An SQP-type method for constrained and unconstrained nonlinear multiobjective optimization

Jörg Fliege[1] A. Ismael F. Vaz[2]

[1]University Southampton, UK

[2]University of Minho, Portugal

Recent Advances in Multi-Objective Optimization

24 June 2016

Outline

Outline

Outline

Outline

Outline

Outline

Multiobjective constrained optimization

Constrained multiobjective optimization problem

$$\min_{x \in \Omega} \quad f(x) = (f_1(x), \ldots, f_m(x))^T$$

with

$$\Omega = \{x \in [\ell, u] \subseteq \mathbb{R}^n : g_j(x) \leq 0, j = 1, \ldots, p, \quad h_l(x) = 0, l = 1, \ldots, q\}$$

- $\ell \in (\mathbb{R} \cup \{-\infty\})^n$, $u \in (\mathbb{R} \cup \{+\infty\})^n$;
- All objective functions are at least C^2;
- All constraint functions are at least C^1;
- We also allow unconstrained or box-constrained optimization ($p, q = 0$ and/or $\ell = \{-\infty\}^n, u = \{\infty\}^n$).

Multiobjective constrained optimization

Constrained multiobjective optimization problem

$$\min_{x \in \Omega} \quad f(x) = (f_1(x), \ldots, f_m(x))^T$$

with

$$\Omega = \{x \in [\ell, u] \subseteq \mathbb{R}^n : g_j(x) \leq 0, j = 1, \ldots, p, \quad h_l(x) = 0, l = 1, \ldots, q\}$$

- $\ell \in (\mathbb{R} \cup \{-\infty\})^n$, $u \in (\mathbb{R} \cup \{+\infty\})^n$;
- All objective functions are at least C^2;
- All constraint functions are at least C^1;
- We also allow unconstrained or box-constrained optimization ($p, q = 0$ and/or $\ell = \{-\infty\}^n, u = \{\infty\}^n$).

Multiobjective constrained optimization

Constrained multiobjective optimization problem

$$\min_{x \in \Omega} \quad f(x) = (f_1(x), \ldots, f_m(x))^T$$

with

$$\Omega = \{x \in [\ell, u] \subseteq \mathbb{R}^n : g_j(x) \leq 0, j = 1, \ldots, p, \quad h_l(x) = 0, l = 1, \ldots, q\}$$

- $\ell \in (\mathbb{R} \cup \{-\infty\})^n$, $u \in (\mathbb{R} \cup \{+\infty\})^n$;
- All objective functions are at least C^2;
- All constraint functions are at least C^1;
- We also allow unconstrained or box-constrained optimization ($p, q = 0$ and/or $\ell = \{-\infty\}^n, u = \{\infty\}^n$).

Multiobjective constrained optimization

Constrained multiobjective optimization problem

$$\min_{x \in \Omega} \quad f(x) = (f_1(x), \ldots, f_m(x))^T$$

with

$$\Omega = \{x \in [\ell, u] \subseteq \mathbb{R}^n : g_j(x) \leq 0, j = 1, \ldots, p, \quad h_l(x) = 0, l = 1, \ldots, q\}$$

- $\ell \in (\mathbb{R} \cup \{-\infty\})^n$, $u \in (\mathbb{R} \cup \{+\infty\})^n$;
- All objective functions are at least C^2;
- All constraint functions are at least C^1;
- We also allow unconstrained or box-constrained optimization ($p, q = 0$ and/or $\ell = \{-\infty\}^n, u = \{\infty\}^n$).

Multiobjective constrained optimization

Constrained multiobjective optimization problem

$$\min_{x \in \Omega} \quad f(x) = (f_1(x), \ldots, f_m(x))^T$$

with

$$\Omega = \{x \in [\ell, u] \subseteq \mathbb{R}^n : g_j(x) \leq 0, j = 1, \ldots, p, \quad h_l(x) = 0, l = 1, \ldots, q\}$$

- $\ell \in (\mathbb{R} \cup \{-\infty\})^n$, $u \in (\mathbb{R} \cup \{+\infty\})^n$;
- All objective functions are at least C^2;
- All constraint functions are at least C^1;
- We also allow unconstrained or box-constrained optimization ($p, q = 0$ and/or $\ell = \{-\infty\}^n, u = \{\infty\}^n$).

Outline

Algorithm main lines

- Does not aggregate any of the objective functions
- Uses SQP based techniques for MOO
- Keeps a list of nondominated points
- Constraints violations are considered as additional objectives in the linesearch steps.
- Tries to capture the whole Pareto front from two algorithmic stages: search and refining

Algorithm main lines

- Does not aggregate any of the objective functions
- Uses SQP based techniques for MOO
- Keeps a list of nondominated points
- Constraints violations are considered as additional objectives in the linesearch steps.
- Tries to capture the whole Pareto front from two algorithmic stages: search and refining

Algorithm main lines

- Does not aggregate any of the objective functions
- Uses SQP based techniques for MOO
- Keeps a list of nondominated points
- Constraints violations are considered as additional objectives in the linesearch steps.
- Tries to capture the whole Pareto front from two algorithmic stages: search and refining

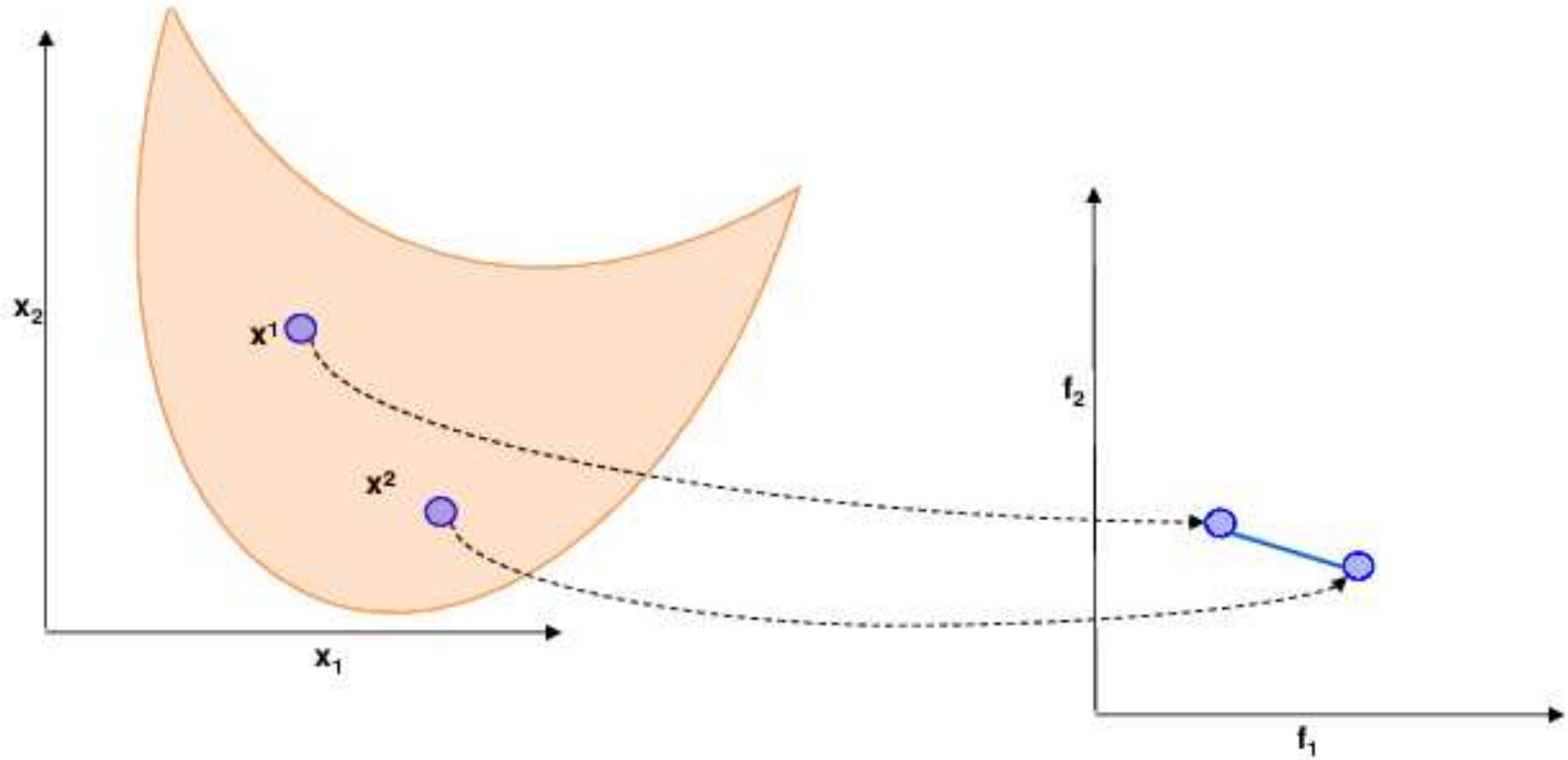
Algorithm main lines

- Does not aggregate any of the objective functions
- Uses SQP based techniques for MOO
- Keeps a list of nondominated points
- Constraints violations are considered as additional objectives in the linesearch steps.
- Tries to capture the whole Pareto front from two algorithmic stages: search and refining

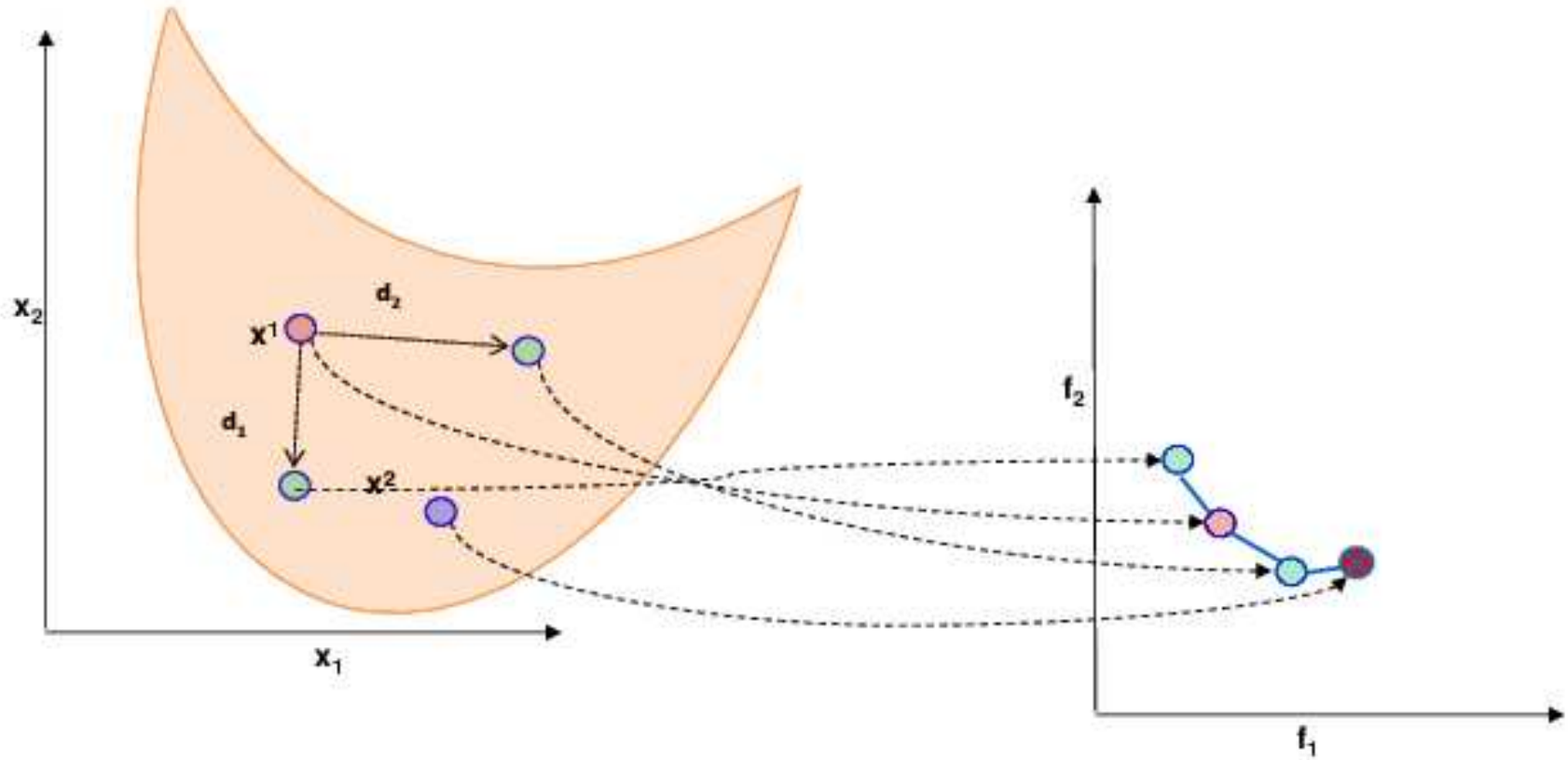
Algorithm main lines

- Does not aggregate any of the objective functions
- Uses SQP based techniques for MOO
- Keeps a list of nondominated points
- Constraints violations are considered as additional objectives in the linesearch steps.
- Tries to capture the whole Pareto front from two algorithmic stages: search and refining

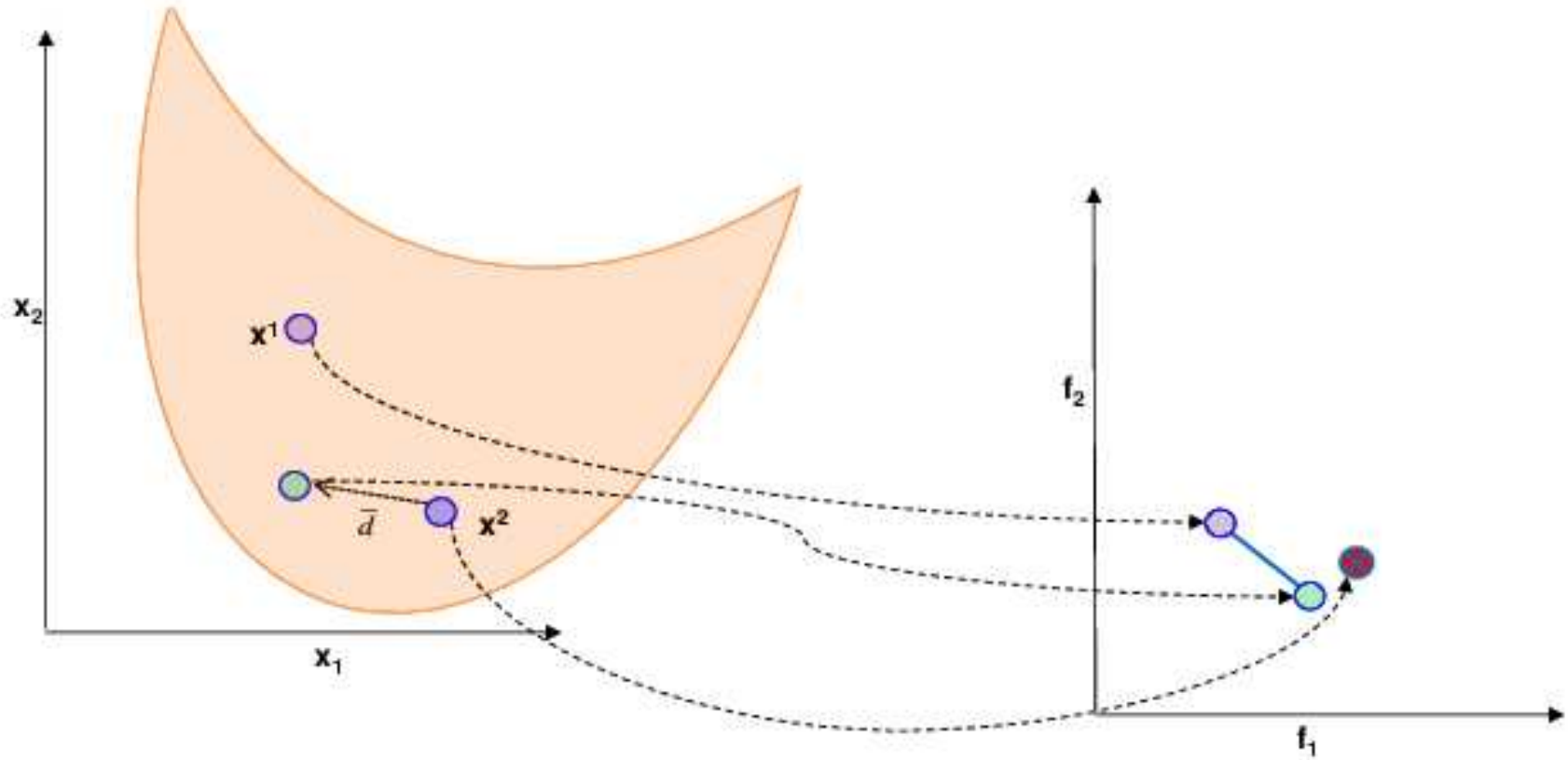
Algorithm illustrated - setup



Algorithm illustrated - spread



Algorithm illustrated - refining



Search direction computation

For each x_k in the list of nondominated points:

Spread $(i = 1, \ldots, m)$

$$
\begin{aligned}
d_i \in \arg\min_{d \in \mathbb{R}^n} \quad & \nabla f_i(x_k)^T d + \frac{1}{2} d^T H_i d \\
\text{s.t.} \quad & g_j(x_k) + \nabla g_j(x_k)^T d \leq 0, \quad j = 1, \ldots, p \\
& h_l(x_k) + \nabla h_l(x_k)^T d = 0, \quad l = 1, \ldots, q \\
& \ell \leq x_k + d \leq u
\end{aligned}
$$

where H_i is a positive definite matrix.

d_i is a descent direction for f_i.

Linesearch: test $x_k + \alpha d_i$ $(\alpha = 2^t,\ t = 0, 1, 2, \ldots)$ for being nondominated.

Search direction computation

For each x_k in the list of nondominated points:

Refining

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & \sum_{i=1}^{m} f_i(x) \\ \text{s.t.} \quad & f_i(x) \leq f_i(x_k), \quad i = 1, \ldots, m \\ & g_j(x) \leq 0, \quad j = 1, \ldots, p \\ & h_l(x) = 0, \quad l = 1, \ldots, q \end{aligned}$$

Iterations of an SQP-type method for this problem are carried out, using x_k as a starting point.

Some theoretical considerations

- From spread stage we obtain new points.
- The spread stage performs a finite number of iterations. (No asymptotics here!)
- The refining stage drives all the available nondominated points to Pareto criticality,
- by obtaining a new point that decreases or maintains all the objective function values.
- (Local) Pareto criticality can be verified based on the refining single-objective optimization problem.
- Convergence theory available for the proposed algorithm.

Some theoretical considerations

- From spread stage we obtain new points.
- The spread stage performs a finite number of iterations. (No asymptotics here!)
- The refining stage drives all the available nondominated points to Pareto criticality,
- by obtaining a new point that decreases or maintains all the objective function values.
- (Local) Pareto criticality can be verified based on the refining single-objective optimization problem.
- Convergence theory available for the proposed algorithm.

Some theoretical considerations

- From spread stage we obtain new points.
- The spread stage performs a finite number of iterations. (No asymptotics here!)
- The refining stage drives all the available nondominated points to Pareto criticality,
- by obtaining a new point that decreases or maintains all the objective function values.
- (Local) Pareto criticality can be verified based on the refining single-objective optimization problem.
- Convergence theory available for the proposed algorithm.

Some theoretical considerations

- From spread stage we obtain new points.
- The spread stage performs a finite number of iterations. (No asymptotics here!)
- The refining stage drives all the available nondominated points to Pareto criticality,
- by obtaining a new point that decreases or maintains all the objective function values.
- (Local) Pareto criticality can be verified based on the refining single-objective optimization problem.
- Convergence theory available for the proposed algorithm.

Some theoretical considerations

- From spread stage we obtain new points.
- The spread stage performs a finite number of iterations. (No asymptotics here!)
- The refining stage drives all the available nondominated points to Pareto criticality,
- by obtaining a new point that decreases or maintains all the objective function values.
- (Local) Pareto criticality can be verified based on the refining single-objective optimization problem.
- Convergence theory available for the proposed algorithm.

Some theoretical considerations

- From spread stage we obtain new points.
- The spread stage performs a finite number of iterations. (No asymptotics here!)
- The refining stage drives all the available nondominated points to Pareto criticality,
- by obtaining a new point that decreases or maintains all the objective function values.
- (Local) Pareto criticality can be verified based on the refining single-objective optimization problem.
- Convergence theory available for the proposed algorithm.

Outline

Implementation

- Implemented in MATLAB (fast prototyping, high performance)
- (Single-objective) subproblems are solved by `quadprog` and `fmincon` MATLAB solvers
- Maximum of 20 iterations on the spread stage
- We consider three possibilities for the H_i matrix:
- Two (list) initialization strategies are implemented:

Implementation

- Implemented in MATLAB (fast prototyping, high performance)
- (Single-objective) subproblems are solved by `quadprog` and `fmincon` MATLAB solvers
- Maximum of 20 iterations on the spread stage
- We consider three possibilities for the H_i matrix:
  - $H_i = I_n$ in both stages
  - $H_i = (\nabla^2 f_i(x_0) + I_n)$ in both stages (p.d. matrix)
  - $H_i = I_n$ in the spread stage and $H_i = (\nabla^2 f_i(x_0) + I_n)$ in the compromise stage
- Two (list) initialization strategies are implemented:

Implementation

- Implemented in MATLAB (fast prototyping, high performance)
- (Single-objective) subproblems are solved by `quadprog` and `fmincon` MATLAB solvers
- Maximum of 20 iterations on the spread stage
- We consider three possibilities for the H_i matrix:
  - $H_i = I_m$ in both stages
  - $H_i = (\nabla^2 f_i(x_0) + I_n)$ in both stages (pd. matrix)
  - $H_i = I_m$ in the spread stage and $H_i = (\nabla^2 f_i(x_0) + I_n)$ in the compromise stage
- Two (list) initialization strategies are implemented:

Implementation

- Implemented in MATLAB (fast prototyping, high performance)
- (Single-objective) subproblems are solved by `quadprog` and `fmincon` MATLAB solvers
- Maximum of 20 iterations on the spread stage
- We consider three possibilities for the H_i matrix:
  - $H_i = I_m$ in both stages
  - $H_i = (\nabla^2 f_i(x_k) + E_i)$ in both stages (E_i p.d. matrix)
  - $H_i = I_m$ in the spread stage and $H_i = (\nabla^2 f_i(x_k) + E_i)$ in the refining stage
- Two (list) initialization strategies are implemented:

Implementation

- Implemented in MATLAB (fast prototyping, high performance)
- (Single-objective) subproblems are solved by `quadprog` and `fmincon` MATLAB solvers
- Maximum of 20 iterations on the spread stage
- We consider three possibilities for the H_i matrix:
  - $H_i = I_m$ in both stages
  - $H_i = (\nabla^2 f_i(x_k) + E_i)$ in both stages (E_i p.d. matrix)
  - $H_i = I_m$ in the spread stage and $H_i = (\nabla^2 f_i(x_k) + E_i)$ in the refining stage
- Two (list) initialization strategies are implemented:

Implementation

- Implemented in MATLAB (fast prototyping, high performance)
- (Single-objective) subproblems are solved by `quadprog` and `fmincon` MATLAB solvers
- Maximum of 20 iterations on the spread stage
- We consider three possibilities for the H_i matrix:
  - $H_i = I_m$ in both stages
  - $H_i = (\nabla^2 f_i(x_k) + E_i)$ in both stages (E_i p.d. matrix)
  - $H_i = I_m$ in the spread stage and $H_i = (\nabla^2 f_i(x_k) + E_i)$ in the refining stage
- Two (list) initialization strategies are implemented:
  - line – a line between ℓ and u
  - rand – a uniform (ℓ, u) random distribution

Implementation

- Implemented in MATLAB (fast prototyping, high performance)
- (Single-objective) subproblems are solved by `quadprog` and `fmincon` MATLAB solvers
- Maximum of 20 iterations on the spread stage
- We consider three possibilities for the H_i matrix:
  - $H_i = I_m$ in both stages
  - $H_i = (\nabla^2 f_i(x_k) + E_i)$ in both stages (E_i p.d. matrix)
  - $H_i = I_m$ in the spread stage and $H_i = (\nabla^2 f_i(x_k) + E_i)$ in the refining stage
- Two (list) initialization strategies are implemented:
  - `Line` = a line between ℓ and u ($x_i = \ell + i\frac{u-\ell}{2n_{\mathcal{L}}}$, $i = 1, \ldots, 2n_{\mathcal{L}}$)
  - `Quad` = 2 quadrants (ℓ and u) + random distribution

Implementation

- Implemented in MATLAB (fast prototyping, high performance)
- (Single-objective) subproblems are solved by `quadprog` and `fmincon` MATLAB solvers
- Maximum of 20 iterations on the spread stage
- We consider three possibilities for the H_i matrix:
  - $H_i = I_m$ in both stages
  - $H_i = (\nabla^2 f_i(x_k) + E_i)$ in both stages (E_i p.d. matrix)
  - $H_i = I_m$ in the spread stage and $H_i = (\nabla^2 f_i(x_k) + E_i)$ in the refining stage
- Two (list) initialization strategies are implemented:
  - `line` – a line between ℓ and u ($x_i = \ell + i\frac{u-\ell}{2n_{\mathcal{S}}}$, $i = 1, \ldots, 2n_{\mathcal{S}}$).
  - `rand` – a uniform (ℓ, u) random distribution.

Implementation

- Implemented in MATLAB (fast prototyping, high performance)
- (Single-objective) subproblems are solved by `quadprog` and `fmincon` MATLAB solvers
- Maximum of 20 iterations on the spread stage
- We consider three possibilities for the H_i matrix:
  - $H_i = I_m$ in both stages
  - $H_i = (\nabla^2 f_i(x_k) + E_i)$ in both stages (E_i p.d. matrix)
  - $H_i = I_m$ in the spread stage and $H_i = (\nabla^2 f_i(x_k) + E_i)$ in the refining stage
- Two (list) initialization strategies are implemented:
  - `line` – a line between ℓ and u ($x_i = \ell + i\frac{u-\ell}{2n_{\mathcal{S}}}$, $i = 1, \ldots, 2n\mathcal{S}$).
  - `rand` – a uniform (ℓ, u) random distribution.

Implementation

- Implemented in MATLAB (fast prototyping, high performance)
- (Single-objective) subproblems are solved by `quadprog` and `fmincon` MATLAB solvers
- Maximum of 20 iterations on the spread stage
- We consider three possibilities for the H_i matrix:
  - $H_i = I_m$ in both stages
  - $H_i = (\nabla^2 f_i(x_k) + E_i)$ in both stages (E_i p.d. matrix)
  - $H_i = I_m$ in the spread stage and $H_i = (\nabla^2 f_i(x_k) + E_i)$ in the refining stage
- Two (list) initialization strategies are implemented:
  - `line` – a line between ℓ and u ($x_i = \ell + i\frac{u-\ell}{2n_{\mathcal{S}}}$, $i = 1, \ldots, 2n\mathcal{S}$).
  - `rand` – a uniform (ℓ, u) random distribution.

Outline

Academic test set

- Problems from the academic literature.
- Problems have been coded in AMPL (and a MATLAB-AMPL interface was used). Exact derivatives are provided by AMPL.
- 67 bound constrained test problems (50 problems with $m = 2$, 17 problems with $m = 3$), n varying between 2 and 30.
- 21 constrained test problems (12 problems with $m = 2$, 9 problems with $m = 3$), 7 with nonlinear constraints, 9 with linear constraints, and 5 with both, n varying between 2 and 20.

Academic test set

- Problems from the academic literature.
- Problems have been coded in AMPL (and a MATLAB-AMPL interface was used). Exact derivatives are provided by AMPL.
- 67 bound constrained test problems (50 problems with $m = 2$, 17 problems with $m = 3$), n varying between 2 and 30.
- 21 constrained test problems (12 problems with $m = 2$, 9 problems with $m = 3$), 7 with nonlinear constraints, 9 with linear constraints, and 5 with both, n varying between 2 and 20.

Academic test set

- Problems from the academic literature.
- Problems have been coded in AMPL (and a MATLAB-AMPL interface was used). Exact derivatives are provided by AMPL.
- 67 bound constrained test problems (50 problems with $m = 2$, 17 problems with $m = 3$), n varying between 2 and 30.
- 21 constrained test problems (12 problems with $m = 2$, 9 problems with $m = 3$), 7 with nonlinear constraints, 9 with linear constraints, and 5 with both, n varying between 2 and 20.

Academic test set

- Problems from the academic literature.
- Problems have been coded in AMPL (and a MATLAB-AMPL interface was used). Exact derivatives are provided by AMPL.
- 67 bound constrained test problems (50 problems with $m = 2$, 17 problems with $m = 3$), n varying between 2 and 30.
- 21 constrained test problems (12 problems with $m = 2$, 9 problems with $m = 3$), 7 with nonlinear constraints, 9 with linear constraints, and 5 with both, n varying between 2 and 20.

Solvers

- We consider six implementations of the MOSQP solver: MOSQP $(H = I,$ line), MOSQP $(H = \nabla^2 f,$ line), MOSQP $(H = (I, \nabla^2 f),$ line), MOSQP $(H = I,$ rand), MOSQP $(H = \nabla^2 f,$ rand), MOSQP $(H = (I, \nabla^2 f),$ rand).
- MOSQP compared against NSGA-II (C version) and MOScalar (weighted-sum, equidistant weights).
- We report numerical results via performance and data profiles.
  - Performance profiles: *Purity*, *Spread-Gamma* Metric, *Spread-Delta* Metric, and *Hypervolume* metric. (Small values $\leftrightarrow$ better performance.)
  - Data profiles: how likely is an algorithm to solve a problem, given a computational budget.

Solvers

- We consider six implementations of the MOSQP solver: MOSQP ($H = I$, line), MOSQP ($H = \nabla^2 f$, line), MOSQP ($H = (I, \nabla^2 f)$, line), MOSQP ($H = I$, rand), MOSQP ($H = \nabla^2 f$, rand), MOSQP ($H = (I, \nabla^2 f)$, rand).

- MOSQP compared against NSGA-II (C version) and MOScalar (weighted-sum, equidistant weights).

- We report numerical results via performance and data profiles.
  - Performance profiles: *Purity*, *Spread-Gamma Metric*, *Spread-Delta Metric*, and *Hypervolume* metric. (Small values $\Leftrightarrow$ better performance.)
  - Data profiles: how likely is an algorithm to achieve a certain level of computational budget?

Solvers

- We consider six implementations of the MOSQP solver: MOSQP ($H = I$, line), MOSQP ($H = \nabla^2 f$, line), MOSQP ($H = (I, \nabla^2 f)$, line), MOSQP ($H = I$, rand), MOSQP ($H = \nabla^2 f$, rand), MOSQP ($H = (I, \nabla^2 f)$, rand).

- MOSQP compared against NSGA-II (C version) and MOScalar (weighted-sum, equidistant weights).

- We report numerical results via performance and data profiles.

  - Performance profiles: *Purity*, *Spread*-Gamma Metric, *Spread*-Delta Metric, and *Hypervolume* metric. (Small values $\leftrightarrow$ better performance.)

  - Data profiles: how likely is an algorithm to solve a problem, given a computational budget.

Solvers

- We consider six implementations of the MOSQP solver: MOSQP ($H = I$, line), MOSQP ($H = \nabla^2 f$, line), MOSQP ($H = (I, \nabla^2 f)$, line), MOSQP ($H = I$, rand), MOSQP ($H = \nabla^2 f$, rand), MOSQP ($H = (I, \nabla^2 f)$, rand).

- MOSQP compared against NSGA-II (C version) and MOScalar (weighted-sum, equidistant weights).

- We report numerical results via performance and data profiles.

  - Performance profiles: *Purity*, *Spread*-Gamma Metric, *Spread*-Delta Metric, and *Hypervolume* metric. (Small values $\leftrightarrow$ better performance.)

  - Data profiles: how likely is an algorithm to solve a problem, given a computational budget.

Solvers

- We consider six implementations of the MOSQP solver: MOSQP ($H = I$, line), MOSQP ($H = \nabla^2 f$, line), MOSQP ($H = (I, \nabla^2 f)$, line), MOSQP ($H = I$, rand), MOSQP ($H = \nabla^2 f$, rand), MOSQP ($H = (I, \nabla^2 f)$, rand).

- MOSQP compared against NSGA-II (C version) and MOScalar (weighted-sum, equidistant weights).

- We report numerical results via performance and data profiles.

  - Performance profiles: *Purity*, *Spread*-Gamma Metric, *Spread*-Delta Metric, and *Hypervolume* metric. (Small values $\leftrightarrow$ better performance.)

  - Data profiles: how likely is an algorithm to solve a problem, given a computational budget.

Performance metrics

- *Spread*-Gamma Metric
  For given solver and MOO problem, measures the largest gap in the approximate Pareto front.
- *Spread*-Delta Metric
  For given solver and MOO problem, measures the uniformity of gaps in the approximate Pareto front.
- *Hypervolume*
  For given solver and MOO problem, measures the volume of the space enclosed by the nondominated points and utopia point.
- *Purity*
  For a given solver, a set of solvers, and a given MOO problem, measures 1 / (percentage of points computed that are not dominated by points from other solvers).

Performance metrics

- *Spread*-Gamma Metric
  For given solver and MOO problem, measures the largest gap in the approximate Pareto front.
- *Spread*-Delta Metric
  For given solver and MOO problem, measures the uniformity of gaps in the approximate Pareto front.
- *Hypervolume*
  For given solver and MOO problem, measures the volume of the space enclosed by the nondominated points and utopia point.
- *Purity*
  For a given solver, a set of solvers, and a given MOO problem, measures 1 / (percentage of points computed that are not dominated by points from other solvers).

Performance metrics

- *Spread*-Gamma Metric
  For given solver and MOO problem, measures the largest gap in the approximate Pareto front.
- *Spread*-Delta Metric
  For given solver and MOO problem, measures the uniformity of gaps in the approximate Pareto front.
- *Hypervolume*
  For given solver and MOO problem, measures the volume of the space enclosed by the nondominated points and utopia point.
- *Purity*
  For a given solver, a set of solvers, and a given MOO problem, measures 1 / (percentage of points computed that are not dominated by points from other solvers).

Performance metrics

- *Spread*-Gamma Metric
  For given solver and MOO problem, measures the largest gap in the approximate Pareto front.
- *Spread*-Delta Metric
  For given solver and MOO problem, measures the uniformity of gaps in the approximate Pareto front.
- *Hypervolume*
  For given solver and MOO problem, measures the volume of the space enclosed by the nondominated points and utopia point.
- *Purity*
  For a given solver, a set of solvers, and a given MOO problem, measures 1 / (percentage of points computed that are not dominated by points from other solvers).

Performance profiles

- Performance profiles.
  Represent in one figure, for each solver s, the cumulative distribution function ρ_s for a given performance metric:
  let $\mathcal{P}$ bet the set of problems considered and $r_{p,s}$ performance metric value of solver s on problem p. Then,

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

- Solvers with larger $\lim_{\tau \to \infty} \rho_s(\tau)$ are more robust. If $\rho_s(\tau) = 1$ then solver s solves all given problems.

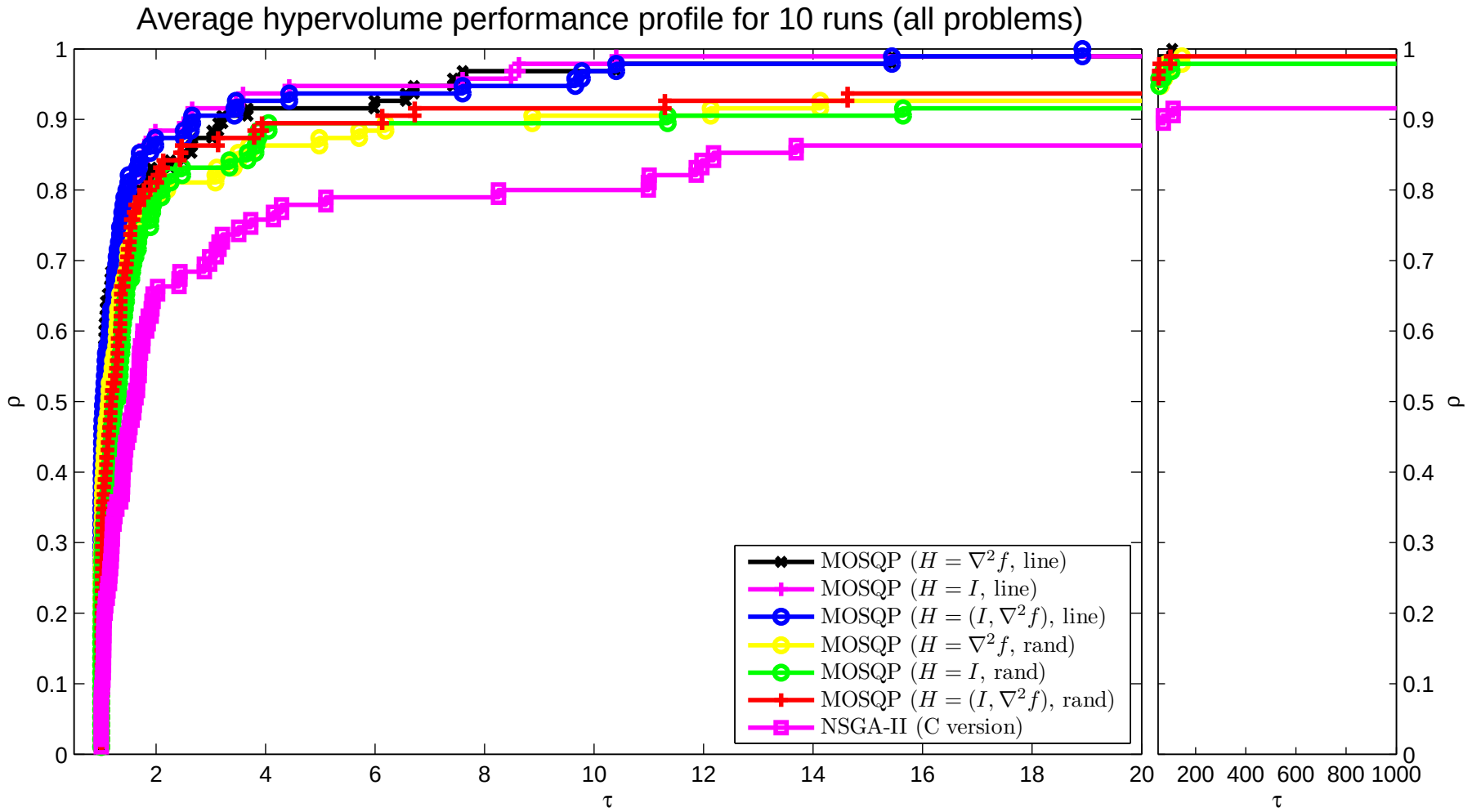
Performance profiles

- Performance profiles.
  Represent in one figure, for each solver s, the cumulative distribution function ρ_s for a given performance metric:
  let $\mathcal{P}$ bet the set of problems considered and $r_{p,s}$ performance metric value of solver s on problem p. Then,

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

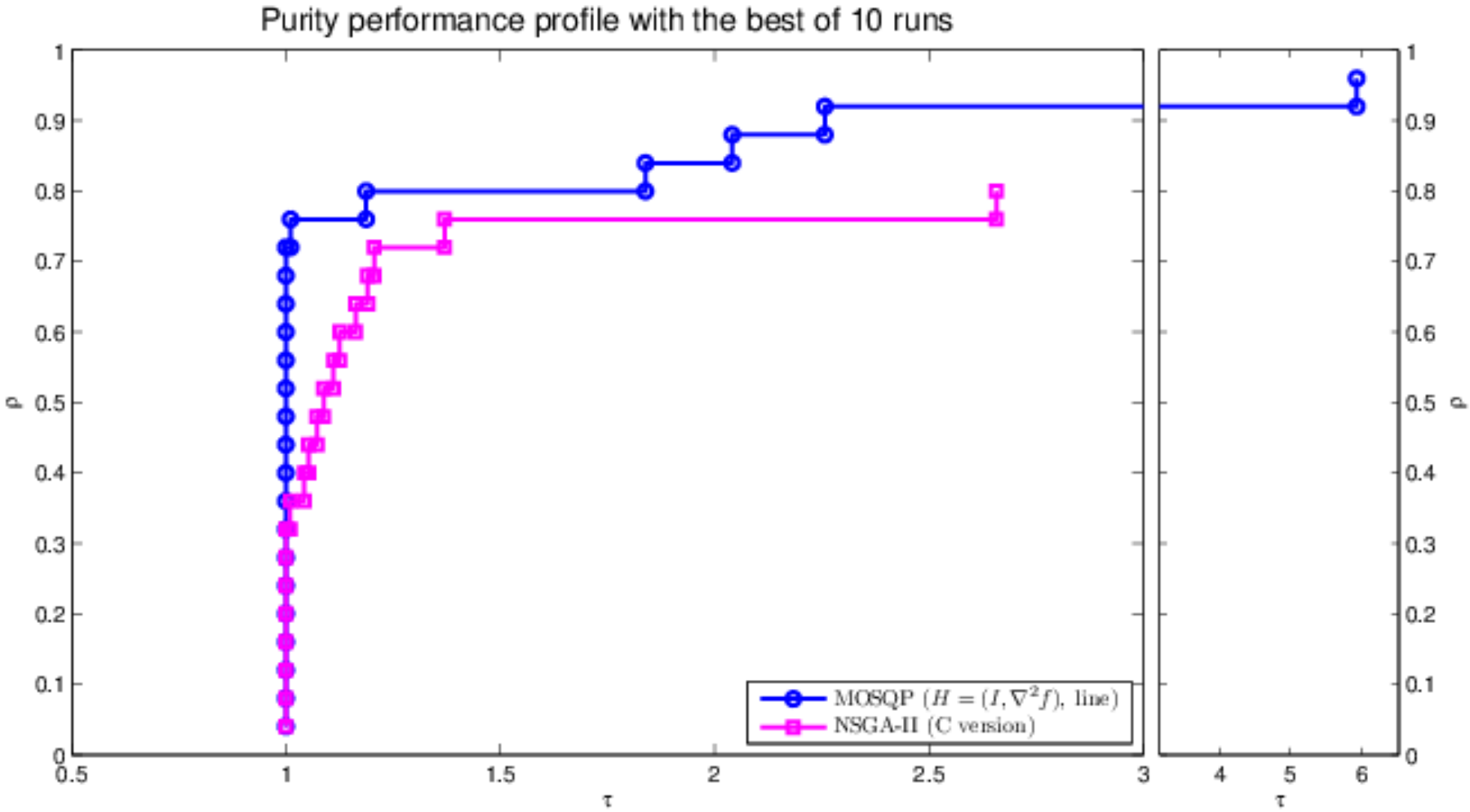
- Solvers with larger $\lim_{\tau \to \infty} \rho_s(\tau)$ are more robust. If $\rho_s(\tau) = 1$ then solver s solves all given problems.

Constrained test set (Hypervolume)



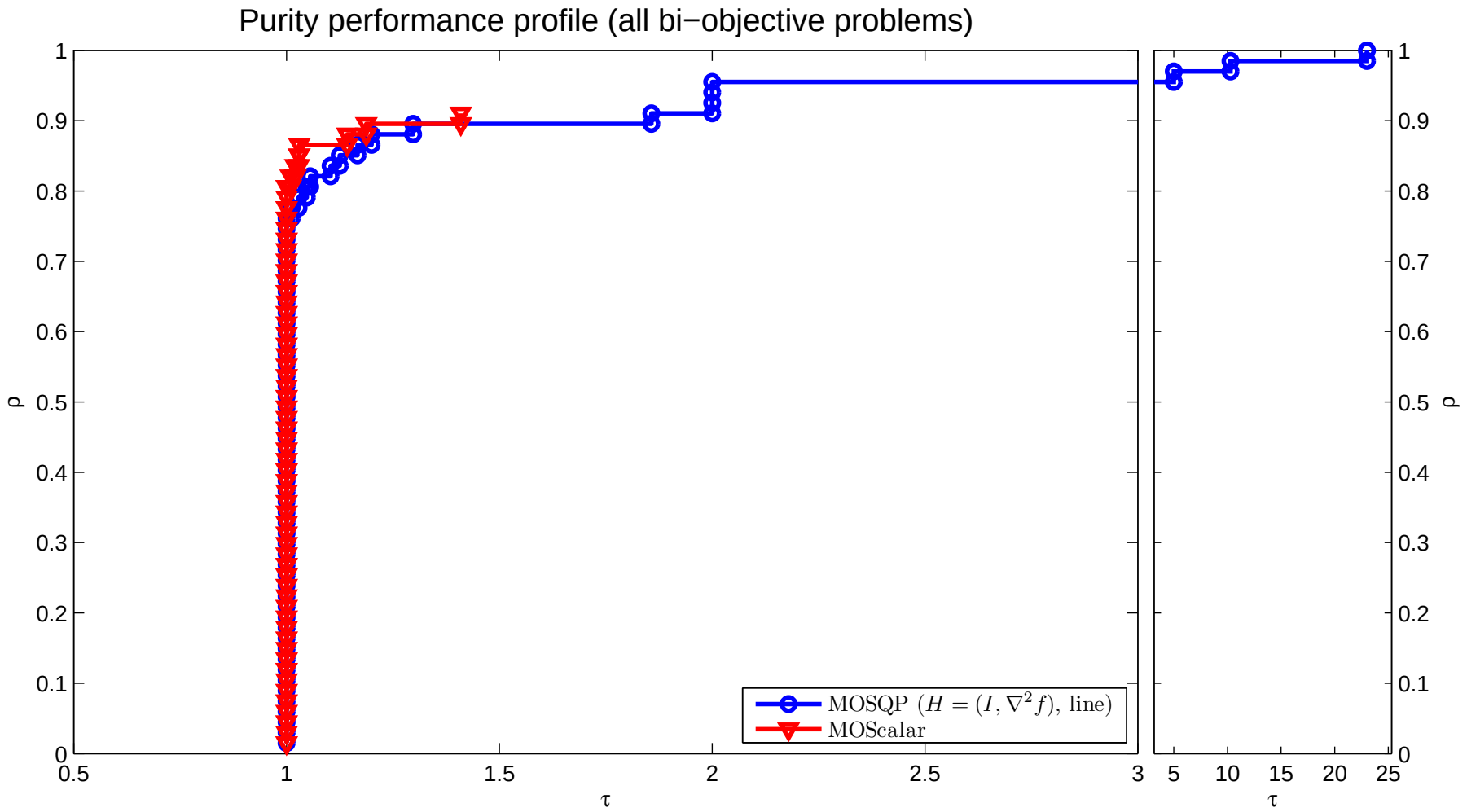
Hypervolume Metric

Constrained test set (Purity)



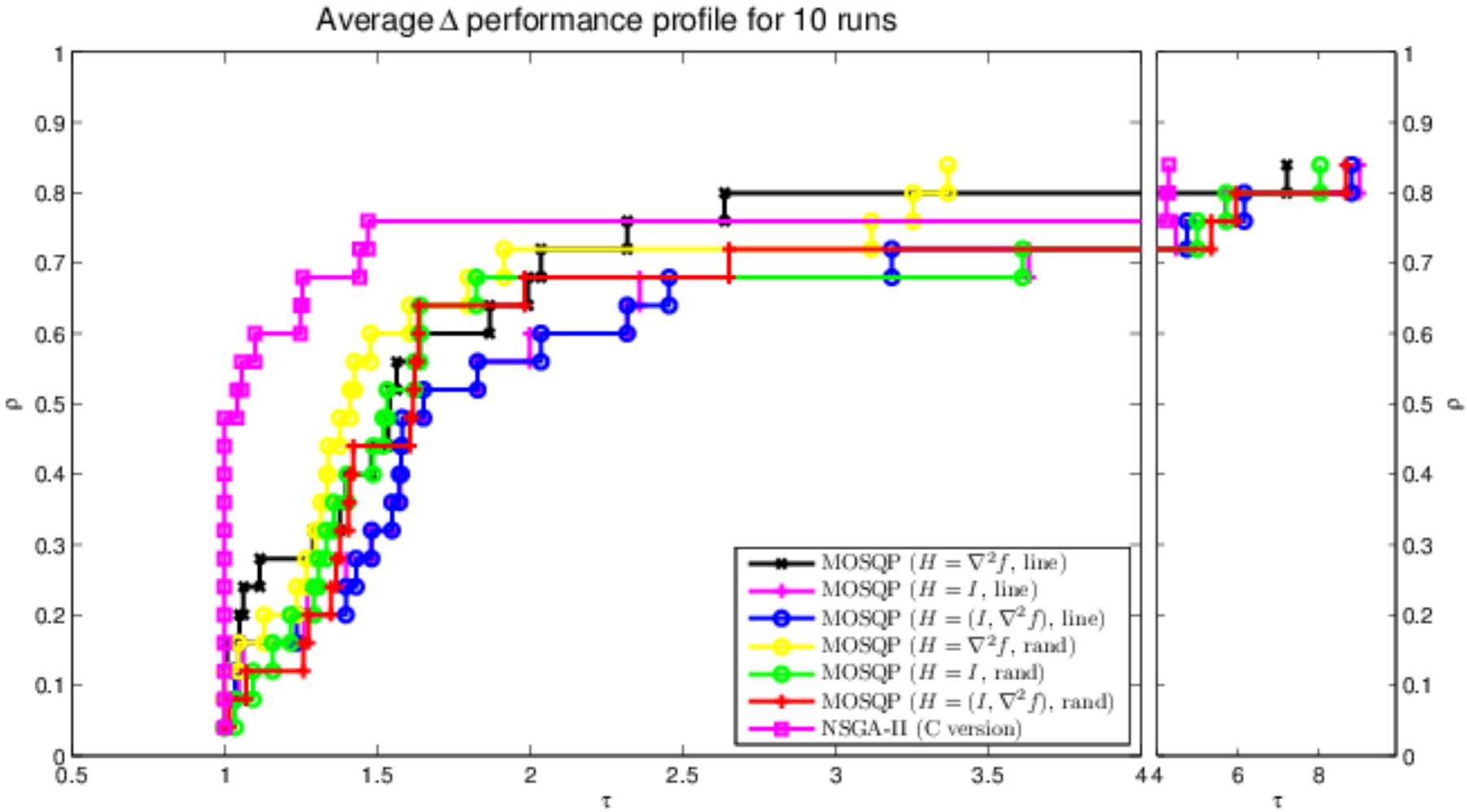
Purity Metric

Constrained test set (Purity)



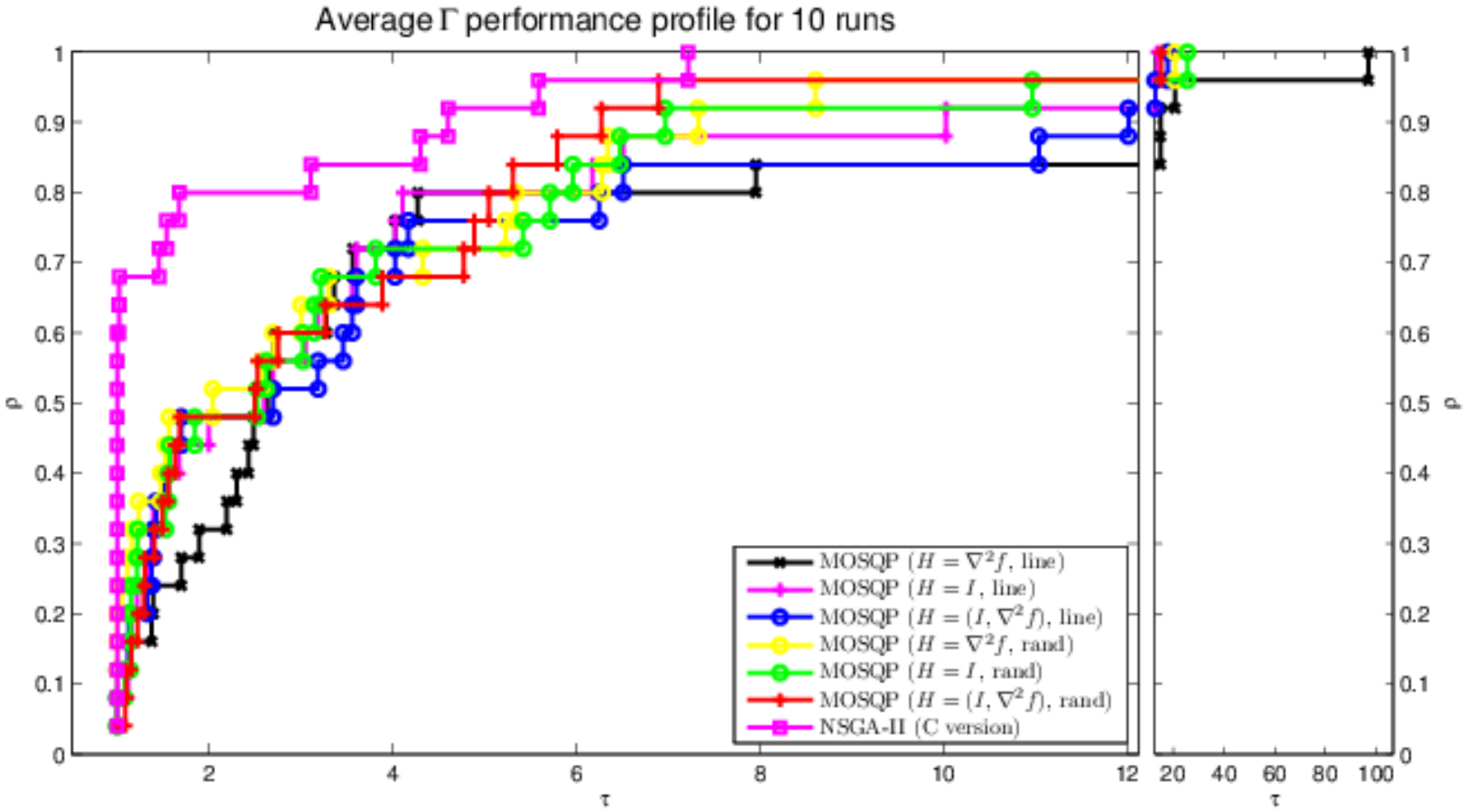
Purity Metric

Constrained test set (Spread)



Delta Metric (uniformity of gaps in the Pareto front)

Constrained test set (Spread)



Gamma Metric (largest gap in the Pareto front)

Data profiles

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

Let $N_{p,s}$ be the number of function and gradient evaluations required for solver s to solve problem p:

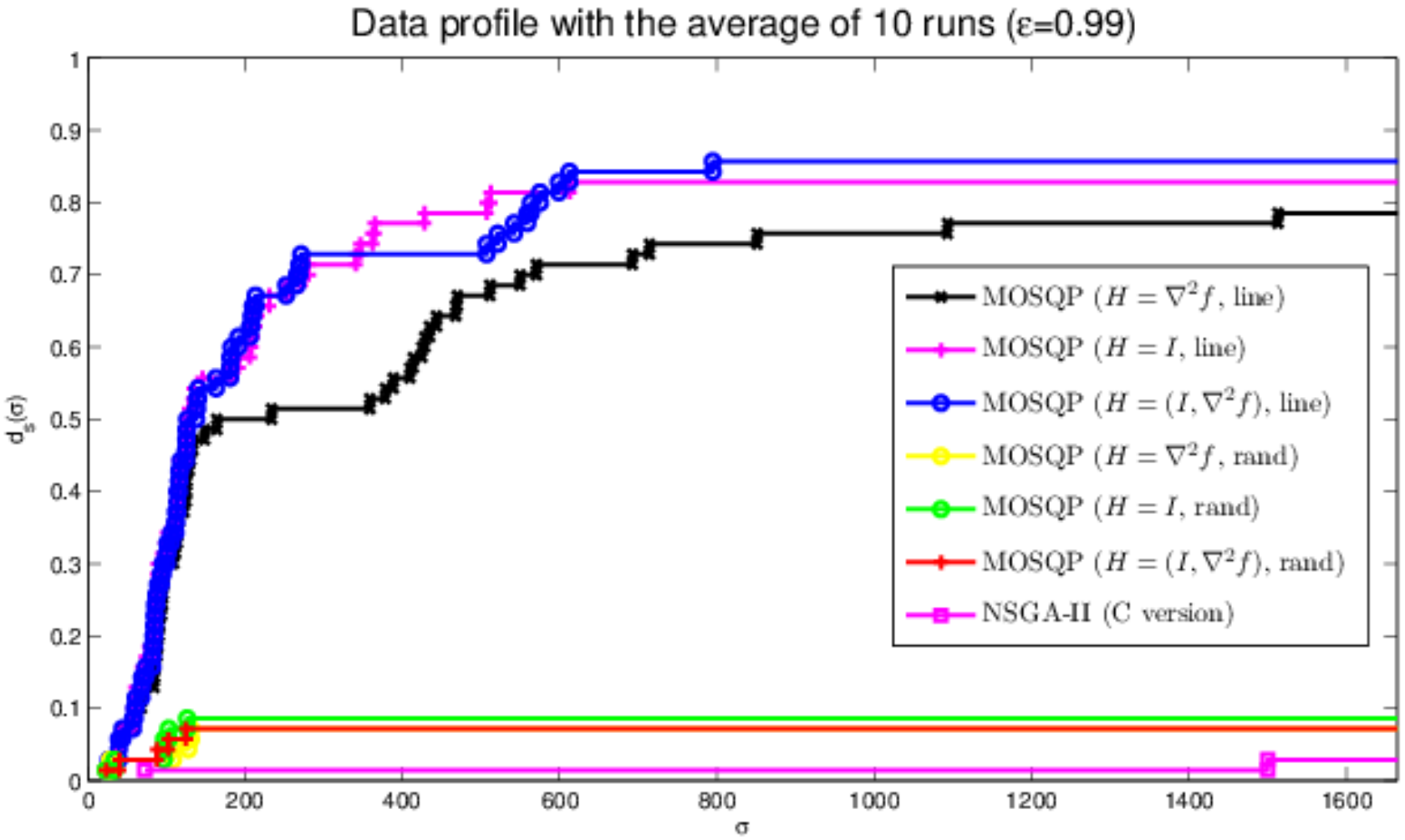
$$N_{p,s,f} := \sum_i \left(\#f_i + n\#\nabla f_i + \frac{(n+1)(n+2)}{2}\#\nabla^2 f_i \right),$$

$$N_{p,s} := N_{p,s,f} + N_{p,s,g} + N_{p,s,h}.$$

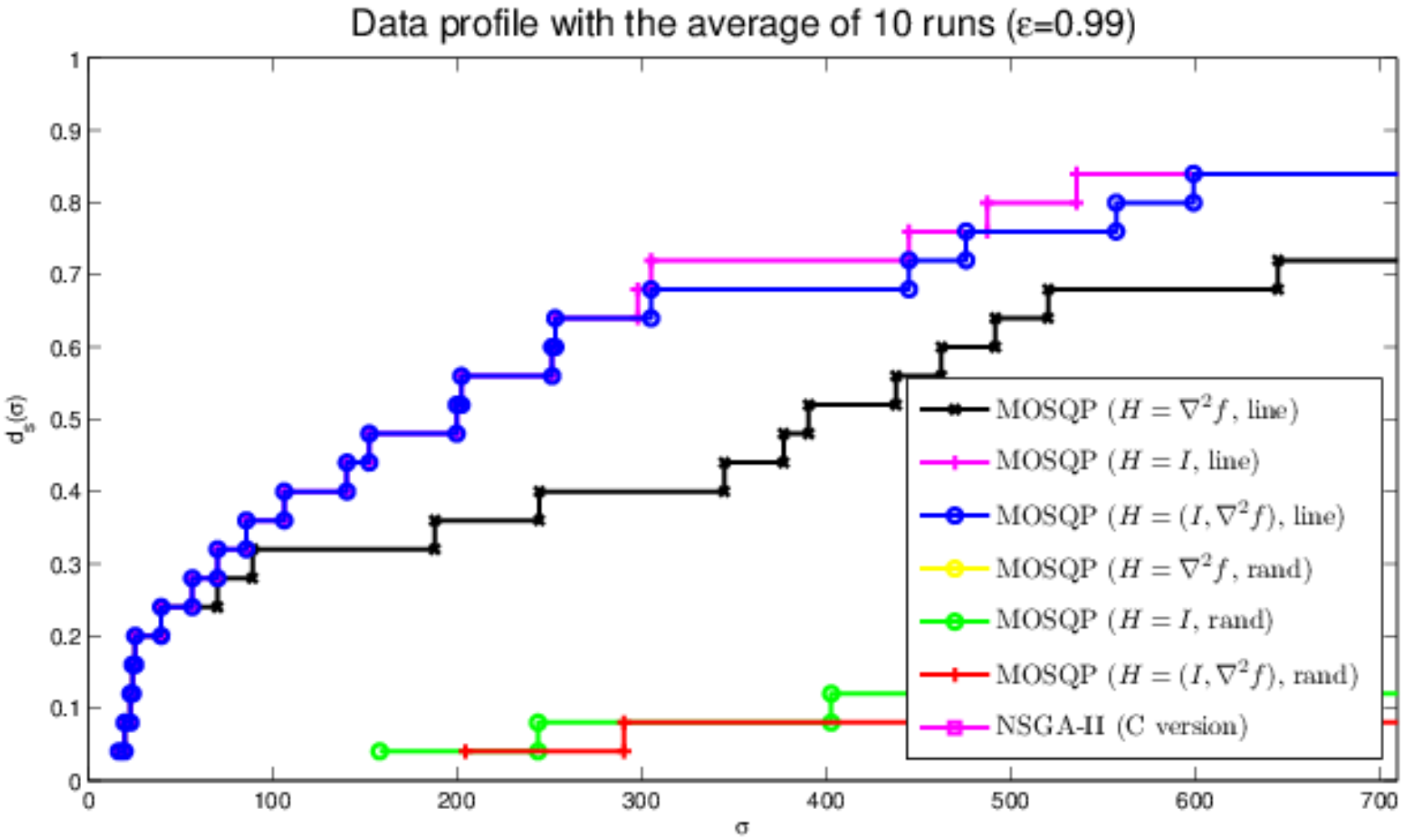
Consider the metric

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

Bound constrained test set

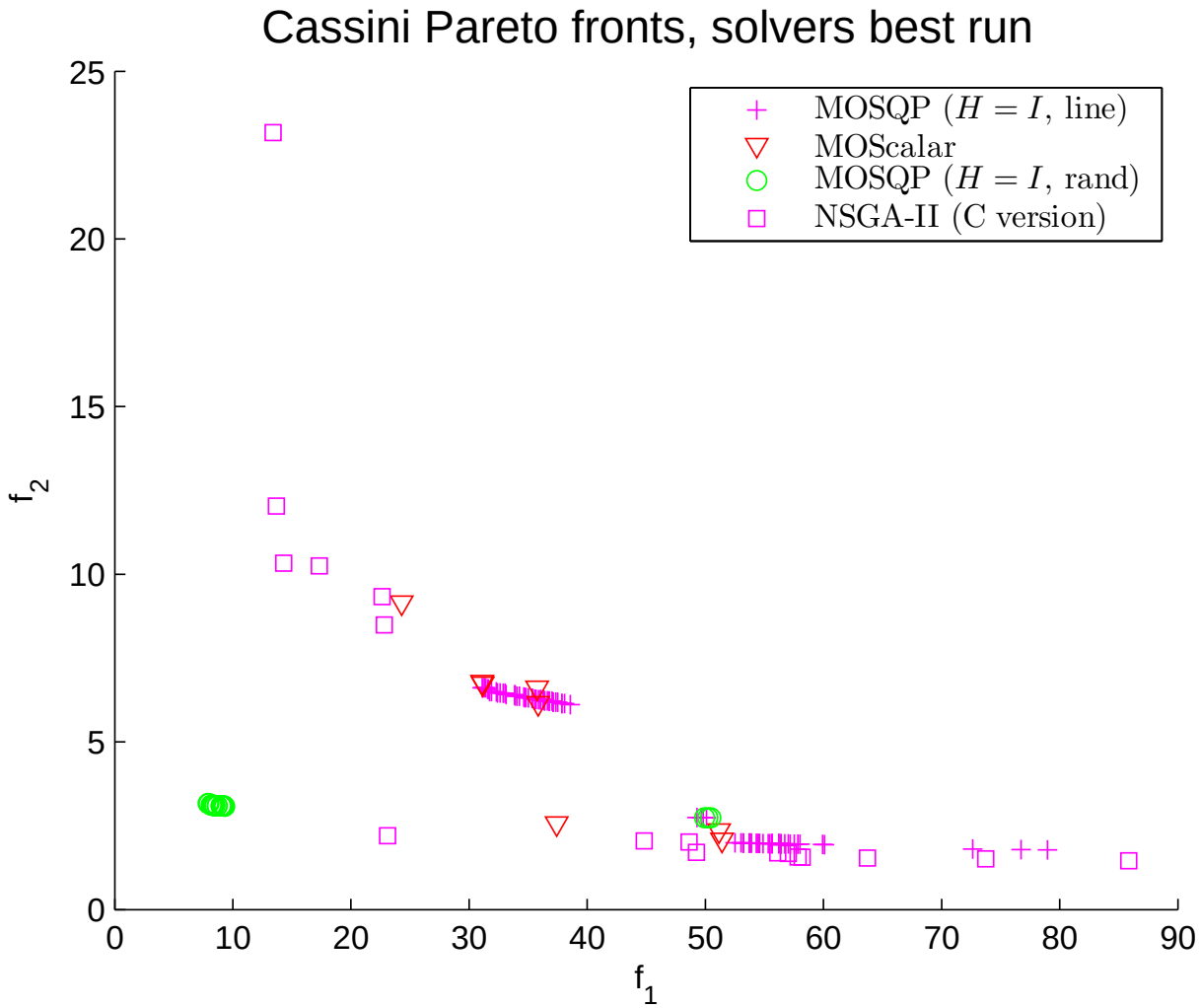


Constrained test set



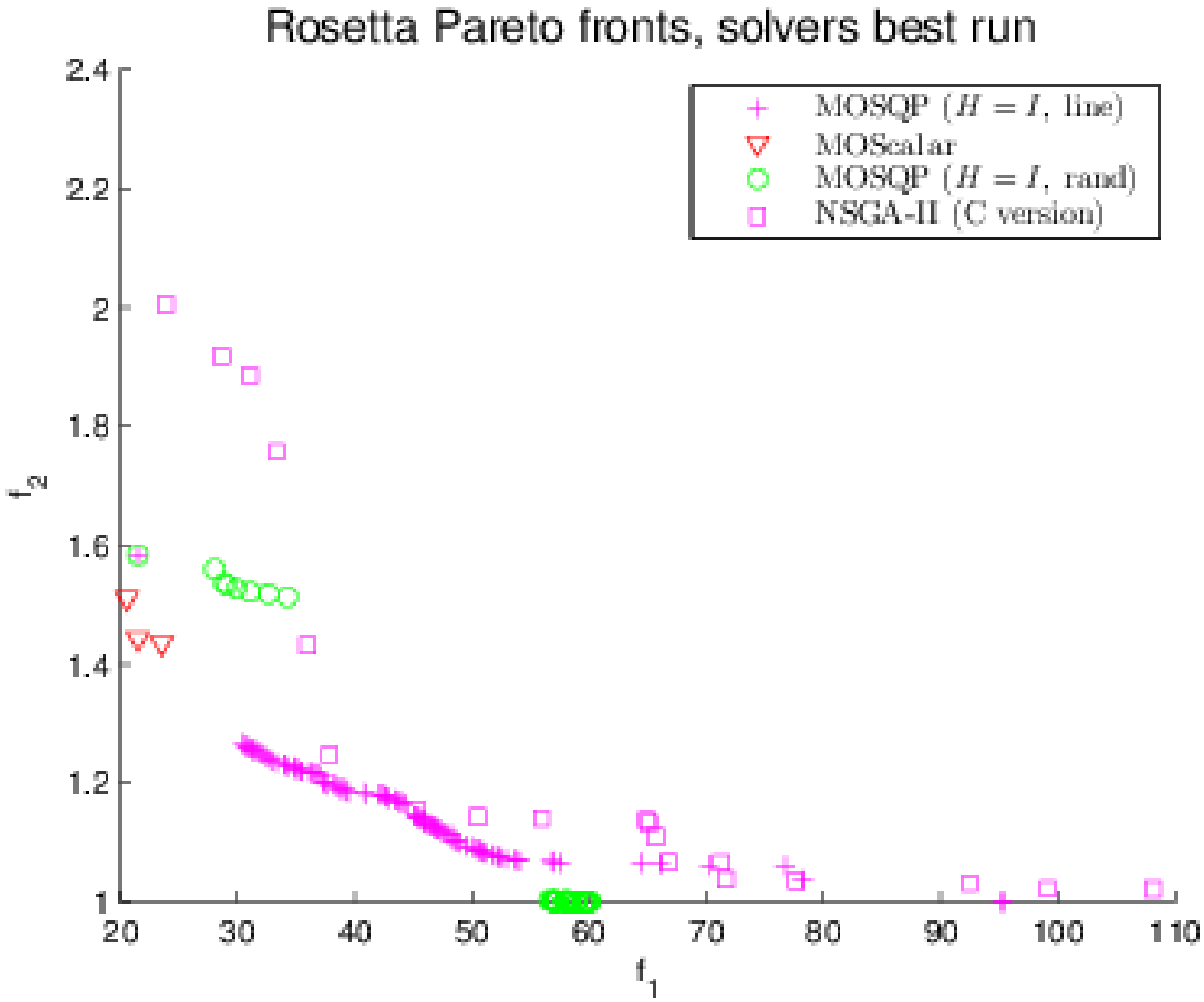
Space Engineering: Earth-Jupiter Mission

$f_1 = \Delta V \sim$ fuel use; and $f_2 =$ total travel time.



Space Engineering: Rosetta bi-objective problem

$f_1 = \Delta$V$\sim$ fuel use; and $f_2 =$ total travel time.



Outline

Conclusions

- We propose a method for constrained multi-objective optimization based on SQP, (MOSQP).
- Convergence proof establishes fast local convergence to Pareto front.
- Implementation of the proposed algorithm in MATLAB.
- Numerical results confirm the solver competitiveness and robustness.

Conclusions

- We propose a method for constrained multi-objective optimization based on SQP, (MOSQP).
- Convergence proof establishes fast local convergence to Pareto front.
- Implementation of the proposed algorithm in MATLAB.
- Numerical results confirm the solver competitiveness and robustness.

Conclusions

- We propose a method for constrained multi-objective optimization based on SQP, (MOSQP).
- Convergence proof establishes fast local convergence to Pareto front.
- Implementation of the proposed algorithm in MATLAB.
- Numerical results confirm the solver competitiveness and robustness.

Conclusions

- We propose a method for constrained multi-objective optimization based on SQP, (MOSQP).
- Convergence proof establishes fast local convergence to Pareto front.
- Implementation of the proposed algorithm in MATLAB.
- Numerical results confirm the solver competitiveness and robustness.

Literature, Data, Code

- All code, test problems, and results available at `http://www.norg.uminho.pt/aivaz/MOSQP/`

- J. Fliege and A. I. F. Vaz: A SQP type method for constrained multiobjective optimization. *SIAM Journal on Optimization*, forthcoming. Available on `optimization-online.org`

- J. Fliege, L. M. Grana Drummond, and B. F. Svaiter: Newton's Method for Multiobjective Optimization. *SIAM Journal Optimization*, 20(2), 602626.

- A. L. Custodio, J. F. A. Madeira, A. I. F. Vaz, and L. N. Vicente: Direct Multisearch for Multiobjective Optimization. *SIAM Journal of Optimization*, 21(3), 11091140.

Literature, Data, Code

- All code, test problems, and results available at `http://www.norg.uminho.pt/aivaz/MOSQP/`

- J. Fliege and A. I. F. Vaz: A SQP type method for constrained multiobjective optimization. *SIAM Journal on Optimization*, forthcoming. Available on `optimization-online.org`

- J. Fliege, L. M. Grana Drummond, and B. F. Svaiter: Newton's Method for Multiobjective Optimization. *SIAM Journal Optimization*, 20(2), 602626.

- A. L. Custodio, J. F. A. Madeira, A. I. F. Vaz, and L. N. Vicente: Direct Multisearch for Multiobjective Optimization. *SIAM Journal of Optimization*, 21(3), 11091140.

Literature, Data, Code

- All code, test problems, and results available at `http://www.norg.uminho.pt/aivaz/MOSQP/`

- J. Fliege and A. I. F. Vaz: A SQP type method for constrained multiobjective optimization. *SIAM Journal on Optimization*, forthcoming. Available on `optimization-online.org`

- J. Fliege, L. M. Grana Drummond, and B. F. Svaiter: Newton's Method for Multiobjective Optimization. *SIAM Journal Optimization*, 20(2), 602626.

- A. L. Custodio, J. F. A. Madeira, A. I. F. Vaz, and L. N. Vicente: Direct Multisearch for Multiobjective Optimization. *SIAM Journal of Optimization*, 21(3), 11091140.

Literature, Data, Code

- All code, test problems, and results available at `http://www.norg.uminho.pt/aivaz/MOSQP/`

- J. Fliege and A. I. F. Vaz: A SQP type method for constrained multiobjective optimization. *SIAM Journal on Optimization*, forthcoming. Available on `optimization-online.org`

- J. Fliege, L. M. Grana Drummond, and B. F. Svaiter: Newton's Method for Multiobjective Optimization. *SIAM Journal Optimization*, 20(2), 602626.

- A. L. Custodio, J. F. A. Madeira, A. I. F. Vaz, and L. N. Vicente: Direct Multisearch for Multiobjective Optimization. *SIAM Journal of Optimization*, 21(3), 11091140.

Thanks – Support

Thanks!

This work has been supported by

- FCT - Fundao para a Cincia e Tecnologia within the Project Scope: PEst-OE/EEI/UI0319/2014.
- European Space Agency, ESTEC: European Space Research and Technology Centre.

Thanks – Support

Thanks!

This work has been supported by

- FCT - Fundao para a Cincia e Tecnologia within the Project Scope: PEst-OE/EEI/UI0319/2014.
- European Space Agency, ESTEC: European Space Research and Technology Centre.