- Remove dominated points from $L_k \cup L_{add} \hookrightarrow L_{filtered}$.
- Select list of feasible nondominated points $\hookrightarrow L_{trial}$.
- Compare L_{trial} to L_k (success if $L_{trial} \neq L_k$, unsuccess otherwise).

DMS search & poll steps

- Evaluate a finite set of feasible points $\hookrightarrow L_{add}$.
- Remove dominated points from $L_k \cup L_{add} \hookrightarrow L_{filtered}$.
- Select list of feasible nondominated points $\hookrightarrow L_{trial}$.
- Compare L_{trial} to L_k (success if $L_{trial} \neq L_k$, unsuccess otherwise).

Direct MultiSearch for MOO

(0) Initialization Choose $x_0 \in \Omega$ with $F(x_0) < +\infty$, $\alpha_0 > 0$. Set $L_0 = \{(x_0; \alpha_0)\}$.

Let $\mathcal{D}$ be a (possibly infinite) set of positive spanning sets.

For $k = 0, 1, 2, \ldots$

(1) Selection of iterate point

Order L_k and select $(x_k; \alpha_k) \in L_k$.

Direct MultiSearch for MOO

(0) Initialization Choose $x_0 \in \Omega$ with $F(x_0) < +\infty$, $\alpha_0 > 0$. Set $L_0 = \{(x_0; \alpha_0)\}$.

Let $\mathcal{D}$ be a (possibly infinite) set of positive spanning sets.

For $k = 0, 1, 2, \ldots$

(1) Selection of iterate point

Order L_k and select $(x_k; \alpha_k) \in L_k$.

Direct MultiSearch for MOO

(0) Initialization Choose $x_0 \in \Omega$ with $F(x_0) < +\infty$, $\alpha_0 > 0$. Set $L_0 = \{(x_0; \alpha_0)\}$.

Let $\mathcal{D}$ be a (possibly infinite) set of positive spanning sets.

For $k = 0, 1, 2, \ldots$

(1) Selection of iterate point

Order L_k and select $(x_k; \alpha_k) \in L_k$.

Direct MultiSearch for MOO

(0) Initialization Choose $x_0 \in \Omega$ with $F(x_0) < +\infty$, $\alpha_0 > 0$. Set $L_0 = \{(x_0; \alpha_0)\}$.

Let $\mathcal{D}$ be a (possibly infinite) set of positive spanning sets.

For $k = 0, 1, 2, \ldots$

(1) Selection of iterate point

Order L_k and select $(x_k; \alpha_k) \in L_k$.

Direct MultiSearch for MOO

(2) Search step (Optional)

Evaluate a finite set of points $L_{add} = \{(z_s; \alpha_k)\}_{s \in S}$ (in the mesh or using a forcing function).

$(L_k; L_{add}) \hookrightarrow L_{filtered} \hookrightarrow L_{trial}$

If success is achieved then set $L_{k+1} = L_{trial}$, declare the iteration and the search step successful, and skip the poll step.

Direct MultiSearch for MOO

(2) Search step (Optional)

Evaluate a finite set of points $L_{add} = \{(z_s; \alpha_k)\}_{s \in S}$ (in the mesh or using a forcing function).

$(L_k;L_{add}) \hookrightarrow L_{filtered} \hookrightarrow L_{trial}$

If success is achieved then set $L_{k+1} = L_{trial}$, declare the iteration and the search step successful, and skip the poll step.

Direct MultiSearch for MOO

(3) Poll step Evaluate $L_{add} = \{(x_k + \alpha_k d; \alpha_k),\ d \in D_k\}$, with $D_k \subseteq \mathcal{D}$

$(L_k;L_{add}) \hookrightarrow L_{filtered} \hookrightarrow L_{trial}$

If success is achieved then set $L_{k+1} = L_{trial}$, declare the iteration and the poll step successful

Otherwise declare the iteration (and the poll step) unsuccessful and set $L_{k+1} = L_{trial}$

Direct MultiSearch for MOO

(3) Poll step Evaluate $L_{add} = \{(x_k + \alpha_k d; \alpha_k),\ d \in D_k\}$, with $D_k \subseteq \mathcal{D}$
$(L_k;L_{add}) \hookrightarrow L_{filtered} \hookrightarrow L_{trial}$

If success is achieved then set $L_{k+1} = L_{trial}$, declare the iteration and the poll step successful

Otherwise declare the iteration (and the poll step) unsuccessful and set $L_{k+1} = L_{trial}$

Direct MultiSearch for MOO

(3) Poll step Evaluate $L_{add} = \{(x_k + \alpha_k d; \alpha_k),\ d \in D_k\}$, with $D_k \subseteq \mathcal{D}$

$(L_k; L_{add}) \hookrightarrow L_{filtered} \hookrightarrow L_{trial}$

If success is achieved then set $L_{k+1} = L_{trial}$, declare the iteration and the poll step successful

Otherwise declare the iteration (and the poll step) unsuccessful and set $L_{k+1} = L_{trial}$

Direct MultiSearch for MOO

(4) Step size update: If the iteration was successful then maintain the step size parameter ($\alpha_{k+1} = \alpha_k$) or double it ($\alpha_{k+1} = 2\alpha_k$) after two consecutive poll successes along the same direction.

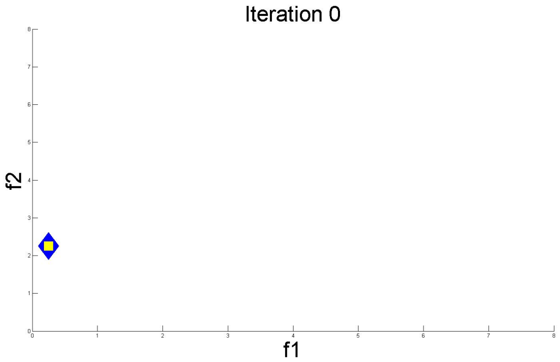
If the iteration was unsuccessful, halve the step size parameter ($\alpha_{k+1} = \alpha_k/2$).

Direct MultiSearch for MOO

(4) Step size update: If the iteration was successful then maintain the step size parameter ($\alpha_{k+1} = \alpha_k$) or double it ($\alpha_{k+1} = 2\alpha_k$) after two consecutive poll successes along the same direction.

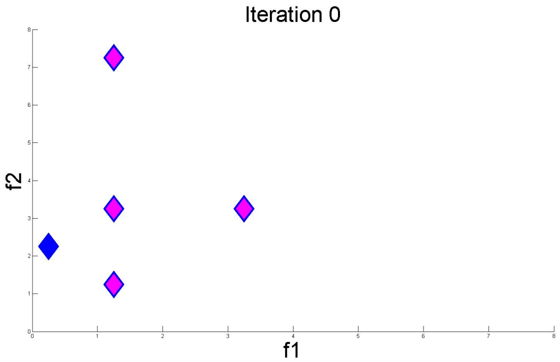
If the iteration was unsuccessful, halve the step size parameter ($\alpha_{k+1} = \alpha_k/2$).

Numerical Example — Problem SP1 [Huband et al.]



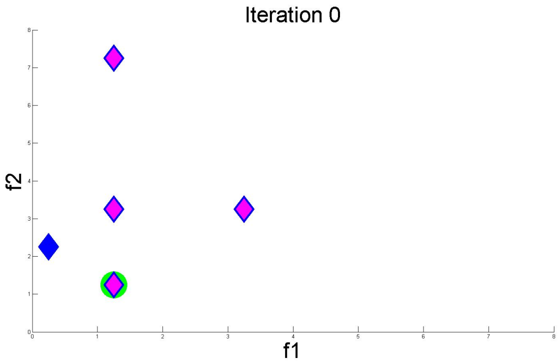
- ◆ Evaluated points since beginning.
- ■ Current iterate list.

Numerical example — problem SP1 [Huband et al.]



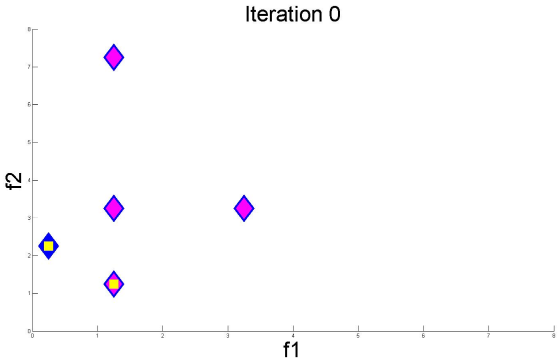
- Evaluated poll points.
- Evaluated points since beginning.

Numerical example — problem SP1 [Huband et al.]



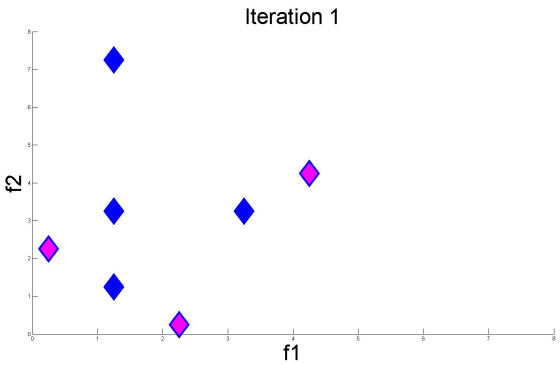
- Nondominated evaluated poll points.

Numerical example — problem SP1 [Huband et al.]



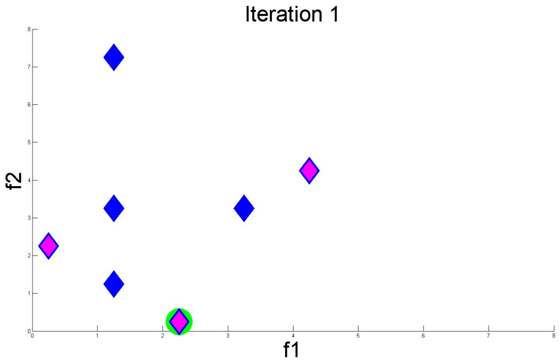
- Evaluated poll points.
- Evaluated points since beginning.
- Current iterate list.

Numerical example — problem SP1 [Huband et al.]



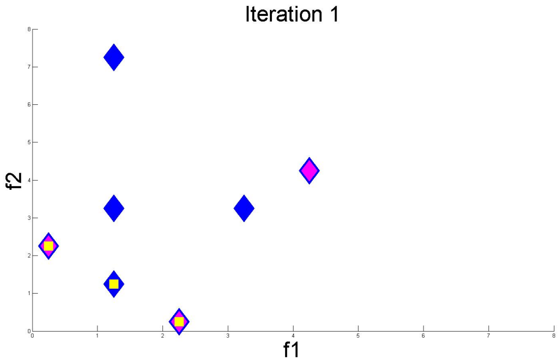
- Evaluated poll points.
- Evaluated points since beginning.

Numerical example — problem SP1 [Huband et al.]



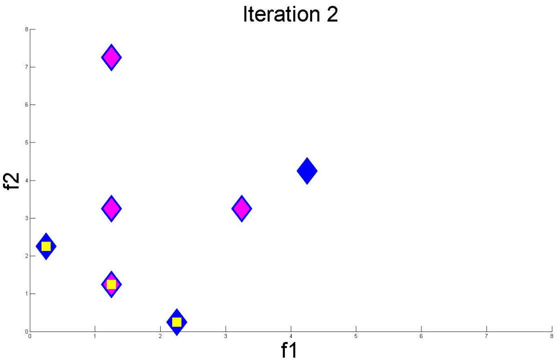
- Nondominated evaluated poll points.

Numerical example — problem SP1 [Huband et al.]



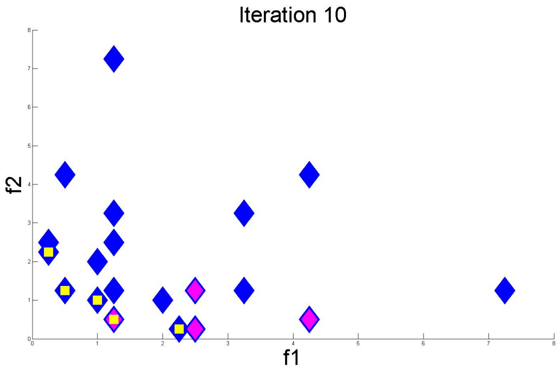
- Evaluated poll points.
- Evaluated points since beginning.
- Current iterate list.

Numerical example — problem SP1 [Huband et al.]



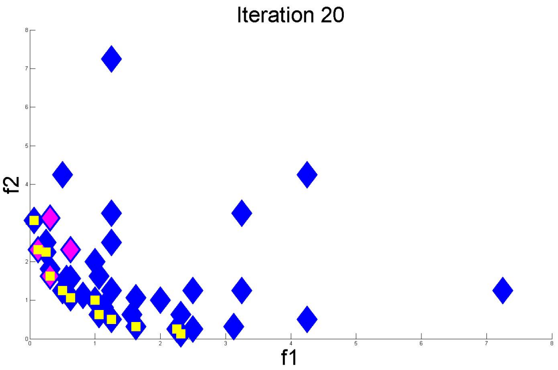
- Evaluated poll points.
- Evaluated points since beginning.
- Current iterate list.

Numerical example — problem SP1 [Huband et al.]



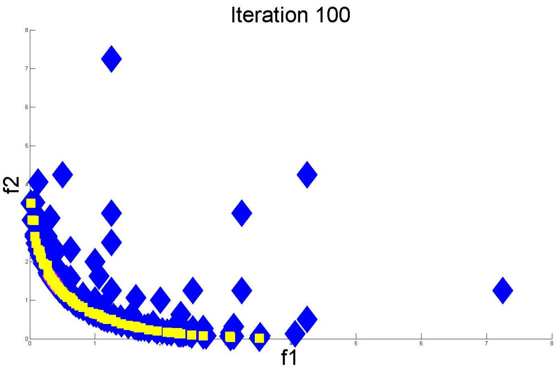
- Evaluated poll points.
- Evaluated points since beginning.
- Current iterate list.

Numerical example — problem SP1 [Huband et al.]



- ♦ Evaluated poll points.
- ♦ Evaluated points since beginning.
- ■ Current iterate list.

Numerical example — problem SP1 [Huband et al.]



- Evaluated poll points.
- Evaluated points since beginning.
- Current iterate list.

Refining subsequences and directions

For both globalization strategies (using the mesh or the forcing function in the search step), one also has:

Theorem (existence of refining subsequences)

There is at least a convergent subsequence of iterates $\{x_k\}_{k \in K}$ *corresponding to unsuccessful poll steps, such that* $\alpha_k \longrightarrow 0$ *in* K.

Definition

Let x_* *be the limit point of a convergent refining subsequence.*

Refining directions for x_* *are limit points of* $\{d_k/\|d_k\|\}_{k \in K}$ *where* $d_k \in D_k$ *and* $x_k + \alpha_k d_k \in \Omega$.

Refining subsequences and directions

For both globalization strategies (using the mesh or the forcing function in the search step), one also has:

Theorem (existence of refining subsequences)

There is at least a convergent subsequence of iterates $\{x_k\}_{k \in K}$ corresponding to unsuccessful poll steps, such that $\alpha_k \longrightarrow 0$ in K.

Definition

Let x_* be the limit point of a convergent refining subsequence.

Refining directions for x_* are limit points of $\{d_k / \|d_k\|\}_{k \in K}$ where $d_k \in D_k$ and $x_k + \alpha_k d_k \in \Omega$.

Refining subsequences and directions

For both globalization strategies (using the mesh or the forcing function in the search step), one also has:

Theorem (existence of refining subsequences)

There is at least a convergent subsequence of iterates $\{x_k\}_{k \in K}$ *corresponding to unsuccessful poll steps, such that* $\alpha_k \longrightarrow 0$ *in* K.

Definition

Let x_* *be the limit point of a convergent refining subsequence.*

Refining directions for x_* *are limit points of* $\{d_k/\|d_k\|\}_{k \in K}$ *where* $d_k \in D_k$ *and* $x_k + \alpha_k d_k \in \Omega$.

Pareto-Clarke critical point

Let us focus (again for simplicity) on the unconstrained case, $\Omega = \mathbb{R}^n$.

Definition

x_* is a Pareto-Clarke critical point of F (Lipschitz continuous near x_*) if

$$\forall d \in \mathbb{R}^n, \exists j = j(d), f_j^\circ(x_*; d) \geq 0.$$

Analysis of DMS

Assumption

- $\{x_k\}_{k \in K}$ refining subsequence converging to x_*.
- F Lipschitz continuous near x_*.

Theorem

If v is a refining direction for x_* then

$$\exists j = j(d) : f_j^\circ(x_*; d) \geq 0.$$

Analysis of DMS

Assumption

- $\{x_k\}_{k \in K}$ refining subsequence converging to x_*.
- F Lipschitz continuous near x_*.

Theorem

If v is a refining direction for x_* then

$$\exists j = j(d) : f_j^\circ(x_*; d) \geq 0.$$

Analysis of DMS

Assumption

- $\{x_k\}_{k \in K}$ refining subsequence converging to x_*.
- F Lipschitz continuous near x_*.

Theorem

If v is a refining direction for x_* then

$$\exists j = j(d) : f_j^\circ(x_*; d) \geq 0.$$

Analysis of DMS

Assumption

- $\{x_k\}_{k \in K}$ refining subsequence converging to x_*.
- F Lipschitz continuous near x_*.

Theorem

If v is a refining direction for x_* then

$$\exists j = j(d) : f_j^\circ(x_*; d) \geq 0.$$

Analysis of DMS

Theorem

If the set of refining directions for x_* is dense in $\mathbb{R}^n$, then x_* is a Pareto-Clarke critical point.

Notes

- When $m = 1$, we obtain the result presented before.
- The convergence analysis is valid for multiobjective problems with general nonlinear constraints.

Analysis of DMS

Theorem

If the set of refining directions for x_* is dense in $\mathbb{R}^n$, then x_* is a Pareto-Clarke critical point.

Notes

- When $m = 1$, we obtain the result presented before.
- This convergence analysis is valid for multiobjective problems with general nonlinear constraints.

Analysis of DMS

Theorem

If the set of refining directions for x_* is dense in $\mathbb{R}^n$, then x_* is a Pareto-Clarke critical point.

Notes

- When $m = 1$, we obtain the result presented before.
- This convergence analysis is valid for multiobjective problems with general nonlinear constraints.

Analysis of DMS

Theorem

If the set of refining directions for x_* is dense in $\mathbb{R}^n$, then x_* is a Pareto-Clarke critical point.

Notes

- When $m = 1$, we obtain the result presented before.
- This convergence analysis is valid for multiobjective problems with general nonlinear constraints.

Outline

Numerical testing framework

Problems

- 100 bound constrained MOO problems (AMPL models available at http://www.mat.uc.pt/dms).
- Number of variables between 1 and 30.
- Number of objectives between 2 and 4.

Solvers

- DMS tested against 8 different MOO solvers (complete results available at http://www.mat.uc.pt/dms).
- Results reported only for
  AMOSA – simulated annealing code.
  BIMADS – based on Mesh Adaptive Direct Search.
  NSGA-II (C version) – genetic algorithm code.

All solvers tested with default values.

Numerical testing framework

Problems

- 100 bound constrained MOO problems (AMPL models available at http://www.mat.uc.pt/dms).
- Number of variables between 1 and 30.
- Number of objectives between 2 and 4.

Solvers

- DMS tested against 8 different MOO solvers (complete results available at http://www.mat.uc.pt/dms).
- Results reported only for
  AMOSA – simulated annealing code.
  BIMADS – based on Mesh Adaptive Direct Search.
  NSGA-II (C version) – genetic algorithm code.

All solvers tested with default values.

Numerical testing framework

Problems

- 100 bound constrained MOO problems (AMPL models available at http://www.mat.uc.pt/dms).
- Number of variables between 1 and 30.
- Number of objectives between 2 and 4.

Solvers

- DMS tested against 8 different MOO solvers (complete results available at http://www.mat.uc.pt/dms).
- Results reported only for
  AMOSA – simulated annealing code.
  BIMADS – based on Mesh Adaptive Direct Search.
  NSGA-II (C version) – genetic algorithm code.

All solvers tested with default values.

Numerical testing framework

Problems

- 100 bound constrained MOO problems (AMPL models available at http://www.mat.uc.pt/dms).
- Number of variables between 1 and 30.
- Number of objectives between 2 and 4.

Solvers

- DMS tested against 8 different MOO solvers (complete results available at http://www.mat.uc.pt/dms).
- Results reported only for
  AMOSA – simulated annealing code.
  BIMADS – based on Mesh Adaptive Direct Search.
  NSGA-II (C version) – genetic algorithm code.

All solvers tested with default values.

Numerical testing framework

Problems

- 100 bound constrained MOO problems (AMPL models available at http://www.mat.uc.pt/dms).
- Number of variables between 1 and 30.
- Number of objectives between 2 and 4.

Solvers

- DMS tested against 8 different MOO solvers (complete results available at http://www.mat.uc.pt/dms).
- Results reported only for
  AMOSA – simulated annealing code.
  BIMADS – based on Mesh Adaptive Direct Search.
  NSGA-II (C version) – genetic algorithm code.

All solvers tested with default values.

DMS numerical options

- No search step.
- List initialization: sample along the line ℓ–u.
- List selection: all current nondominated points.
- List ordering: new points added at the end of the list, poll center moved to the end of the list.
- Positive basis: $[I \;\; -I]$.
- Step size parameter: $\alpha_0 = 1$, halved at unsuccessful iterations.
- Stopping criteria: minimum step size of 10^{-3} or a maximum of 20000 function evaluations.

DMS numerical options

- No search step.
- List initialization: sample along the line ℓ–u.
- List selection: all current nondominated points.
- List ordering: new points added at the end of the list, poll center moved to the end of the list.
- Positive basis: $[I \ \ -I]$.
- Step size parameter: $\alpha_0 = 1$, halved at unsuccessful iterations.
- Stopping criteria: minimum step size of 10^{-3} or a maximum of 20000 function evaluations.

DMS numerical options

- No search step.
- List initialization: sample along the line ℓ–u.
- List selection: all current nondominated points.
- List ordering: new points added at the end of the list, poll center moved to the end of the list.
- Positive basis: $[I \;\; -I]$.
- Step size parameter: $\alpha_0 = 1$, halved at unsuccessful iterations.
- Stopping criteria: minimum step size of 10^{-3} or a maximum of 20000 function evaluations.

DMS numerical options

- No search step.
- List initialization: sample along the line ℓ–u.
- List selection: all current nondominated points.
- List ordering: new points added at the end of the list, poll center moved to the end of the list.
- Positive basis: $[I \ \ -I]$.
- Step size parameter: $\alpha_0 = 1$, halved at unsuccessful iterations.
- Stopping criteria: minimum step size of 10^{-3} or a maximum of 20000 function evaluations.

DMS numerical options

- No search step.
- List initialization: sample along the line ℓ–u.
- List selection: all current nondominated points.
- List ordering: new points added at the end of the list, poll center moved to the end of the list.
- Positive basis: $[I \ \ -I]$.
- Step size parameter: $\alpha_0 = 1$, halved at unsuccessful iterations.
- Stopping criteria: minimum step size of 10^{-3} or a maximum of 20000 function evaluations.

DMS numerical options

- No search step.
- List initialization: sample along the line ℓ–u.
- List selection: all current nondominated points.
- List ordering: new points added at the end of the list, poll center moved to the end of the list.
- Positive basis: $[I \ \ -I]$.
- Step size parameter: $\alpha_0 = 1$, halved at unsuccessful iterations.
- Stopping criteria: minimum step size of 10^{-3} or a maximum of 20000 function evaluations.

DMS numerical options

- No search step.
- List initialization: sample along the line ℓ–u.
- List selection: all current nondominated points.
- List ordering: new points added at the end of the list, poll center moved to the end of the list.
- Positive basis: $[I \quad -I]$.
- Step size parameter: $\alpha_0 = 1$, halved at unsuccessful iterations.
- Stopping criteria: minimum step size of 10^{-3} or a maximum of 20000 function evaluations.

Performance metrics — Purity

$F_{p,s}$ (approximated Pareto front computed by solver s for problem p).

F_p (approximated Pareto front computed for problem p, using results for all solvers).

Purity value for solver s on problem p:

$$\frac{|F_{p,s} \cap F_p|}{|F_{p,s}|}.$$

Performance profiles [Dolan and Moré]

Let $t_{p,s}$ be a metric for which lower values indicate better performance.

Consider

$$\rho_s(\tau) = \frac{|\{p \in \mathcal{P} : r_{p,s} \leq \tau\}|}{|\mathcal{P}|}$$

with $r_{p,s} = t_{p,s} / \min\{t_{p,s} : s \in \mathcal{S}\}$, where $\mathcal{S}$ is the set of solvers and $\mathcal{P}$ is the set of problems.

- Incorporates results for all problems and all solvers.
- Allows to access 'efficiency' and robustness.
- $\rho_s(1)$ represents 'efficiency' of solver s.
- $\rho_s(\tau)$, with τ large, gives robustness of solver s.
- The lower the value $t_{p,s}$ the better.

Performance profiles [Dolan and Moré]

Let $t_{p,s}$ be a metric for which lower values indicate better performance.

Consider

$$\rho_s(\tau) = \frac{|\{p \in \mathcal{P} : r_{p,s} \leq \tau\}|}{|\mathcal{P}|}$$

with $r_{p,s} = t_{p,s} / \min\{t_{p,s} : s \in \mathcal{S}\}$, where $\mathcal{S}$ is the set of solvers and $\mathcal{P}$ is the set of problems.

- Incorporates results for all problems and all solvers.
- Allows to access 'efficiency' and robustness.
- $\rho_s(1)$ represents 'efficiency' of solver s.
- $\rho_s(\tau)$, with τ large, gives robustness of solver s.
- The lower the value $t_{p,s}$ the better.

Performance profiles [Dolan and Moré]

Let $t_{p,s}$ be a metric for which lower values indicate better performance.

Consider

$$\rho_s(\tau) = \frac{|\{p \in \mathcal{P} : r_{p,s} \leq \tau\}|}{|\mathcal{P}|}$$

with $r_{p,s} = t_{p,s} / \min\{t_{p,s} : s \in \mathcal{S}\}$, where $\mathcal{S}$ is the set of solvers and $\mathcal{P}$ is the set of problems.

- Incorporates results for all problems and all solvers.
- Allows to access 'efficiency' and robustness.
- $\rho_s(1)$ represents 'efficiency' of solver s.
- $\rho_s(\tau)$, with τ large, gives robustness of solver s.
- The lower the value $t_{p,s}$ the better.

Performance profiles [Dolan and Moré]

Let $t_{p,s}$ be a metric for which lower values indicate better performance.

Consider

$$\rho_s(\tau) = \frac{|\{p \in \mathcal{P} : r_{p,s} \leq \tau\}|}{|\mathcal{P}|}$$

with $r_{p,s} = t_{p,s} / \min\{t_{p,s} : s \in \mathcal{S}\}$, where $\mathcal{S}$ is the set of solvers and $\mathcal{P}$ is the set of problems.

- Incorporates results for all problems and all solvers.
- Allows to access 'efficiency' and robustness.
- $\rho_s(1)$ represents 'efficiency' of solver s.
- $\rho_s(\tau)$, with τ large, gives robustness of solver s.
- The lower the value $t_{p,s}$ the better.

Performance profiles [Dolan and Moré]

Let $t_{p,s}$ be a metric for which lower values indicate better performance.

Consider

$$\rho_s(\tau) = \frac{|\{p \in \mathcal{P} : r_{p,s} \leq \tau\}|}{|\mathcal{P}|}$$

with $r_{p,s} = t_{p,s} / \min\{t_{p,s} : s \in \mathcal{S}\}$, where $\mathcal{S}$ is the set of solvers and $\mathcal{P}$ is the set of problems.

- Incorporates results for all problems and all solvers.
- Allows to access 'efficiency' and robustness.
- $\rho_s(1)$ represents 'efficiency' of solver s.
- $\rho_s(\tau)$, with τ large, gives robustness of solver s.
- The lower the value $t_{p,s}$ the better.

Performance profiles [Dolan and Moré]

Let $t_{p,s}$ be a metric for which lower values indicate better performance.

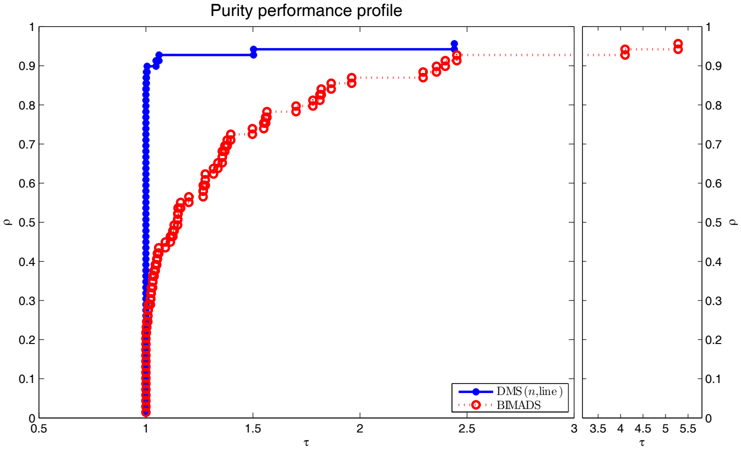
Consider

$$\rho_s(\tau) = \frac{|\{p \in \mathcal{P} : r_{p,s} \leq \tau\}|}{|\mathcal{P}|}$$

with $r_{p,s} = t_{p,s} / \min\{t_{p,s} : s \in \mathcal{S}\}$, where $\mathcal{S}$ is the set of solvers and $\mathcal{P}$ is the set of problems.

- Incorporates results for all problems and all solvers.
- Allows to access 'efficiency' and robustness.
- $\rho_s(1)$ represents 'efficiency' of solver s.
- $\rho_s(\tau)$, with τ large, gives robustness of solver s.
- The lower the value $t_{p,s}$ the better.

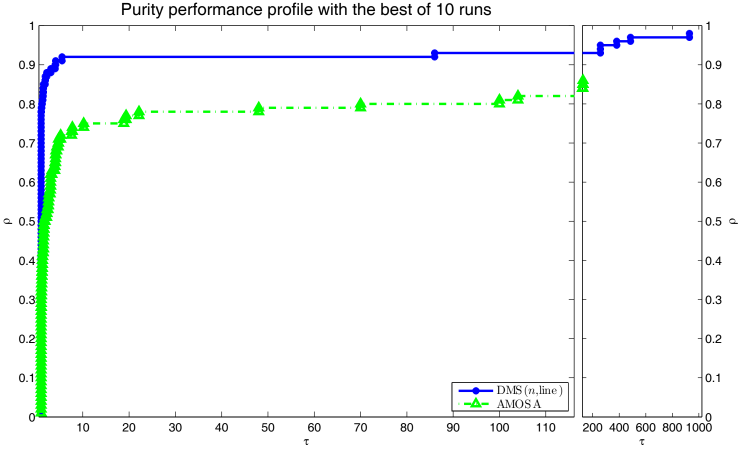
Comparing DMS to other solvers (Purity)



Purity Metric (percentage of points generated in the reference Pareto front)

$$t_{p,s} = \frac{|F_{p,s}|}{|F_{p,s} \cap F_p|}$$

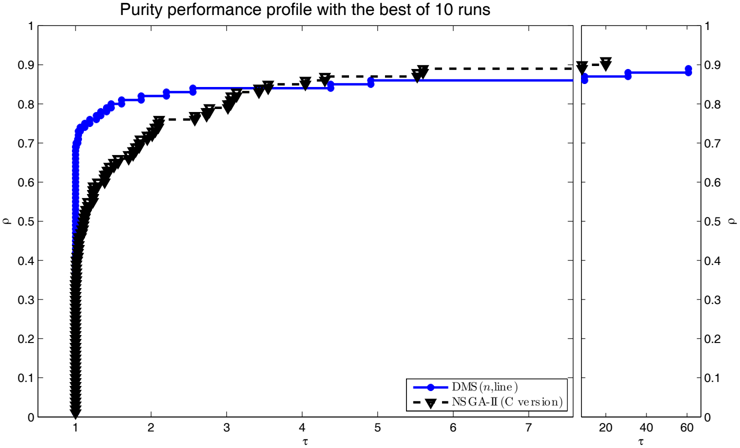
Comparing DMS to other solvers (Purity)



Purity Metric (percentage of points generated in the reference Pareto front)

$$t_{p,s} = \frac{|F_{p,s}|}{|F_{p,s} \cap F_p|}$$

Comparing DMS to other solvers (Purity)



Purity Metric (percentage of points generated in the reference Pareto front)

$$t_{p,s} = \frac{|F_{p,s}|}{|F_{p,s} \cap F_p|}$$

Performance metrics — Spread

Gamma Metric

(largest gap in the Pareto front)

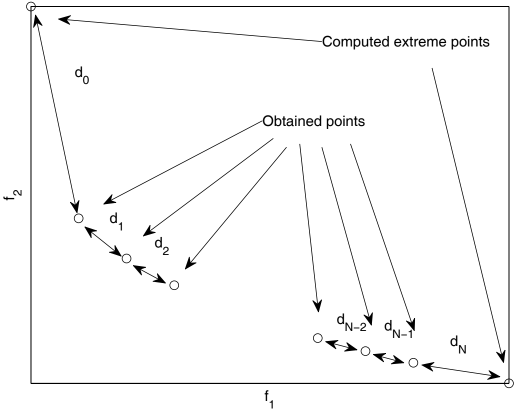
$$\Gamma_{p,s} \;=\; \max_{i \in \{0,\ldots,N\}} \{d_i\}$$

Delta Metric

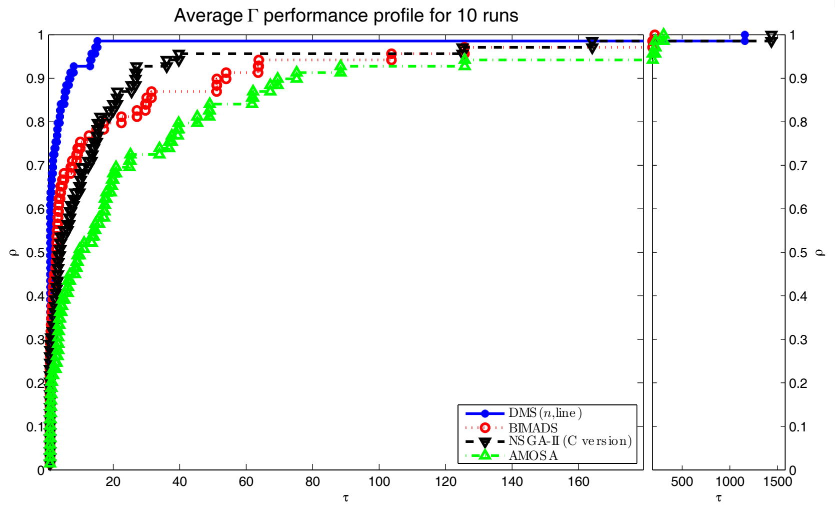
(uniformity of gaps in the Pareto front)

$$\Delta_{p,s} \;=\; \frac{d_0 + d_N + \sum_{i=1}^{N-1} |d_i - \bar{d}|}{d_0 + d_N + (N-1)\bar{d}}$$

where $\bar{d}$ is the d_i average



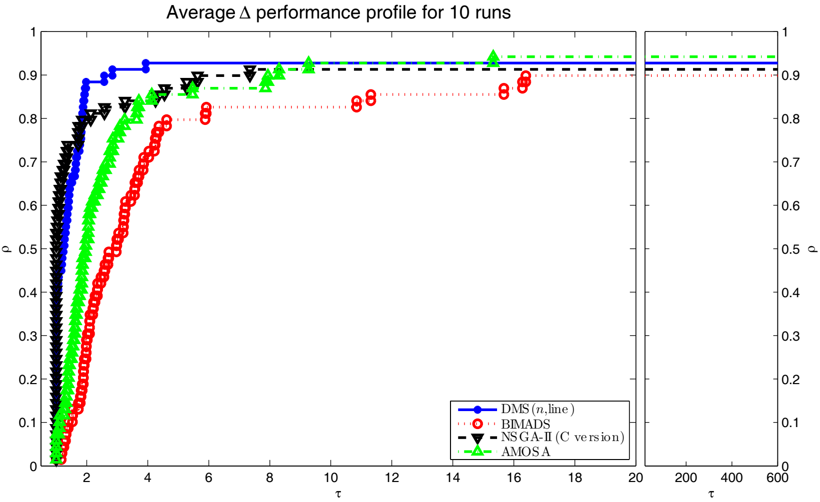
Comparing DMS to other solvers (Spread)



Gamma Metric (largest gap in the Pareto front)

$$\Gamma_{p,s} = \max_{i \in \{0,\ldots,N\}} \{d_i\}$$

Comparing DMS to other solvers (Spread)



Delta Metric (uniformity of gaps in the Pareto front)

$$\Delta_{p,s} = \frac{d_0 + d_N + \sum_{i=1}^{N-1} |d_i - \bar{d}|}{d_0 + d_N + (N-1)\bar{d}}$$

Data profiles [Moré and Wild]

Indicate how likely is an algorithm to 'solve' a problem, given some computational budget.

Let $h_{p,s}$ be the number of function evaluations required for solver s to solve problem p.

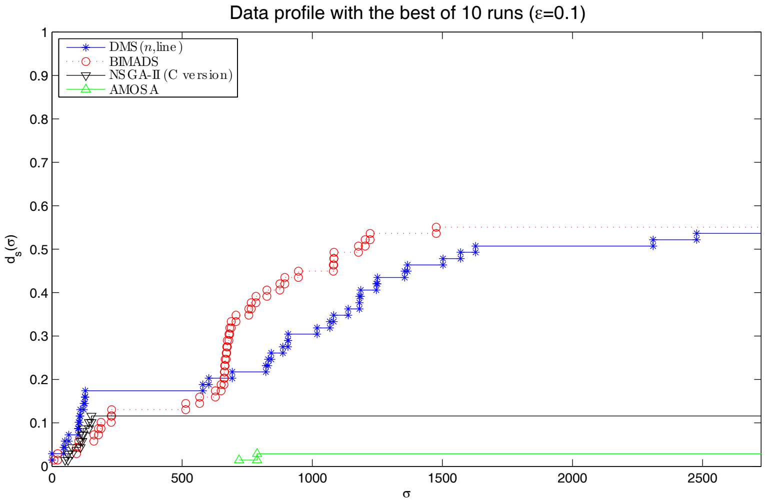
Consider

$$d_s(\sigma) \;=\; \frac{|\{p \in \mathcal{P} : h_{p,s} \leq \sigma\}|}{|\mathcal{P}|}.$$

Problem solved to ϵ–accuracy:

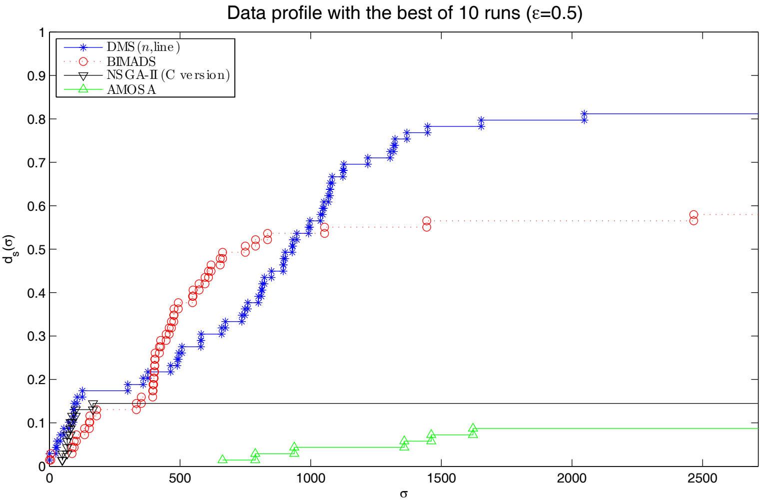
$$\frac{|F_{p,s} \cap F_p|}{|F_p|/|\mathcal{S}|} \;\geq\; 1 - \varepsilon.$$

Comparing DMS to other solvers



maximum function evaluations = 5000

Comparing DMS to other solvers



maximum function evaluations = 5000

Outline

Conclusions and references

- Development and analysis of a novel approach (Direct MultiSearch) for MOO, generalizing ALL direct-search methods.
- Direct MultiSearch (DMS) exhibits highly competitive numerical results for MOO.

DMS (Matlab implementation) and problems (coded in AMPL) freely available at: `http://www.mat.uc.pt/dms`.

A. L. Custódio, J. F. A. Madeira, A. I. F. Vaz, and L. N. Vicente, Direct multisearch for multiobjective optimization, preprint 10-18, Dept. of Mathematics, Univ. Coimbra, 2010.

Conclusions and references

- Development and analysis of a novel approach (Direct MultiSearch) for MOO, generalizing ALL direct-search methods.

- Direct MultiSearch (DMS) exhibits highly competitive numerical results for MOO.

DMS (Matlab implementation) and problems (coded in AMPL) freely available at: `http://www.mat.uc.pt/dms`.

A. L. Custódio, J. F. A. Madeira, A. I. F. Vaz, and L. N. Vicente, Direct multisearch for multiobjective optimization, preprint 10-18, Dept. of Mathematics, Univ. Coimbra, 2010.

Conclusions and references

- Development and analysis of a novel approach (Direct MultiSearch) for MOO, generalizing ALL direct-search methods.
- Direct MultiSearch (DMS) exhibits highly competitive numerical results for MOO.

DMS (Matlab implementation) and problems (coded in AMPL) freely available at: `http://www.mat.uc.pt/dms`.

A. L. Custódio, J. F. A. Madeira, A. I. F. Vaz, and L. N. Vicente, Direct multisearch for multiobjective optimization, preprint 10-18, Dept. of Mathematics, Univ. Coimbra, 2010.

Conclusions and references

- Development and analysis of a novel approach (Direct MultiSearch) for MOO, generalizing ALL direct-search methods.
- Direct MultiSearch (DMS) exhibits highly competitive numerical results for MOO.

DMS (Matlab implementation) and problems (coded in AMPL) freely available at: `http://www.mat.uc.pt/dms`.

A. L. Custódio, J. F. A. Madeira, A. I. F. Vaz, and L. N. Vicente, Direct multisearch for multiobjective optimization, preprint 10-18, Dept. of Mathematics, Univ. Coimbra, 2010.

Optimization 2011 (July 24–27, Portugal)

2011
optimization
Faculdade de Ciências e Tecnologia
Universidade Nova de Lisboa
Caparica *Campus*
July 24-27

plenary speakers

Gilbert Laporte | HEC Montréal
New trends in vehicle routing

Jean Bernard Lasserre | LAAS-CNRS, Toulouse
Moments and semidefinite relaxations for parametric optimization

José Mario Martínez | State University of Campinas
Unifying inexact restoration, SQP, and augmented Lagrangian methods

Mauricio G.C. Resende | AT&T Labs - Research
Using metaheuristics to solve real optimization problems in telecommunications

Nick Sahinidis | Carnegie Mellon University
Recent advances in nonconvex optimization

Stephen J. Wright | University of Wisconsin
Algorithms and applications in sparse optimization