A Direct search method and an application to Astrophysics

A. Ismael F. Vaz

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

with special thanks to Luís Nunes Vicente and João Fernandes

Universidade Federal do Rio de Janeiro

13 November 2007

Outline

1. Introduction
2. Particle swarm
3. Coordinate search
4. The hybrid algorithm
5. Numerical results with a set of test problems
6. Parameter estimation in Astrophysics

Outline

1. Introduction
2. Particle swarm
3. Coordinate search
4. The hybrid algorithm
5. Numerical results with a set of test problems
6. Parameter estimation in Astrophysics

Outline

1. Introduction
2. Particle swarm
3. Coordinate search
4. The hybrid algorithm
5. Numerical results with a set of test problems
6. Parameter estimation in Astrophysics

Outline

1. Introduction
2. Particle swarm
3. Coordinate search
4. The hybrid algorithm
5. Numerical results with a set of test problems
6. Parameter estimation in Astrophysics

Outline

1. Introduction
2. Particle swarm
3. Coordinate search
4. The hybrid algorithm
5. Numerical results with a set of test problems
6. Parameter estimation in Astrophysics

Outline

1. Introduction
2. Particle swarm
3. Coordinate search
4. The hybrid algorithm
5. Numerical results with a set of test problems
6. Parameter estimation in Astrophysics

Outline

1. Introduction
2. Particle swarm
3. Coordinate search
4. The hybrid algorithm
5. Numerical results with a set of test problems
6. Parameter estimation in Astrophysics

Problem formulation

The problem we are addressing is:

Problem definition

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

$$\text{s.t.} \quad \ell \leq z \leq u,$$

where $\ell \leq z \leq u$ are understood componentwise.

Smoothness

To apply particle swarm or coordinate search, smoothness of the objective function $f(z)$ is not required.

Assumption

For the convergence analysis of coordinate search, and therefore of the hybrid algorithm, some smoothness of the objective function $f(z)$ is imposed.

Problem formulation

The problem we are addressing is:

Problem definition

$$\min_{z\in\mathbb{R}^n} \; f(z)$$
$$\text{s.t.} \quad \ell \;\leq\; z \;\leq u,$$

where $\ell \leq z \leq u$ are understood componentwise.

Smoothness

To apply particle swarm or coordinate search, smoothness of the objective function $f(z)$ is not required.

Assumption

For the convergence analysis of coordinate search, and therefore of the hybrid algorithm, some smoothness of the objective function $f(z)$ is imposed.

Problem formulation

The problem we are addressing is:

Problem definition

$$\min_{z \in \mathbb{R}^n} \; f(z)$$
$$\text{s.t.} \quad \ell \leq z \leq u,$$

where $\ell \leq z \leq u$ are understood componentwise.

Smoothness

To apply particle swarm or coordinate search, smoothness of the objective function $f(z)$ is not required.

Assumption

For the convergence analysis of coordinate search, and therefore of the hybrid algorithm, some smoothness of the objective function $f(z)$ is imposed.

Outline

1. Introduction
2. Particle swarm
3. Coordinate search
4. The hybrid algorithm
5. Numerical results with a set of test problems
6. Parameter estimation in Astrophysics

Particle Swarm paradigm (PS)

- Population based algorithms that try to mimic the social behavior of a population (swarm) of individuals (particles).
- An individual behavior is a combination of its past experience (cognitive influence) and of the society experience (social influence).
- In the optimization context, one particle p, at time instance t, is represented by its current position ($x^p(t)$), its best ever position ($y^p(t)$) and a *traveling* velocity ($v^p(t)$).
- Let $\hat{y}(t)$ represent the best particle position of the population.

Particle Swarm paradigm (PS)

- Population based algorithms that try to mimic the social behavior of a population (swarm) of individuals (particles).
- An individual behavior is a combination of its past experience (cognitive influence) and of the society experience (social influence).
- In the optimization context, one particle p, at time instance t, is represented by its current position ($x^p(t)$), its best ever position ($y^p(t)$) and a *traveling* velocity ($v^p(t)$).
- Let $\hat{y}(t)$ represent the best particle position of the population.

Particle Swarm paradigm (PS)

- Population based algorithms that try to mimic the social behavior of a population (swarm) of individuals (particles).
- An individual behavior is a combination of its past experience (cognitive influence) and of the society experience (social influence).
- In the optimization context, one particle p, at time instance t, is represented by its current position ($x^p(t)$), its best ever position ($y^p(t)$) and a *traveling* velocity ($v^p(t)$).
- Let $\hat{y}(t)$ represent the best particle position of the population.

Particle Swarm paradigm (PS)

- Population based algorithms that try to mimic the social behavior of a population (swarm) of individuals (particles).
- An individual behavior is a combination of its past experience (cognitive influence) and of the society experience (social influence).
- In the optimization context, one particle p, at time instance t, is represented by its current position ($x^p(t)$), its best ever position ($y^p(t)$) and a *traveling* velocity ($v^p(t)$).
- Let $\hat{y}(t)$ represent the best particle position of the population.

New position and velocity

The new particle position is updated by

Update particle

$$x^p(t+1) = x^p(t) + v^p(t+1),$$

where $v^p(t+1)$ is the new velocity given by

Update velocity

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

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

- $\iota(t)$ is the inertial factor
- μ is the *cognitive* parameter and ν is the *social* parameter
- $\omega_{1j}(t)$ and $\omega_{2j}(t)$ are random numbers drawn from the uniform $(0, 1)$ distribution.

New position and velocity

The new particle position is updated by

Update particle

$$x^p(t+1) = x^p(t) + v^p(t+1),$$

where $v^p(t+1)$ is the new velocity given by

Update velocity

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

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

- $\iota(t)$ is the inertial factor
- μ is the *cognitive* parameter and ν is the *social* parameter
- $\omega_{1j}(t)$ and $\omega_{2j}(t)$ are random numbers drawn from the uniform $(0,1)$ distribution.

New position and velocity

The new particle position is updated by

Update particle

$$x^p(t+1) = x^p(t) + v^p(t+1),$$

where $v^p(t+1)$ is the new velocity given by

Update velocity

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

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

- $\iota(t)$ is the inertial factor
- μ is the *cognitive* parameter and ν is the *social* parameter
- $\omega_{1j}(t)$ and $\omega_{2j}(t)$ are random numbers drawn from the uniform $(0,1)$ distribution.

New position and velocity

The new particle position is updated by

Update particle

$$x^p(t+1) = x^p(t) + v^p(t+1),$$

where $v^p(t+1)$ is the new velocity given by

Update velocity

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

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

- $\iota(t)$ is the inertial factor
- μ is the *cognitive* parameter and ν is the *social* parameter
- $\omega_{1j}(t)$ and $\omega_{2j}(t)$ are random numbers drawn from the uniform $(0, 1)$ distribution.

New position and velocity

The new particle position is updated by

Update particle

$$x^p(t+1) = x^p(t) + v^p(t+1),$$

where $v^p(t+1)$ is the new velocity given by

Update velocity

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

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

- $\iota(t)$ is the inertial factor
- μ is the *cognitive* parameter and ν is the *social* parameter
- $\omega_{1j}(t)$ and $\omega_{2j}(t)$ are random numbers drawn from the uniform $(0, 1)$ distribution.

The best ever particle

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

Best position

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

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

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

Best position

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

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.

Handling bound constraints

In particle swarm, simple bound constraints are handled by a projection onto $\Omega = \{x \in \mathbb{R}^n : \ \ell \leq x \leq u\}$, for all particles $i = 1, \ldots, s$.

Projection

$$proj_\Omega(x_j^i(t)) = \begin{cases} \ell_j & \text{if } \ x_j^i(t) < \ell_j, \\ u_j & \text{if } \ x_j^i(t) > u_j, \\ x_j^i(t) & \text{otherwise,} \end{cases}$$

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

The algorithm

1. Given s and $v_{tol} > 0$. Let $\{x^1(0), \ldots, x^s(0)\}$ and $\{v^1(0), \ldots, v^s(0)\}$.
2. $y^i(0) = x^i(0)$, $i = 1, \ldots, s$, and $\hat{y}(0) = \arg\min_{z \in \{y^1(0), \ldots, y^s(0)\}} f(z)$. $t = 0$.
3. $\hat{y}(t+1) = \hat{y}(t)$.
   For $i = 1, \ldots, s$ do:
   - $x^i(t) = proj_\Omega(x^i(t))$.
   - If $f(x^i(t)) < f(y^i(t))$ then
   - Otherwise $y^i(t+1) = y^i(t)$
4. Compute $v^i(t+1)$ e $x^i(t+1)$, $i = 1, \ldots, s$.
5. If $\|v^i(t+1)\| < v_{tol}$, $\forall i = 1, \ldots, s$, then stop. Otherwise set $t = t + 1$ and goto Step 3.

The algorithm

1. Given s and $v_{tol} > 0$. Let $\{x^1(0), \ldots, x^s(0)\}$ and $\{v^1(0), \ldots, v^s(0)\}$.
2. $y^i(0) = x^i(0)$, $i = 1, \ldots, s$, and $\hat{y}(0) = \arg\min_{z \in \{y^1(0), \ldots, y^s(0)\}} f(z)$. $t = 0$.
3. $\hat{y}(t+1) = \hat{y}(t)$.
   For $i = 1, \ldots, s$ do:
   - $\hat{x}^i(t) = proj_{\Omega}(x^i(t))$.
   - If $f(\hat{x}^i(t)) < f(y^i(t))$ then
   - Otherwise $y^i(t+1) = y^i(t)$.
4. Compute $v^i(t+1)$ e $x^i(t+1)$, $i = 1, \ldots, s$.
5. If $\|v^i(t+1)\| < v_{tol}$, $\forall i = 1, \ldots, s$, then stop. Otherwise set $t = t + 1$ and goto Step 3.

The algorithm

1. Given s and $v_{tol} > 0$. Let $\{x^1(0), \ldots, x^s(0)\}$ and $\{v^1(0), \ldots, v^s(0)\}$.
2. $y^i(0) = x^i(0)$, $i = 1, \ldots, s$, and $\hat{y}(0) = \arg\min_{z \in \{y^1(0), \ldots, y^s(0)\}} f(z)$. $t = 0$.
3. $\hat{y}(t+1) = \hat{y}(t)$.
   For $i = 1, \ldots, s$ do:
   - $\hat{x}^i(t) = proj_\Omega(x^i(t))$.
   - If $f(\hat{x}^i(t)) < f(y^i(t))$ then
     - $y^i(t+1) = \hat{x}^i(t)$.
     - If $f(y^i(t+1)) < f(\hat{y}(t+1))$ then
   - Otherwise $y^i(t+1) = y^i(t)$.
4. Compute $v^i(t+1)$ e $x^i(t+1)$, $i = 1, \ldots, s$.
5. If $\|v^i(t+1)\| < v_{tol}$, $\forall i = 1, \ldots, s$, then stop. Otherwise set $t = t + 1$ and goto Step 3.

The algorithm

1. Given s and $v_{tol} > 0$. Let $\{x^1(0), \ldots, x^s(0)\}$ and $\{v^1(0), \ldots, v^s(0)\}$.
2. $y^i(0) = x^i(0)$, $i = 1, \ldots, s$, and $\hat{y}(0) = \arg\min_{z \in \{y^1(0), \ldots, y^s(0)\}} f(z)$. $t = 0$.
3. $\hat{y}(t+1) = \hat{y}(t)$.
   For $i = 1, \ldots, s$ do:
   - $\hat{x}^i(t) = proj_{\Omega}(x^i(t))$.
   - If $f(\hat{x}^i(t)) < f(y^i(t))$ then
     - $y^i(t+1) = \hat{x}^i(t)$.
     - If $f(y^i(t+1)) < f(\hat{y}(t+1))$ then $\hat{y}(t+1) = y^i(t+1)$.
   - Otherwise $y^i(t+1) = y^i(t)$.
4. Compute $v^i(t+1)$ e $x^i(t+1)$, $i = 1, \ldots, s$.
5. If $\|v^i(t+1)\| < v_{tol}$, $\forall i = 1, \ldots, s$, then stop. Otherwise set $t = t + 1$ and goto Step 3.

The algorithm

1. Given s and $v_{tol} > 0$. Let $\{x^1(0), \ldots, x^s(0)\}$ and $\{v^1(0), \ldots, v^s(0)\}$.
2. $y^i(0) = x^i(0)$, $i = 1, \ldots, s$, and $\hat{y}(0) = \arg\min_{z \in \{y^1(0), \ldots, y^s(0)\}} f(z)$. $t = 0$.
3. $\hat{y}(t+1) = \hat{y}(t)$.
   For $i = 1, \ldots, s$ do:
   - $\hat{x}^i(t) = proj_\Omega(x^i(t))$.
   - If $f(\hat{x}^i(t)) < f(y^i(t))$ then
     - $y^i(t+1) = \hat{x}^i(t)$.
     - If $f(y^i(t+1)) < f(\hat{y}(t+1))$ then $\hat{y}(t+1) = y^i(t+1)$.
   - Otherwise $y^i(t+1) = y^i(t)$.
4. Compute $v^i(t+1)$ e $x^i(t+1)$, $i = 1, \ldots, s$.
5. If $\|v^i(t+1)\| < v_{tol}$, $\forall i = 1, \ldots, s$, then stop. Otherwise set $t = t+1$ and goto Step 3.

The algorithm

1. Given s and $v_{tol} > 0$. Let $\{x^1(0), \ldots, x^s(0)\}$ and $\{v^1(0), \ldots, v^s(0)\}$.
2. $y^i(0) = x^i(0)$, $i = 1, \ldots, s$, and $\hat{y}(0) = \arg\min_{z \in \{y^1(0), \ldots, y^s(0)\}} f(z)$. $t = 0$.
3. $\hat{y}(t+1) = \hat{y}(t)$.
   For $i = 1, \ldots, s$ do:
   - $\hat{x}^i(t) = proj_\Omega(x^i(t))$.
   - If $f(\hat{x}^i(t)) < f(y^i(t))$ then
     - $y^i(t+1) = \hat{x}^i(t)$.
     - If $f(y^i(t+1)) < f(\hat{y}(t+1))$ then $\hat{y}(t+1) = y^i(t+1)$.
   - Otherwise $y^i(t+1) = y^i(t)$.
4. Compute $v^i(t+1)$ e $x^i(t+1)$, $i = 1, \ldots, s$.
5. If $\|v^i(t+1)\| < v_{tol}$, $\forall i = 1, \ldots, s$, then stop. Otherwise set $t = t + 1$ and goto Step 3.

The algorithm

1. Given s and $v_{tol} > 0$. Let $\{x^1(0), \ldots, x^s(0)\}$ and $\{v^1(0), \ldots, v^s(0)\}$.
2. $y^i(0) = x^i(0)$, $i = 1, \ldots, s$, and $\hat{y}(0) = \arg\min_{z \in \{y^1(0), \ldots, y^s(0)\}} f(z)$. $t = 0$.
3. $\hat{y}(t+1) = \hat{y}(t)$.
   For $i = 1, \ldots, s$ do:
   - $\hat{x}^i(t) = proj_\Omega(x^i(t))$.
   - If $f(\hat{x}^i(t)) < f(y^i(t))$ then
     - $y^i(t+1) = \hat{x}^i(t)$.
     - If $f(y^i(t+1)) < f(\hat{y}(t+1))$ then $\hat{y}(t+1) = y^i(t+1)$.
   - Otherwise $y^i(t+1) = y^i(t)$.
4. Compute $v^i(t+1)$ e $x^i(t+1)$, $i = 1, \ldots, s$.
5. If $\|v^i(t+1)\| < v_{tol}$, $\forall i = 1, \ldots, s$, then stop. Otherwise set $t = t + 1$ and goto Step 3.

The algorithm

1. Given s and $v_{tol} > 0$. Let $\{x^1(0), \ldots, x^s(0)\}$ and $\{v^1(0), \ldots, v^s(0)\}$.
2. $y^i(0) = x^i(0)$, $i = 1, \ldots, s$, and $\hat{y}(0) = \arg\min_{z \in \{y^1(0), \ldots, y^s(0)\}} f(z)$. $t = 0$.
3. $\hat{y}(t+1) = \hat{y}(t)$.
   For $i = 1, \ldots, s$ do:
   - $\hat{x}^i(t) = proj_{\Omega}(x^i(t))$.
   - If $f(\hat{x}^i(t)) < f(y^i(t))$ then
     - $y^i(t+1) = \hat{x}^i(t)$.
     - If $f(y^i(t+1)) < f(\hat{y}(t+1))$ then $\hat{y}(t+1) = y^i(t+1)$.
   - Otherwise $y^i(t+1) = y^i(t)$.
4. Compute $v^i(t+1)$ e $x^i(t+1)$, $i = 1, \ldots, s$.
5. If $\|v^i(t+1)\| < v_{tol}$, $\forall i = 1, \ldots, s$, then stop. Otherwise set $t = t + 1$ and goto Step 3.

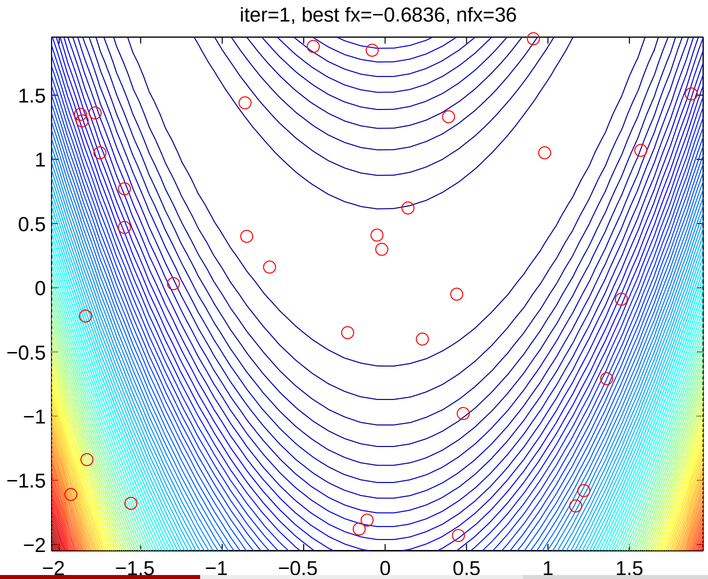
The algorithm

1. Given s and $v_{tol} > 0$. Let $\{x^1(0), \ldots, x^s(0)\}$ and $\{v^1(0), \ldots, v^s(0)\}$.
2. $y^i(0) = x^i(0)$, $i = 1, \ldots, s$, and $\hat{y}(0) = \arg\min_{z \in \{y^1(0), \ldots, y^s(0)\}} f(z)$. $t = 0$.
3. $\hat{y}(t+1) = \hat{y}(t)$.
   For $i = 1, \ldots, s$ do:
   - $\hat{x}^i(t) = proj_\Omega(x^i(t))$.
   - If $f(\hat{x}^i(t)) < f(y^i(t))$ then
     - $y^i(t+1) = \hat{x}^i(t)$.
     - If $f(y^i(t+1)) < f(\hat{y}(t+1))$ then $\hat{y}(t+1) = y^i(t+1)$.
   - Otherwise $y^i(t+1) = y^i(t)$.
4. Compute $v^i(t+1)$ e $x^i(t+1)$, $i = 1, \ldots, s$.
5. If $\|v^i(t+1)\| < v_{tol}$, $\forall i = 1, \ldots, s$, then stop. Otherwise set $t = t + 1$ and goto Step 3.

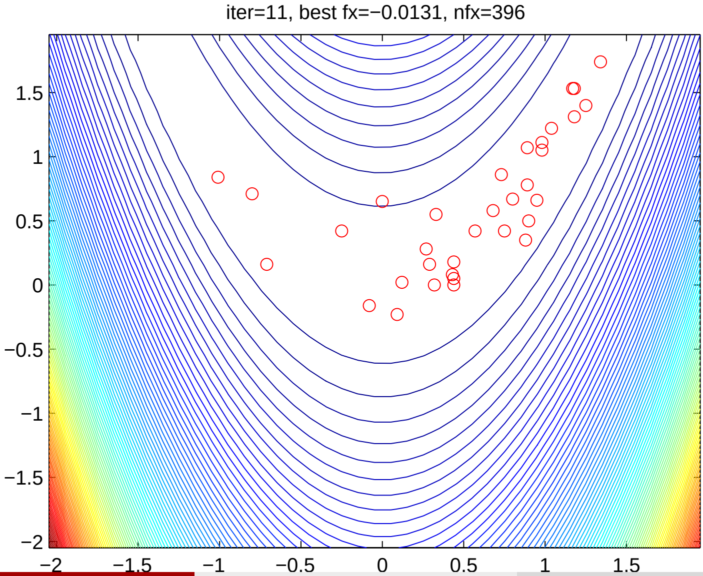
The algorithm

1. Given s and $v_{tol} > 0$. Let $\{x^1(0), \ldots, x^s(0)\}$ and $\{v^1(0), \ldots, v^s(0)\}$.
2. $y^i(0) = x^i(0)$, $i = 1, \ldots, s$, and $\hat{y}(0) = \arg\min_{z \in \{y^1(0), \ldots, y^s(0)\}} f(z)$. $t = 0$.
3. $\hat{y}(t+1) = \hat{y}(t)$.
   For $i = 1, \ldots, s$ do:
   - $\hat{x}^i(t) = proj_\Omega(x^i(t))$.
   - If $f(\hat{x}^i(t)) < f(y^i(t))$ then
     - $y^i(t+1) = \hat{x}^i(t)$.
     - If $f(y^i(t+1)) < f(\hat{y}(t+1))$ then $\hat{y}(t+1) = y^i(t+1)$.
   - Otherwise $y^i(t+1) = y^i(t)$.
4. Compute $v^i(t+1)$ e $x^i(t+1)$, $i = 1, \ldots, s$.
5. If $\|v^i(t+1)\| < v_{tol}$, $\forall i = 1, \ldots, s$, then stop. Otherwise set $t = t+1$ and goto Step 3.

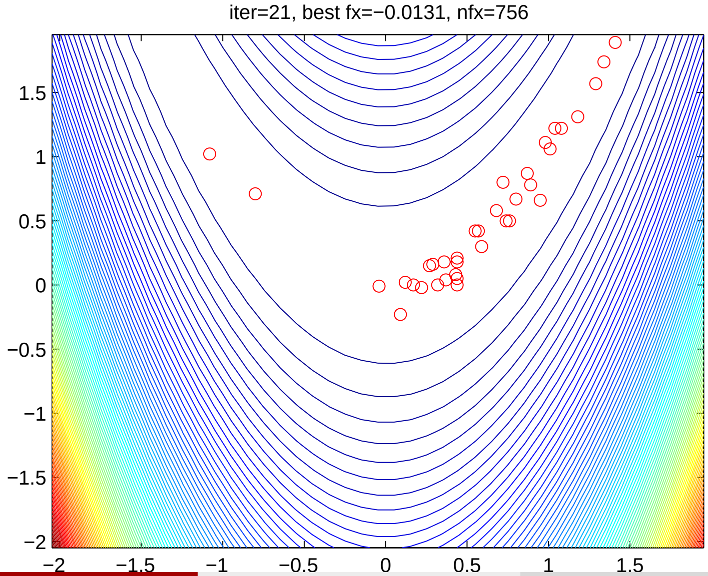
Example



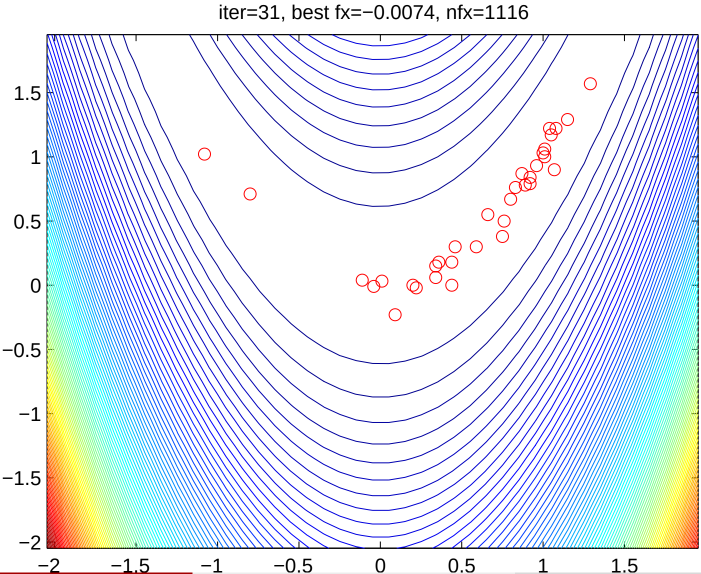
Example



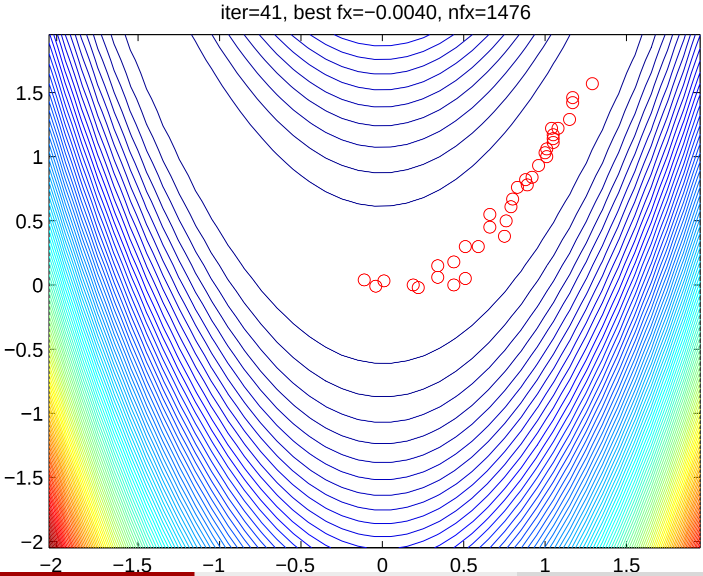
Example



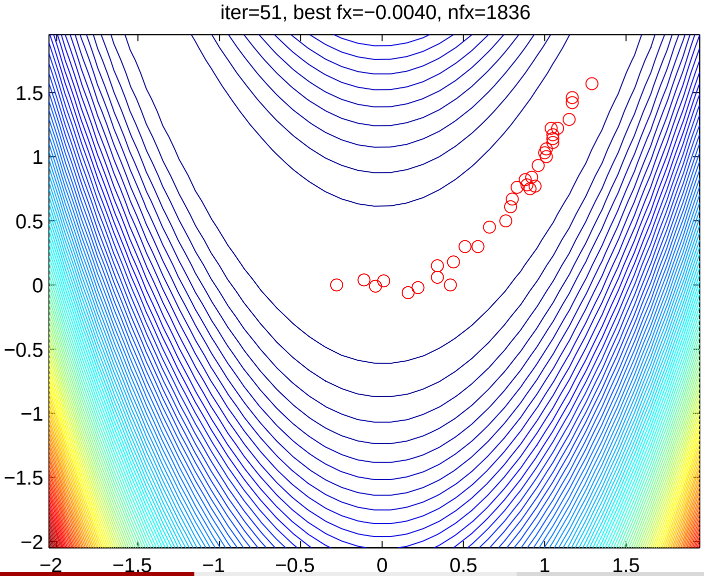
Example



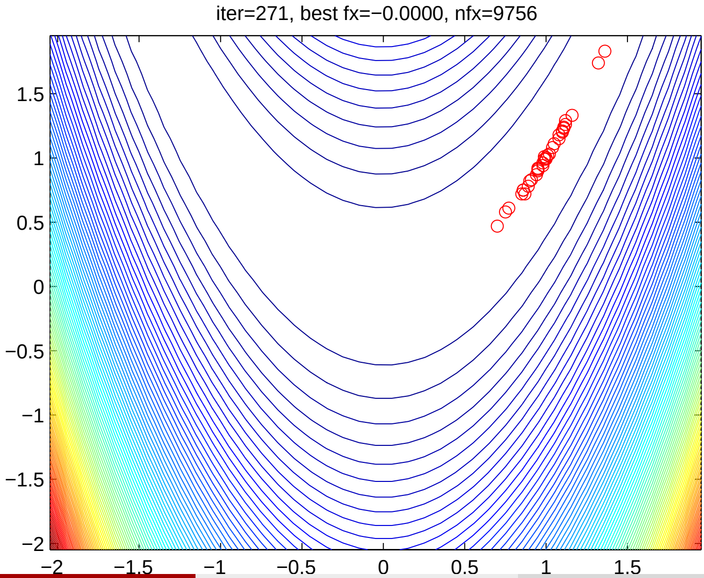
Example



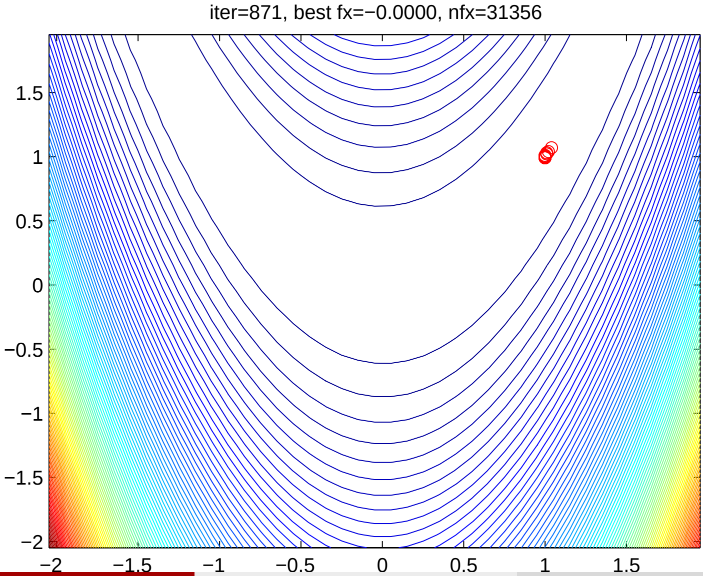
Example



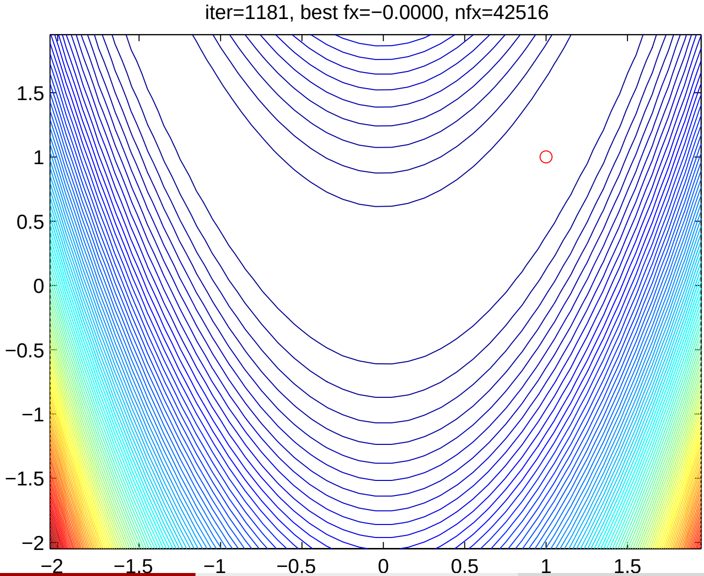
Example



Example


iter=871, best fx=−0.0000, nfx=31356
1.5
1
0.5
0
−0.5
−1
−1.5
−2
−2
−1.5
−1
−0.5
0
0.5
1
1.5

Example


iter=1181, best fx=−0.0000, nfx=42516
1.5
1
0.5
0
−0.5
−1
−1.5
−2
−2
−1.5
−1
−0.5
0
0.5
1
1.5

Some properties

- Easy to implement.
- Easy to deal with discrete variables.
- Easy to parallelize.
- For a correct choice of parameters the algorithm terminates ($\lim_{t \to +\infty} v(t) = 0$).
- Uses only objective function values.
- Convergence for a global optimum under strong assumptions (unpractical).
- High number of function evaluations.

Some properties

- Easy to implement.
- Easy to deal with discrete variables.
- Easy to parallelize.
- For a correct choice of parameters the algorithm terminates ($\lim_{t\to+\infty} v(t) = 0$).
- Uses only objective function values.
- Convergence for a global optimum under strong assumptions (unpractical).
- High number of function evaluations.

Some properties

- Easy to implement.
- Easy to deal with discrete variables.
- Easy to parallelize.
- For a correct choice of parameters the algorithm terminates ($\lim_{t \to +\infty} v(t) = 0$).
- Uses only objective function values.
- Convergence for a global optimum under strong assumptions (unpractical).
- High number of function evaluations.

Some properties

- Easy to implement.
- Easy to deal with discrete variables.
- Easy to parallelize.
- For a correct choice of parameters the algorithm terminates ($\lim_{t \to +\infty} v(t) = 0$).
- Uses only objective function values.
- Convergence for a global optimum under strong assumptions (unpractical).
- High number of function evaluations.

Some properties

- Easy to implement.
- Easy to deal with discrete variables.
- Easy to parallelize.
- For a correct choice of parameters the algorithm terminates ($\lim_{t \to +\infty} v(t) = 0$).
- Uses only objective function values.
- Convergence for a global optimum under strong assumptions (unpractical).
- High number of function evaluations.

Some properties

- Easy to implement.
- Easy to deal with discrete variables.
- Easy to parallelize.
- For a correct choice of parameters the algorithm terminates ($\lim_{t \to +\infty} v(t) = 0$).
- Uses only objective function values.
- Convergence for a global optimum under strong assumptions (unpractical).
- High number of function evaluations.

Some properties

- Easy to implement.
- Easy to deal with discrete variables.
- Easy to parallelize.
- For a correct choice of parameters the algorithm terminates ($\lim_{t\rightarrow+\infty} v(t) = 0$).
- Uses only objective function values.
- Convergence for a global optimum under strong assumptions (unpractical).
- High number of function evaluations.

Outline

1. Introduction
2. Particle swarm
3. Coordinate search
4. The hybrid algorithm
5. Numerical results with a set of test problems
6. Parameter estimation in Astrophysics

Introduction to direct search methods

- Direct search methods are an important class of optimization methods that try to minimize a function by comparing objective function values at a finite number of points.
- Direct search methods do not use derivative information of the objective function nor try to approximate it.
- Coordinate search is a simple direct search method.
- One of the most important definitions in coordinate search is *positive spanning sets*.

Introduction to direct search methods

- Direct search methods are an important class of optimization methods that try to minimize a function by comparing objective function values at a finite number of points.
- Direct search methods do not use derivative information of the objective function nor try to approximate it.

- Coordinate search is a simple direct search method.
- One of the most important definitions in coordinate search is *positive spanning sets*.

Introduction to direct search methods

- Direct search methods are an important class of optimization methods that try to minimize a function by comparing objective function values at a finite number of points.
- Direct search methods do not use derivative information of the objective function nor try to approximate it.

- Coordinate search is a simple direct search method.
- One of the most important definitions in coordinate search is *positive spanning sets.*

Introduction to direct search methods

- Direct search methods are an important class of optimization methods that try to minimize a function by comparing objective function values at a finite number of points.
- Direct search methods do not use derivative information of the objective function nor try to approximate it.

- Coordinate search is a simple direct search method.
- One of the most important definitions in coordinate search is *positive spanning sets*.

Positive spanning sets

What is a (positive) spanning set for $\mathbb{R}^n$?

Is a set of vector that generate all the space ($\mathbb{R}^n$), *i.e.*, all the point in the space are a linear combination (with nonnegative coefficients) of the vector in the set.

Example for $\mathbb{R}^2$

$$\left\langle \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} -1 \\ -1 \end{pmatrix} \right\rangle$$

$$\begin{aligned} \begin{pmatrix} 5 \\ -2 \end{pmatrix} &= 9 \begin{pmatrix} 1 \\ 0 \end{pmatrix} + 2 \begin{pmatrix} 0 \\ 1 \end{pmatrix} + 4 \begin{pmatrix} -1 \\ -1 \end{pmatrix} \\ &= 8 \begin{pmatrix} 1 \\ 0 \end{pmatrix} + 1 \begin{pmatrix} 0 \\ 1 \end{pmatrix} + 3 \begin{pmatrix} -1 \\ -1 \end{pmatrix} \end{aligned}$$

Positive spanning sets

What is a (positive) spanning set for $\mathbb{R}^n$?

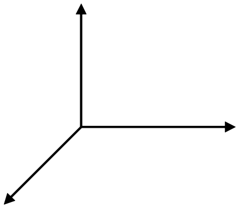
Is a set of vector that generate all the space ($\mathbb{R}^n$), *i.e.*, all the point in the space are a linear combination (with nonnegative coefficients) of the vector in the set.

Example for $\mathbb{R}^2$

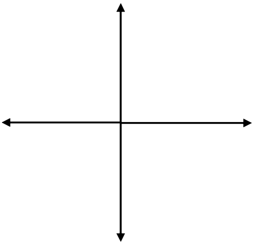
$$\left\langle \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} -1 \\ -1 \end{pmatrix} \right\rangle$$

$$\begin{aligned} \begin{pmatrix} 5 \\ -2 \end{pmatrix} &= 9 \begin{pmatrix} 1 \\ 0 \end{pmatrix} + 2 \begin{pmatrix} 0 \\ 1 \end{pmatrix} + 4 \begin{pmatrix} -1 \\ -1 \end{pmatrix} \\ &= 8 \begin{pmatrix} 1 \\ 0 \end{pmatrix} + 1 \begin{pmatrix} 0 \\ 1 \end{pmatrix} + 3 \begin{pmatrix} -1 \\ -1 \end{pmatrix} \end{aligned}$$

Type of basis



Minimal basis
with $n+1$ vectors
(3 in the $\mathbb{R}^2$ case).



Maximal basis
with $2n$ vectors
(4 in the $\mathbb{R}^2$ case).

Some definitions

Positive maximal basis

Formed by the coordinate vectors and their negative counterparts:

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

$D_\oplus$ spans $\mathbb{R}^n$ with nonnegative coefficients.

Coordinate search

The direct search method based on $D_\oplus$ is known as coordinate or compass search.

Some definitions

Positive maximal basis

Formed by the coordinate vectors and their negative counterparts:

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

$D_\oplus$ spans $\mathbb{R}^n$ with nonnegative coefficients.

Coordinate search

The direct search method based on $D_\oplus$ is known as coordinate or compass search.

Some definitions

Sets

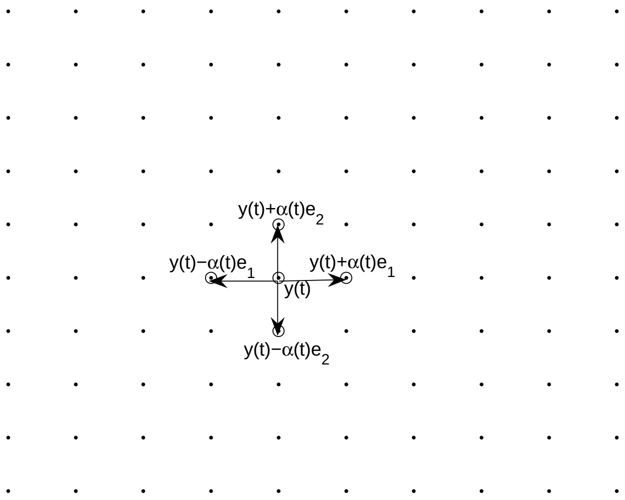
Given $D_\oplus$ and the current point $y(t)$, two sets of points are defined: a grid M_t and the poll set P_t.
The grid M_t is given by

$$M_t \; = \; \left\{ y(t) + \alpha(t) D_\oplus z, \; z \in \mathbb{N}_0^{|D_\oplus|} \right\},$$

where $\alpha(t) > 0$ is the grid size parameter.
The poll set is given by

$$P_t \; = \; \left\{ y(t) + \alpha(t) d, \; d \in D_\oplus \right\}.$$

Example of M_t and P_t



The grid M_t and the set P_t when $D_\oplus = \{e_1, e_2, -e_1, -e_2\}$

Coordinate search

- The search step conducts a finite search on the grid M_t.
- If no success is obtained in the search step then a poll step follows.
- The poll step evaluates the objective function at the elements of P_t, searching for points which have a lower objective function value.
- If success is attained, the value of $\alpha(t)$ may be increased, otherwise it is reduced.

Coordinate search

- The search step conducts a finite search on the grid M_t.
- If no success is obtained in the search step then a poll step follows.
- The poll step evaluates the objective function at the elements of P_t, searching for points which have a lower objective function value.
- If success is attained, the value of $\alpha(t)$ may be increased, otherwise it is reduced.

Coordinate search

- The search step conducts a finite search on the grid M_t.
- If no success is obtained in the search step then a poll step follows.
- The poll step evaluates the objective function at the elements of P_t, searching for points which have a lower objective function value.
- If success is attained, the value of $\alpha(t)$ may be increased, otherwise it is reduced.

Coordinate search

- The search step conducts a finite search on the grid M_t.
- If no success is obtained in the search step then a poll step follows.
- The poll step evaluates the objective function at the elements of P_t, searching for points which have a lower objective function value.
- If success is attained, the value of $\alpha(t)$ may be increased, otherwise it is reduced.

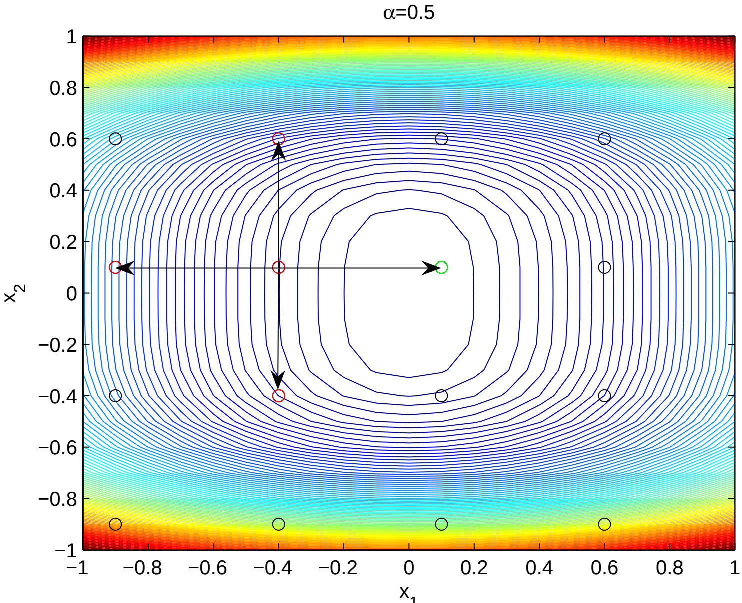
Handling bound constraints

For the coordinate search method it is sufficient to initialize the algorithm with a feasible initial guess ($y(0) \in \Omega$) and to use $\hat{f}$ as the objective function.

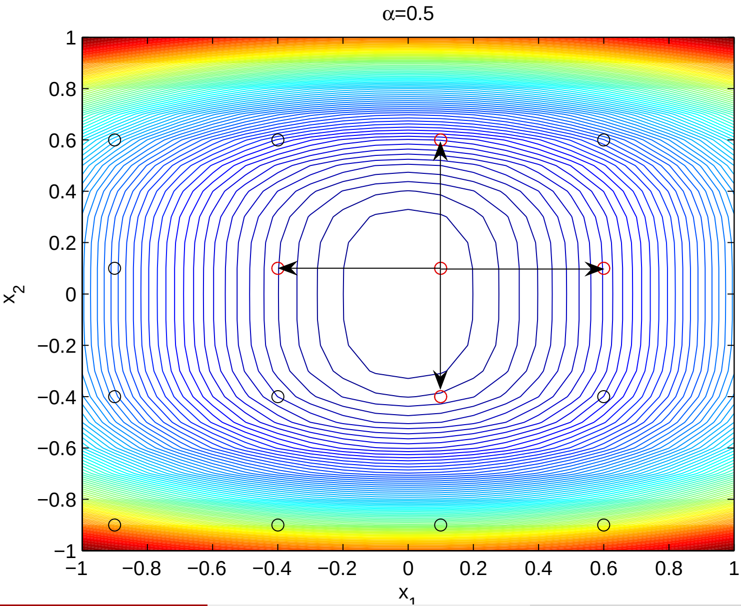
Penalty/Barrier function

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

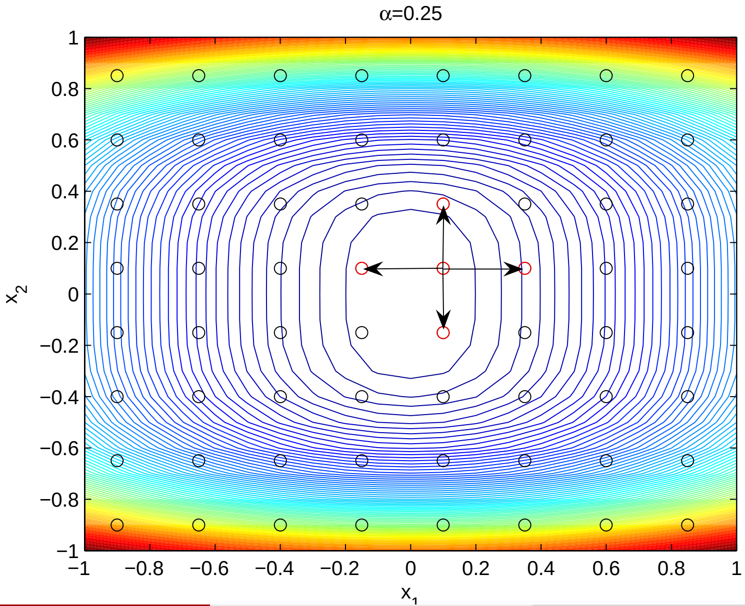
Example (without search step)



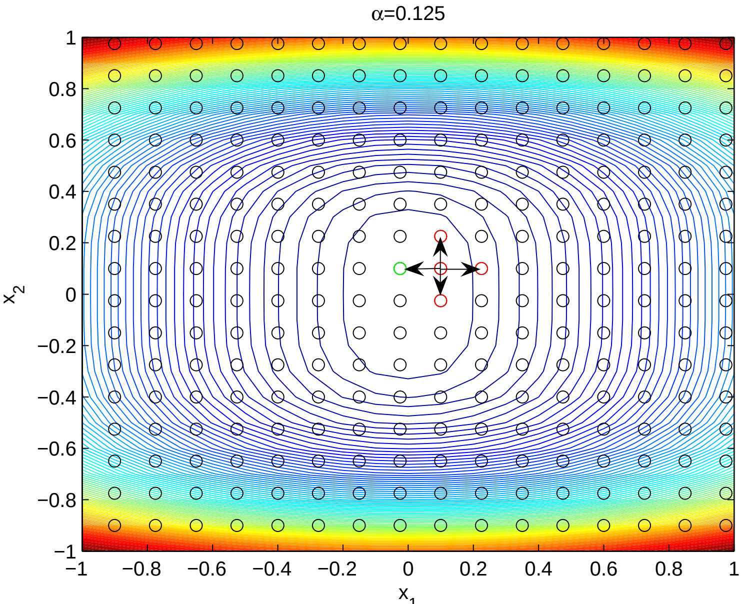
Example (without search step)



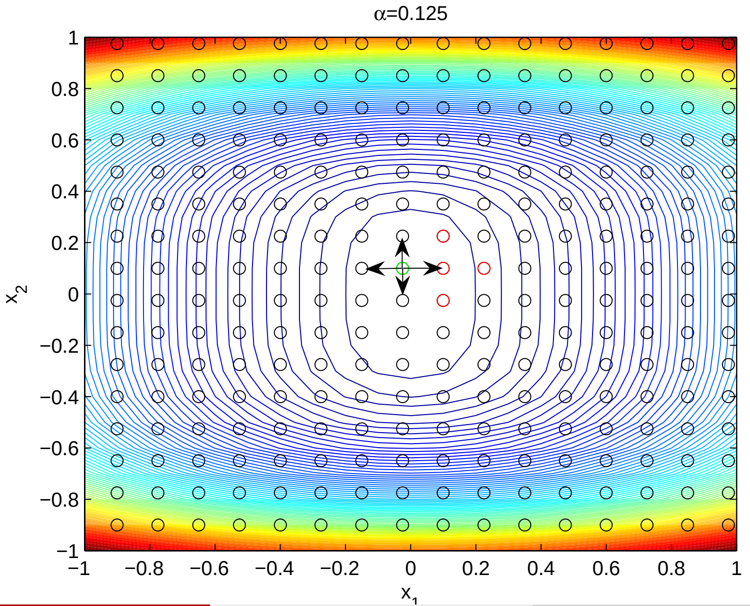
Example (without search step)



Example (without search step)



Example (without search step)



The algorithm

1. Given $\alpha_{tol} > 0$, $D = D_{\oplus}$, $\alpha(0) > 0$ e $y(0)$ (feasible). $t = 0$.
2. [Search step]
   Compute f at a finite set of points in the grid M_t. If there is a $z(t) \in M_t$ such that $\hat{f}(z(t)) < \hat{f}(y(t))$ then set $y(t+1) = z(t)$, $\alpha(t+1) = \phi(t)\alpha(t)$ (expansion — success).
3. [Poll step]
   The poll step is skipped with the search step was successful.
   - If there exists $d(t) \in D$ such that $\hat{f}(y(t) + \alpha(t)d(t)) < \hat{f}(y(t))$ then
   - Otherwise, $\hat{f}(y(t) + \alpha(t)d(t)) \geq \hat{f}(y(t))$ for all $d(t) \in D$ and
4. If $\alpha(t+1) < \alpha_{tol}$ then stop. Otherwise, $t = t + 1$ and go to Step 2.

The algorithm

1. Given $\alpha_{tol} > 0$, $D = D_{\oplus}$, $\alpha(0) > 0$ e $y(0)$ (feasible). $t = 0$.
2. **[Search step]**
   Compute f at a finite set of points in the grid M_t. If there is a $z(t) \in M_t$ such that $\hat{f}(z(t)) < \hat{f}(y(t))$ then set $y(t+1) = z(t)$, $\alpha(t+1) = \phi(t)\alpha(t)$ (expansion — success).
3. **[Poll step]**
   The poll step is skipped with the search step was successful.
   - If there exists $d(t) \in D$ such that $\hat{f}(y(t) + \alpha(t)d(t)) < \hat{f}(y(t))$ then
   - Otherwise, $\hat{f}(y(t) + \alpha(t)d(t)) \geq \hat{f}(y(t))$ for all $d(t) \in D$ and
4. If $\alpha(t+1) < \alpha_{tol}$ then stop. Otherwise, $t = t + 1$ and go to Step 2.

The algorithm

1. Given $\alpha_{tol} > 0$, $D = D_{\oplus}$, $\alpha(0) > 0$ e $y(0)$ (feasible). $t = 0$.
2. **[Search step]**
   Compute f at a finite set of points in the grid M_t. If there is a $z(t) \in M_t$ such that $\hat{f}(z(t)) < \hat{f}(y(t))$ then set $y(t+1) = z(t)$, $\alpha(t+1) = \phi(t)\alpha(t)$ (expansion — success).
3. **[Poll step]**
   The poll step is skipped with the search step was successful.
   - If there exists $d(t) \in D$ such that $\hat{f}(y(t) + \alpha(t)d(t)) < \hat{f}(y(t))$ then
     - $y(t+1) = y(t) + \alpha(t)d(t)$ (success).
   - Otherwise, $\hat{f}(y(t) + \alpha(t)d(t)) \geq \hat{f}(y(t))$ for all $d(t) \in D$ and
4. If $\alpha(t+1) < \alpha_{tol}$ then stop. Otherwise, $t = t + 1$ and go to Step 2.

The algorithm

1. Given $\alpha_{tol} > 0$, $D = D_{\oplus}$, $\alpha(0) > 0$ e $y(0)$ (feasible). $t = 0$.
2. **[Search step]**
   Compute f at a finite set of points in the grid M_t. If there is a $z(t) \in M_t$ such that $\hat{f}(z(t)) < \hat{f}(y(t))$ then set $y(t+1) = z(t)$, $\alpha(t+1) = \phi(t)\alpha(t)$ (expansion — success).
3. **[Poll step]**
   The poll step is skipped with the search step was successful.
   - If there exists $d(t) \in D$ such that $\hat{f}(y(t) + \alpha(t)d(t)) < \hat{f}(y(t))$ then
     - $y(t+1) = y(t) + \alpha(t)d(t)$ (success).
     - $\alpha(t+1) = \phi(t)\alpha(t)$ (expansion).
   - Otherwise, $\hat{f}(y(t) + \alpha(t)d(t)) \geq \hat{f}(y(t))$ for all $d(t) \in D$ and
     - $y(t+1) = y(t)$ (unsuccessful).
     - $\alpha(t+1) = \theta(t)\alpha(t)$ (contraction).
4. If $\alpha(t+1) < \alpha_{tol}$ then stop. Otherwise, $t = t + 1$ and go to Step 2.

The algorithm

1. Given $\alpha_{tol} > 0$, $D = D_{\oplus}$, $\alpha(0) > 0$ e $y(0)$ (feasible). $t = 0$.
2. **[Search step]**
   Compute f at a finite set of points in the grid M_t. If there is a $z(t) \in M_t$ such that $\hat{f}(z(t)) < \hat{f}(y(t))$ then set $y(t+1) = z(t)$, $\alpha(t+1) = \phi(t)\alpha(t)$ (expansion — success).
3. **[Poll step]**
   The poll step is skipped with the search step was successful.
   - If there exists $d(t) \in D$ such that $\hat{f}(y(t) + \alpha(t)d(t)) < \hat{f}(y(t))$ then
     - $y(t+1) = y(t) + \alpha(t)d(t)$ (success).
     - $\alpha(t+1) = \phi(t)\alpha(t)$ (expansion).
   - Otherwise, $\hat{f}(y(t) + \alpha(t)d(t)) \geq \hat{f}(y(t))$ for all $d(t) \in D$ and
     - $y(t+1) = y(t)$ (unsuccessful).
     - $\alpha(t+1) = \theta(t)\alpha(t)$ (contraction).
4. If $\alpha(t+1) < \alpha_{tol}$ then stop. Otherwise, $t = t+1$ and go to Step 2.

The algorithm

1. Given $\alpha_{tol} > 0$, $D = D_{\oplus}$, $\alpha(0) > 0$ e $y(0)$ (feasible). $t = 0$.
2. **[Search step]**
   Compute f at a finite set of points in the grid M_t. If there is a $z(t) \in M_t$ such that $\hat{f}(z(t)) < \hat{f}(y(t))$ then set $y(t+1) = z(t)$, $\alpha(t+1) = \phi(t)\alpha(t)$ (expansion — success).
3. **[Poll step]**
   The poll step is skipped with the search step was successful.
   - If there exists $d(t) \in D$ such that $\hat{f}(y(t) + \alpha(t)d(t)) < \hat{f}(y(t))$ then
     - $y(t+1) = y(t) + \alpha(t)d(t)$ (success).
     - $\alpha(t+1) = \phi(t)\alpha(t)$ (expansion).
   - Otherwise, $\hat{f}(y(t) + \alpha(t)d(t)) \geq \hat{f}(y(t))$ for all $d(t) \in D$ and
     - $y(t+1) = y(t)$ (unsuccessful).
     - $\alpha(t+1) = \theta(t)\alpha(t)$ (contraction).
4. If $\alpha(t+1) < \alpha_{tol}$ then stop. Otherwise, $t = t+1$ and go to Step 2.

The algorithm

1. Given $\alpha_{tol} > 0$, $D = D_{\oplus}$, $\alpha(0) > 0$ e $y(0)$ (feasible). $t = 0$.
2. **[Search step]**
   Compute f at a finite set of points in the grid M_t. If there is a $z(t) \in M_t$ such that $\hat{f}(z(t)) < \hat{f}(y(t))$ then set $y(t+1) = z(t)$, $\alpha(t+1) = \phi(t)\alpha(t)$ (expansion — success).
3. **[Poll step]**
   The poll step is skipped with the search step was successful.
   - If there exists $d(t) \in D$ such that $\hat{f}(y(t) + \alpha(t)d(t)) < \hat{f}(y(t))$ then
     - $y(t+1) = y(t) + \alpha(t)d(t)$ (success).
     - $\alpha(t+1) = \phi(t)\alpha(t)$ (expansion).
   - Otherwise, $\hat{f}(y(t) + \alpha(t)d(t)) \geq \hat{f}(y(t))$ for all $d(t) \in D$ and
     - $y(t+1) = y(t)$ (unsuccessful).
     - $\alpha(t+1) = \theta(t)\alpha(t)$ (contraction).
4. If $\alpha(t+1) < \alpha_{tol}$ then stop. Otherwise, $t = t+1$ and go to Step 2.

The algorithm

1. Given $\alpha_{tol} > 0$, $D = D_\oplus$, $\alpha(0) > 0$ e $y(0)$ (feasible). $t = 0$.
2. **[Search step]**
   Compute f at a finite set of points in the grid M_t. If there is a $z(t) \in M_t$ such that $\hat{f}(z(t)) < \hat{f}(y(t))$ then set $y(t+1) = z(t)$, $\alpha(t+1) = \phi(t)\alpha(t)$ (expansion — success).
3. **[Poll step]**
   The poll step is skipped with the search step was successful.
   - If there exists $d(t) \in D$ such that $\hat{f}(y(t) + \alpha(t)d(t)) < \hat{f}(y(t))$ then
     - $y(t+1) = y(t) + \alpha(t)d(t)$ (success).
     - $\alpha(t+1) = \phi(t)\alpha(t)$ (expansion).
   - Otherwise, $\hat{f}(y(t) + \alpha(t)d(t)) \geq \hat{f}(y(t))$ for all $d(t) \in D$ and
     - $y(t+1) = y(t)$ (unsuccessful).
     - $\alpha(t+1) = \theta(t)\alpha(t)$ (contraction).
4. If $\alpha(t+1) < \alpha_{tol}$ then stop. Otherwise, $t = t+1$ and go to Step 2.

The algorithm

1. Given $\alpha_{tol} > 0$, $D = D_\oplus$, $\alpha(0) > 0$ e $y(0)$ (feasible). $t = 0$.
2. **[Search step]**
   Compute f at a finite set of points in the grid M_t. If there is a $z(t) \in M_t$ such that $\hat{f}(z(t)) < \hat{f}(y(t))$ then set $y(t+1) = z(t)$, $\alpha(t+1) = \phi(t)\alpha(t)$ (expansion — success).
3. **[Poll step]**
   The poll step is skipped with the search step was successful.
   - If there exists $d(t) \in D$ such that $\hat{f}(y(t) + \alpha(t)d(t)) < \hat{f}(y(t))$ then
     - $y(t+1) = y(t) + \alpha(t)d(t)$ (success).
     - $\alpha(t+1) = \phi(t)\alpha(t)$ (expansion).
   - Otherwise, $\hat{f}(y(t) + \alpha(t)d(t)) \geq \hat{f}(y(t))$ for all $d(t) \in D$ and
     - $y(t+1) = y(t)$ (unsuccessful).
     - $\alpha(t+1) = \theta(t)\alpha(t)$ (contraction).
4. If $\alpha(t+1) < \alpha_{tol}$ then stop. Otherwise, $t = t + 1$ and go to Step 2.

The algorithm

1. Given $\alpha_{tol} > 0$, $D = D_\oplus$, $\alpha(0) > 0$ e $y(0)$ (feasible). $t = 0$.
2. **[Search step]**
   Compute f at a finite set of points in the grid M_t. If there is a $z(t) \in M_t$ such that $\hat{f}(z(t)) < \hat{f}(y(t))$ then set $y(t+1) = z(t)$, $\alpha(t+1) = \phi(t)\alpha(t)$ (expansion — success).
3. **[Poll step]**
   The poll step is skipped with the search step was successful.
   - If there exists $d(t) \in D$ such that $\hat{f}(y(t) + \alpha(t)d(t)) < \hat{f}(y(t))$ then
     - $y(t+1) = y(t) + \alpha(t)d(t)$ (success).
     - $\alpha(t+1) = \phi(t)\alpha(t)$ (expansion).
   - Otherwise, $\hat{f}(y(t) + \alpha(t)d(t)) \geq \hat{f}(y(t))$ for all $d(t) \in D$ and
     - $y(t+1) = y(t)$ (unsuccessful).
     - $\alpha(t+1) = \theta(t)\alpha(t)$ (contraction).
4. If $\alpha(t+1) < \alpha_{tol}$ then stop. Otherwise, $t = t+1$ and go to Step 2.

Outline

1 Introduction

2 Particle swarm

3 Coordinate search

4 The hybrid algorithm

5 Numerical results with a set of test problems

6 Parameter estimation in Astrophysics

Motivation

Hybrid algorithm

The hybrid algorithm tries to combine the best of both algorithms.

From particle swarm

The particle swarm ability of searching for the global optimum.

From coordinate search

The guarantee to obtain at least a stationary point. Some robustness.

Central idea

To apply the particle swarm algorithm in the search step and when no further success is possible to apply the poll step.

Motivation

Hybrid algorithm

The hybrid algorithm tries to combine the best of both algorithms.

From particle swarm

The particle swarm ability of searching for the global optimum.

From coordinate search

The guarantee to obtain at least a stationary point. Some robustness.

Central idea

To apply the particle swarm algorithm in the search step and when no further success is possible to apply the poll step.

Motivation

Hybrid algorithm

The hybrid algorithm tries to combine the best of both algorithms.

From particle swarm

The particle swarm ability of searching for the global optimum.

From coordinate search

The guarantee to obtain at least a stationary point. Some robustness.

Central idea

To apply the particle swarm algorithm in the search step and when no further success is possible to apply the poll step.

Motivation

Hybrid algorithm

The hybrid algorithm tries to combine the best of both algorithms.

From particle swarm

The particle swarm ability of searching for the global optimum.

From coordinate search

The guarantee to obtain at least a stationary point. Some robustness.

Central idea

To apply the particle swarm algorithm in the search step and when no further success is possible to apply the poll step.

Motivation for PSwarm

Central idea

A particle swarm iteration is performed in the search step and if no progress is attained a poll step is taken.

Key points

- In the first iterations the algorithm takes advantage of the particle swarm ability to find a global optimum (exploiting the search space), while in the last iterations the algorithm takes advantage of the pattern search robustness to find a stationary point.
- The number of particles in the swarm search can be decreased along the iterations (no need to have a large number of particles around a local optimum).

Motivation for PSwarm

Central idea

A particle swarm iteration is performed in the search step and if no progress is attained a poll step is taken.

Key points

- In the first iterations the algorithm takes advantage of the particle swarm ability to find a global optimum (exploiting the search space), while in the last iterations the algorithm takes advantage of the pattern search robustness to find a stationary point.
- The number of particles in the swarm search can be decreased along the iterations (no need to have a large number of particles around a local optimum).

Motivation for PSwarm

Central idea

A particle swarm iteration is performed in the search step and if no progress is attained a poll step is taken.

Key points

- In the first iterations the algorithm takes advantage of the particle swarm ability to find a global optimum (exploiting the search space), while in the last iterations the algorithm takes advantage of the pattern search robustness to find a stationary point.
- The number of particles in the swarm search can be decreased along the iterations (no need to have a large number of particles around a local optimum).

Motivation for PSwarm

Central idea

A particle swarm iteration is performed in the search step and if no progress is attained a poll step is taken.

Key points

- In the first iterations the algorithm takes advantage of the particle swarm ability to find a global optimum (exploiting the search space), while in the last iterations the algorithm takes advantage of the pattern search robustness to find a stationary point.
- The number of particles in the swarm search can be decreased along the iterations (no need to have a large number of particles around a local optimum).

The hybrid algorithm

1. Given $\alpha_{tol} > 0$, $D = D_{\oplus}$, $\alpha(0) > 0$, s, $v_{tol} > 0$. Set $\{x^1(0), \ldots, x^s(0)\}$ and $\{v^1(0), \ldots, v^s(0)\}$. $t = 0$.
2. $y^i(0) = x^i(0)$, $i = 1, \ldots, s$, e $\hat{y}(0) = \arg\min_{z \in \{y^1(0), \ldots, y^s(0)\}} f(z)$.
3. [Search step]
   $\hat{y}(t+1) = \hat{y}(t)$.
   For $i = 1, \ldots, s$ do:
   - $\hat{x}^i(t) = proj_{M_t}(x^i(t))$.
   - If $f(\hat{x}^i(t)) < f(y^i(t))$ then
   - Otherwise $y^i(t+1) = y^i(t)$.

The hybrid algorithm

1. Given $\alpha_{tol} > 0$, $D = D_{\oplus}$, $\alpha(0) > 0$, s, $v_{tol} > 0$. Set $\left\{x^1(0), \ldots, x^s(0)\right\}$ and $\{v^1(0), \ldots, v^s(0)\}$. $t = 0$.
2. $y^i(0) = x^i(0)$, $i = 1, \ldots, s$, e $\hat{y}(0) = \arg\min_{z \in \{y^1(0), \ldots, y^s(0)\}} f(z)$.
3. [Search step]
   $\hat{y}(t+1) = \hat{y}(t)$.
   For $i = 1, \ldots, s$ do:

The hybrid algorithm

1. Given $\alpha_{tol} > 0$, $D = D_{\oplus}$, $\alpha(0) > 0$, s, $v_{tol} > 0$. Set $\left\{x^1(0), \ldots, x^s(0)\right\}$ and $\{v^1(0), \ldots, v^s(0)\}$. $t = 0$.
2. $y^i(0) = x^i(0)$, $i = 1, \ldots, s$, e $\hat{y}(0) = \arg\min_{z \in \{y^1(0), \ldots, y^s(0)\}} f(z)$.
3. **[Search step]**
   $\hat{y}(t+1) = \hat{y}(t)$.
   For $i = 1, \ldots, s$ do:
   - $\hat{x}^i(t) = proj_{M_t}(x^i(t))$.
   - If $\hat{f}(\hat{x}^i(t)) < \hat{f}(y^i(t))$ then
     - $y^i(t+1) = \hat{x}^i(t)$.
     - $f(y^i(t+1)) < f(\hat{y}(t+1))$ then
       $\hat{y}(t+1) = y^i(t+1)$ (success).
       $\alpha(t+1) = \phi(t)\alpha(t)$ (expansion).
   - Otherwise $y^i(t+1) = y^i(t)$.

The hybrid algorithm

1. Given $\alpha_{tol} > 0$, $D = D_\oplus$, $\alpha(0) > 0$, s, $v_{tol} > 0$. Set $\left\{x^1(0), \ldots, x^s(0)\right\}$ and $\{v^1(0), \ldots, v^s(0)\}$. $t = 0$.
2. $y^i(0) = x^i(0)$, $i = 1, \ldots, s$, e $\hat{y}(0) = \arg\min_{z \in \{y^1(0), \ldots, y^s(0)\}} f(z)$.
3. **[Search step]**
   $\hat{y}(t+1) = \hat{y}(t)$.
   For $i = 1, \ldots, s$ do:
   - $\hat{x}^i(t) = proj_{M_t}(x^i(t))$.
   - If $\hat{f}(\hat{x}^i(t)) < \hat{f}(y^i(t))$ then
     - $y^i(t+1) = \hat{x}^i(t)$.
     - $f(y^i(t+1)) < f(\hat{y}(t+1))$ then
       $\hat{y}(t+1) = y^i(t+1)$ [success].
       $\alpha(t+1) = \phi(t)\alpha(t)$ [expansion].
   - Otherwise $y^i(t+1) = y^i(t)$.

The hybrid algorithm

1. Given $\alpha_{tol} > 0$, $D = D_{\oplus}$, $\alpha(0) > 0$, s, $v_{tol} > 0$. Set $\{x^1(0), \ldots, x^s(0)\}$ and $\{v^1(0), \ldots, v^s(0)\}$. $t = 0$.
2. $y^i(0) = x^i(0)$, $i = 1, \ldots, s$, e $\hat{y}(0) = \arg\min_{z \in \{y^1(0), \ldots, y^s(0)\}} f(z)$.
3. **[Search step]**
   $\hat{y}(t+1) = \hat{y}(t)$.
   For $i = 1, \ldots, s$ do:
   - $\hat{x}^i(t) = proj_{M_t}(x^i(t))$.
   - If $\hat{f}(\hat{x}^i(t)) < \hat{f}(y^i(t))$ then
     - $y^i(t+1) = \hat{x}^i(t)$.
     - $f(y^i(t+1)) < f(\hat{y}(t+1))$ then
       $\hat{y}(t+1) = y^i(t+1)$ (success).
       $\alpha(t+1) = \phi(t)\alpha(t)$ (expansion).
   - Otherwise $y^i(t+1) = y^i(t)$.

The hybrid algorithm

1. Given $\alpha_{tol} > 0$, $D = D_{\oplus}$, $\alpha(0) > 0$, s, $v_{tol} > 0$. Set $\{x^1(0), \dots, x^s(0)\}$ and $\{v^1(0), \dots, v^s(0)\}$. $t = 0$.
2. $y^i(0) = x^i(0)$, $i = 1, \dots, s$, e $\hat{y}(0) = \arg\min_{z \in \{y^1(0), \dots, y^s(0)\}} f(z)$.
3. **[Search step]**
   $\hat{y}(t+1) = \hat{y}(t)$.
   For $i = 1, \dots, s$ do:
   - $\hat{x}^i(t) = proj_{M_t}(x^i(t))$.
   - If $\hat{f}(\hat{x}^i(t)) < \hat{f}(y^i(t))$ then
     - $y^i(t+1) = \hat{x}^i(t)$.
     - $f(y^i(t+1)) < f(\hat{y}(t+1))$ then
       $\hat{y}(t+1) = y^i(t+1)$ (success).
       $\alpha(t+1) = \phi(t)\alpha(t)$ (expansion).
   - Otherwise $y^i(t+1) = y^i(t)$.

The hybrid algorithm

1. Given $\alpha_{tol} > 0$, $D = D_{\oplus}$, $\alpha(0) > 0$, s, $v_{tol} > 0$. Set $\{x^1(0), \ldots, x^s(0)\}$ and $\{v^1(0), \ldots, v^s(0)\}$. $t = 0$.
2. $y^i(0) = x^i(0)$, $i = 1, \ldots, s$, e $\hat{y}(0) = \arg\min_{z \in \{y^1(0), \ldots, y^s(0)\}} f(z)$.
3. **[Search step]**
   $\hat{y}(t+1) = \hat{y}(t)$.
   For $i = 1, \ldots, s$ do:
   - $\hat{x}^i(t) = proj_{M_t}(x^i(t))$.
   - If $\hat{f}(\hat{x}^i(t)) < \hat{f}(y^i(t))$ then
     - $y^i(t+1) = \hat{x}^i(t)$.
     - $f(y^i(t+1)) < f(\hat{y}(t+1))$ then
       $\hat{y}(t+1) = y^i(t+1)$ (success).
       $\alpha(t+1) = \phi(t)\alpha(t)$ (expansion).
   - Otherwise $y^i(t+1) = y^i(t)$.

The hybrid algorithm

1. Given $\alpha_{tol} > 0$, $D = D_{\oplus}$, $\alpha(0) > 0$, s, $v_{tol} > 0$. Set $\{x^1(0), \ldots, x^s(0)\}$ and $\{v^1(0), \ldots, v^s(0)\}$. $t = 0$.
2. $y^i(0) = x^i(0)$, $i = 1, \ldots, s$, e $\hat{y}(0) = \arg\min_{z \in \{y^1(0), \ldots, y^s(0)\}} f(z)$.
3. **[Search step]**
   $\hat{y}(t+1) = \hat{y}(t)$.
   For $i = 1, \ldots, s$ do:
   - $\hat{x}^i(t) = proj_{M_t}(x^i(t))$.
   - If $\hat{f}(\hat{x}^i(t)) < \hat{f}(y^i(t))$ then
     - $y^i(t+1) = \hat{x}^i(t)$.
     - $f(y^i(t+1)) < f(\hat{y}(t+1))$ then
       $\hat{y}(t+1) = y^i(t+1)$ (success).
       $\alpha(t+1) = \phi(t)\alpha(t)$ (expansion).
   - Otherwise $y^i(t+1) = y^i(t)$.

The hybrid algorithm

4. **[Poll Step]**
   The poll step is skipped with the search step was successful.
   - If there exists $d(t) \in D$ such that $\hat{f}(\hat{y}(t) + \alpha(t)d(t)) < \hat{f}(\hat{y}(t))$ then
     - $\hat{y}(t+1) = \hat{y}(t) + \alpha(t)d(t)$ (success).
     - $\alpha(t+1) = \phi(t)\alpha(t)$ (expansion).
   - Otherwise, $\hat{f}(\hat{y}(t) + \alpha(t)d(t)) \geq \hat{f}(\hat{y}(t))$ para todo o $d(t) \in D$, and
     - $\hat{y}(t+1) = \hat{y}(t)$ (unsuccessful).
     - $\alpha(t+1) = \theta(t)\alpha(t)$ (contraction).
5. Compute $v^i(t+1)$ and $x^i(t+1)$, $i = 1, \ldots, s$.
6. If $\alpha(t+1) < \alpha_{tol}$ and $\|v^i(t+1)\| < v_{tol}$, for all $i = 1, \ldots, s$, then stop. Otherwise, $t = t + 1$ and go to Step 3.

The hybrid algorithm

4. **[Poll Step]**
   The poll step is skipped with the search step was successful.
   - If there exists $d(t) \in D$ such that $\hat{f}(\hat{y}(t) + \alpha(t)d(t)) < \hat{f}(\hat{y}(t))$ then
     - $\hat{y}(t+1) = \hat{y}(t) + \alpha(t)d(t)$ (success).
     - $\alpha(t+1) = \phi(t)\alpha(t)$ (expansion).
   - Otherwise, $\hat{f}(\hat{y}(t) + \alpha(t)d(t)) \geq \hat{f}(\hat{y}(t))$ para todo o $d(t) \in D$, and
     - $\hat{y}(t+1) = \hat{y}(t)$ (unsuccessful).
     - $\alpha(t+1) = \theta(t)\alpha(t)$ (contraction).
5. Compute $v^i(t+1)$ and $x^i(t+1)$, $i = 1, \ldots, s$.
6. If $\alpha(t+1) < \alpha_{tol}$ and $\|v^i(t+1)\| < v_{tol}$, for all $i = 1, \ldots, s$, then stop. Otherwise, $t = t + 1$ and go to Step 3.

The hybrid algorithm

4. **[Poll Step]**
   The poll step is skipped with the search step was successful.
   - If there exists $d(t) \in D$ such that $\hat{f}(\hat{y}(t) + \alpha(t)d(t)) < \hat{f}(\hat{y}(t))$ then
     - $\hat{y}(t+1) = \hat{y}(t) + \alpha(t)d(t)$ (success).
     - $\alpha(t+1) = \phi(t)\alpha(t)$ (expansion).
   - Otherwise, $\hat{f}(\hat{y}(t) + \alpha(t)d(t)) \geq \hat{f}(\hat{y}(t))$ para todo o $d(t) \in D$, and
     - $\hat{y}(t+1) = \hat{y}(t)$ (unsuccessful).
     - $\alpha(t+1) = \theta(t)\alpha(t)$ (contraction).
5. Compute $v^i(t+1)$ and $x^i(t+1)$, $i = 1, \ldots, s$.
6. If $\alpha(t+1) < \alpha_{tol}$ and $\|v^i(t+1)\| < v_{tol}$, for all $i = 1, \ldots, s$, then stop. Otherwise, $t = t+1$ and go to Step 3.

The hybrid algorithm

4. **[Poll Step]**
   The poll step is skipped with the search step was successful.
   - If there exists $d(t) \in D$ such that $\hat{f}(\hat{y}(t) + \alpha(t)d(t)) < \hat{f}(\hat{y}(t))$ then
     - $\hat{y}(t+1) = \hat{y}(t) + \alpha(t)d(t)$ (success).
     - $\alpha(t+1) = \phi(t)\alpha(t)$ (expansion).
   - Otherwise, $\hat{f}(\hat{y}(t) + \alpha(t)d(t)) \geq \hat{f}(\hat{y}(t))$ para todo o $d(t) \in D$, and
     - $\hat{y}(t+1) = \hat{y}(t)$ (unsuccessful).
     - $\alpha(t+1) = \theta(t)\alpha(t)$ (contraction).
5. Compute $v^i(t+1)$ and $x^i(t+1)$, $i = 1, \ldots, s$.
6. If $\alpha(t+1) < \alpha_{tol}$ and $\|v^i(t+1)\| < v_{tol}$, for all $i = 1, \ldots, s$, then stop. Otherwise, $t = t+1$ and go to Step 3.

The hybrid algorithm

4. **[Poll Step]**
   The poll step is skipped with the search step was successful.
   - If there exists $d(t) \in D$ such that $\hat{f}(\hat{y}(t) + \alpha(t)d(t)) < \hat{f}(\hat{y}(t))$ then
     - $\hat{y}(t+1) = \hat{y}(t) + \alpha(t)d(t)$ (success).
     - $\alpha(t+1) = \phi(t)\alpha(t)$ (expansion).
   - Otherwise, $\hat{f}(\hat{y}(t) + \alpha(t)d(t)) \geq \hat{f}(\hat{y}(t))$ para todo o $d(t) \in D$, and
     - $\hat{y}(t+1) = \hat{y}(t)$ (unsuccessful).
     - $\alpha(t+1) = \theta(t)\alpha(t)$ (contraction).
5. Compute $v^i(t+1)$ and $x^i(t+1)$, $i = 1, \ldots, s$.
6. If $\alpha(t+1) < \alpha_{tol}$ and $\|v^i(t+1)\| < v_{tol}$, for all $i = 1, \ldots, s$, then stop. Otherwise, $t = t+1$ and go to Step 3.

The hybrid algorithm

4. **[Poll Step]**
   The poll step is skipped with the search step was successful.
   - If there exists $d(t) \in D$ such that $\hat{f}(\hat{y}(t) + \alpha(t)d(t)) < \hat{f}(\hat{y}(t))$ then
     - $\hat{y}(t+1) = \hat{y}(t) + \alpha(t)d(t)$ (success).
     - $\alpha(t+1) = \phi(t)\alpha(t)$ (expansion).
   - Otherwise, $\hat{f}(\hat{y}(t) + \alpha(t)d(t)) \geq \hat{f}(\hat{y}(t))$ para todo o $d(t) \in D$, and
     - $\hat{y}(t+1) = \hat{y}(t)$ (unsuccessful).
     - $\alpha(t+1) = \theta(t)\alpha(t)$ (contraction).
5. Compute $v^i(t+1)$ and $x^i(t+1)$, $i = 1, \ldots, s$.
6. If $\alpha(t+1) < \alpha_{tol}$ and $\|v^i(t+1)\| < v_{tol}$, for all $i = 1, \ldots, s$, then stop. Otherwise, $t = t+1$ and go to Step 3.

The hybrid algorithm

4. **[Poll Step]**
   The poll step is skipped with the search step was successful.
   - If there exists $d(t) \in D$ such that $\hat{f}(\hat{y}(t) + \alpha(t)d(t)) < \hat{f}(\hat{y}(t))$ then
     - $\hat{y}(t+1) = \hat{y}(t) + \alpha(t)d(t)$ (success).
     - $\alpha(t+1) = \phi(t)\alpha(t)$ (expansion).
   - Otherwise, $\hat{f}(\hat{y}(t) + \alpha(t)d(t)) \geq \hat{f}(\hat{y}(t))$ para todo o $d(t) \in D$, and
     - $\hat{y}(t+1) = \hat{y}(t)$ (unsuccessful).
     - $\alpha(t+1) = \theta(t)\alpha(t)$ (contraction).
5. Compute $v^i(t+1)$ and $x^i(t+1)$, $i = 1, \ldots, s$.
6. If $\alpha(t+1) < \alpha_{tol}$ and $\|v^i(t+1)\| < v_{tol}$, for all $i = 1, \ldots, s$, then stop. Otherwise, $t = t + 1$ and go to Step 3.

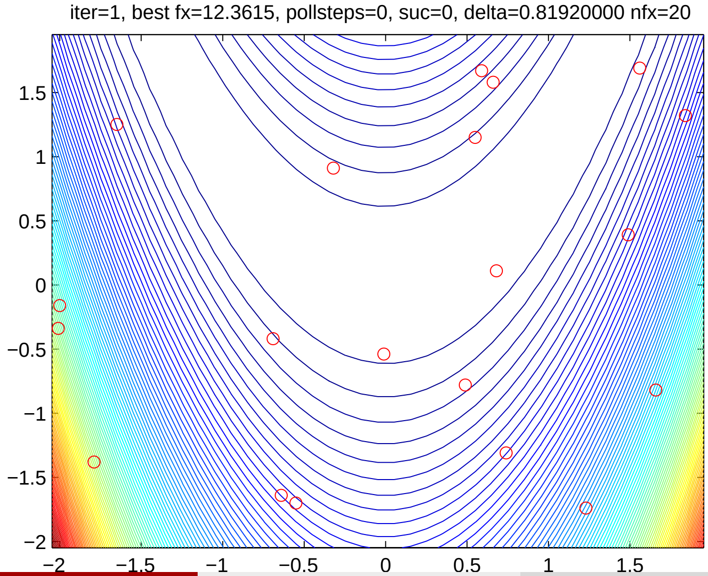
The hybrid algorithm

4. **[Poll Step]**
   The poll step is skipped with the search step was successful.
   - If there exists $d(t) \in D$ such that $\hat{f}(\hat{y}(t) + \alpha(t)d(t)) < \hat{f}(\hat{y}(t))$ then
     - $\hat{y}(t+1) = \hat{y}(t) + \alpha(t)d(t)$ (success).
     - $\alpha(t+1) = \phi(t)\alpha(t)$ (expansion).
   - Otherwise, $\hat{f}(\hat{y}(t) + \alpha(t)d(t)) \geq \hat{f}(\hat{y}(t))$ para todo o $d(t) \in D$, and
     - $\hat{y}(t+1) = \hat{y}(t)$ (unsuccessful).
     - $\alpha(t+1) = \theta(t)\alpha(t)$ (contraction).
5. Compute $v^i(t+1)$ and $x^i(t+1)$, $i = 1, \ldots, s$.
6. If $\alpha(t+1) < \alpha_{tol}$ and $\|v^i(t+1)\| < v_{tol}$, for all $i = 1, \ldots, s$, then stop. Otherwise, $t = t + 1$ and go to Step 3.

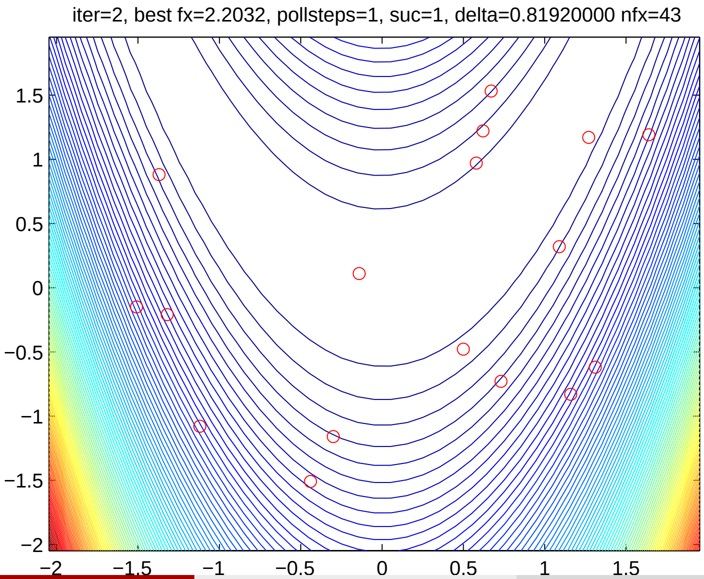
The hybrid algorithm

4. **[Poll Step]**
   The poll step is skipped with the search step was successful.
   - If there exists $d(t) \in D$ such that $\hat{f}(\hat{y}(t) + \alpha(t)d(t)) < \hat{f}(\hat{y}(t))$ then
     - $\hat{y}(t+1) = \hat{y}(t) + \alpha(t)d(t)$ (success).
     - $\alpha(t+1) = \phi(t)\alpha(t)$ (expansion).
   - Otherwise, $\hat{f}(\hat{y}(t) + \alpha(t)d(t)) \geq \hat{f}(\hat{y}(t))$ para todo o $d(t) \in D$, and
     - $\hat{y}(t+1) = \hat{y}(t)$ (unsuccessful).
     - $\alpha(t+1) = \theta(t)\alpha(t)$ (contraction).
5. Compute $v^i(t+1)$ and $x^i(t+1)$, $i = 1, \ldots, s$.
6. If $\alpha(t+1) < \alpha_{tol}$ and $\|v^i(t+1)\| < v_{tol}$, for all $i = 1, \ldots, s$, then stop. Otherwise, $t = t+1$ and go to Step 3.

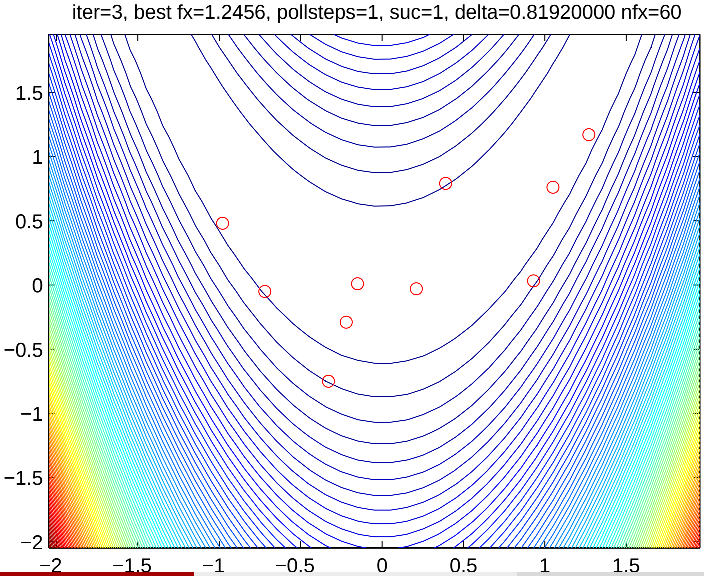
Example


iter=1, best fx=12.3615, pollsteps=0, suc=0, delta=0.81920000 nfx=20
1.5
1
0.5
0
−0.5
−1
−1.5
−2
−2
−1.5
−1
−0.5
0
0.5
1
1.5

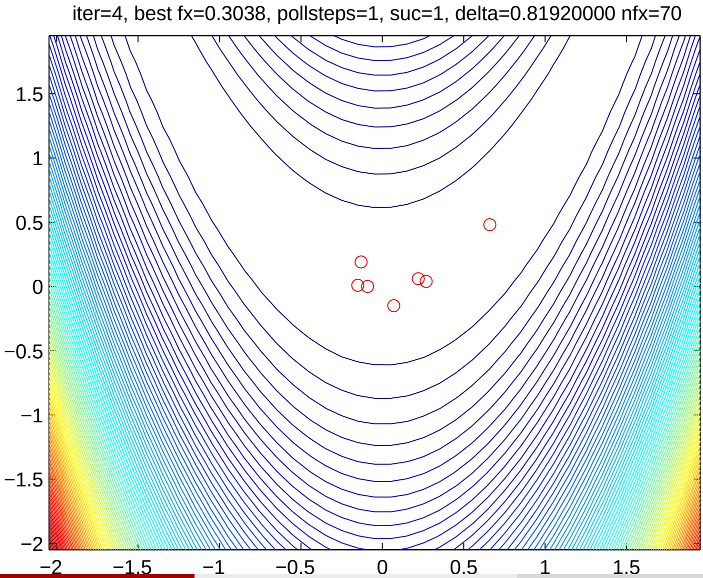
Example


iter=2, best fx=2.2032, pollsteps=1, suc=1, delta=0.81920000 nfx=43
1.5
1
0.5
0
−0.5
−1
−1.5
−2
−2
−1.5
−1
−0.5
0
0.5
1
1.5

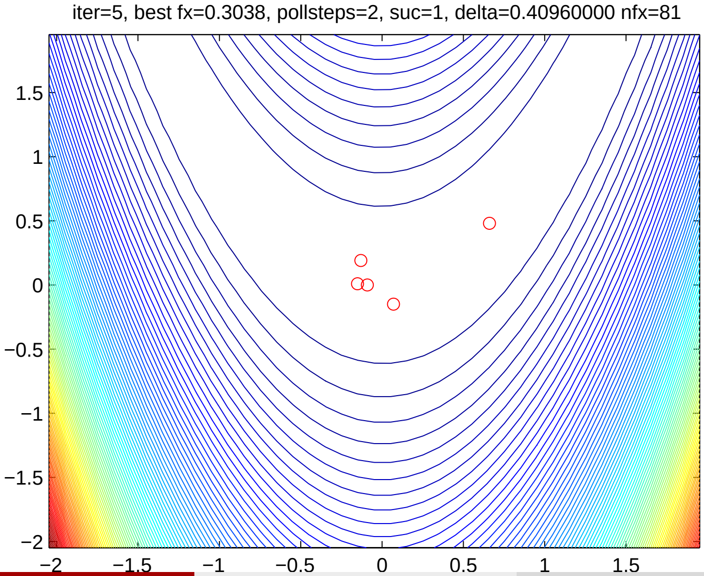
Example


iter=3, best fx=1.2456, pollsteps=1, suc=1, delta=0.81920000 nfx=60
1.5
1
0.5
0
−0.5
−1
−1.5
−2
−2 −1.5 −1 −0.5 0 0.5 1 1.5

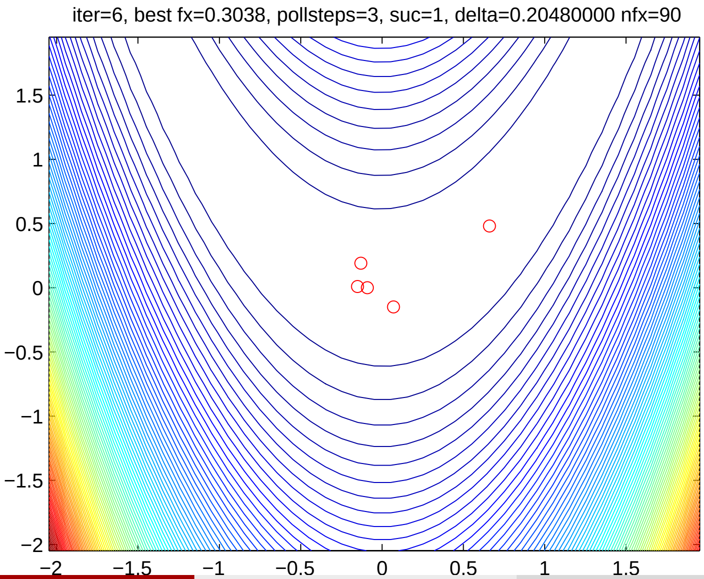
Example


iter=4, best fx=0.3038, pollsteps=1, suc=1, delta=0.81920000 nfx=70
1.5
1
0.5
0
−0.5
−1
−1.5
−2
−2
−1.5
−1
−0.5
0
0.5
1
1.5

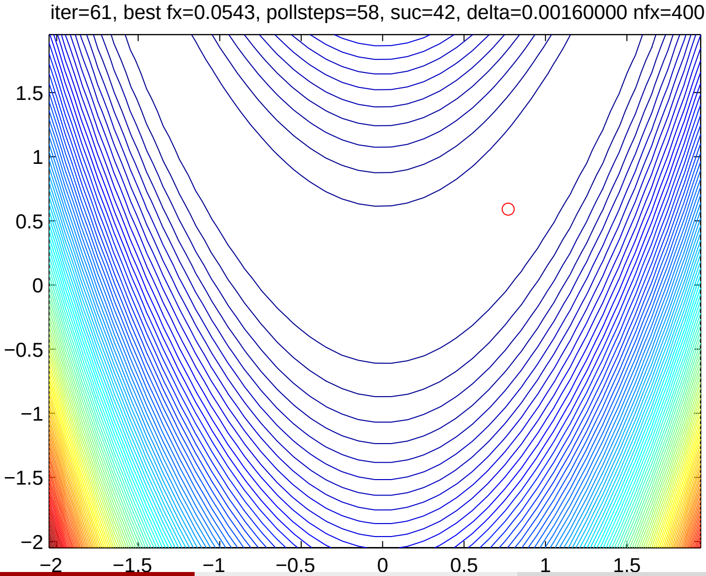
Example


iter=5, best fx=0.3038, pollsteps=2, suc=1, delta=0.40960000 nfx=81
1.5
1
0.5
0
−0.5
−1
−1.5
−2
−2
−1.5
−1
−0.5
0
0.5
1
1.5

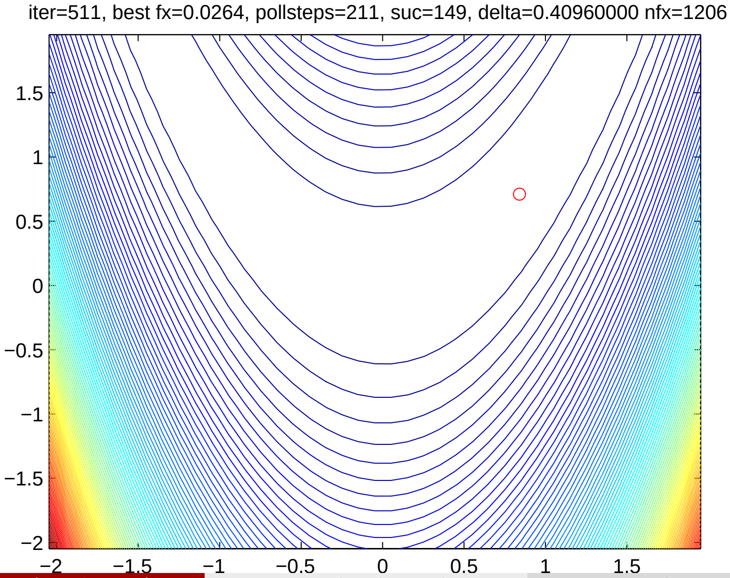
Example


iter=6, best fx=0.3038, pollsteps=3, suc=1, delta=0.20480000 nfx=90
1.5
1
0.5
0
−0.5
−1
−1.5
−2
−2
−1.5
−1
−0.5
0
0.5
1
1.5

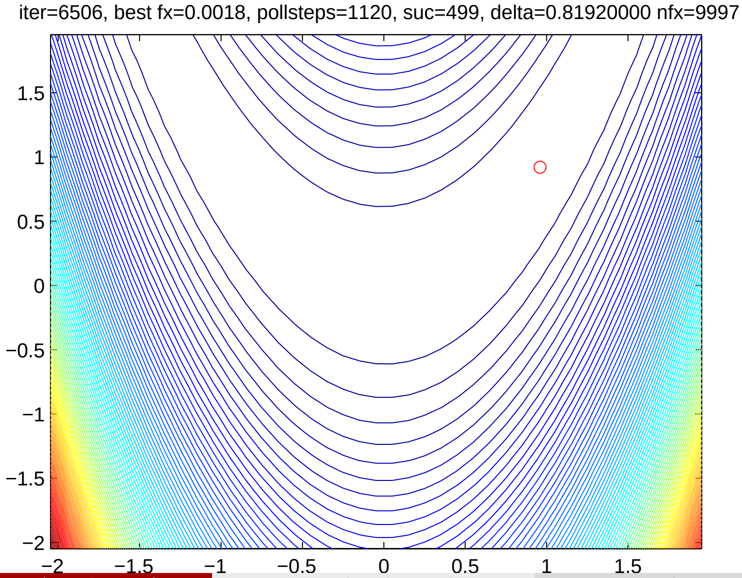
Example


iter=61, best fx=0.0543, pollsteps=58, suc=42, delta=0.00160000 nfx=400
1.5
1
0.5
0
−0.5
−1
−1.5
−2
−2
−1.5
−1
−0.5
0
0.5
1
1.5

Example


iter=511, best fx=0.0264, pollsteps=211, suc=149, delta=0.40960000 nfx=1206
1.5
1
0.5
0
−0.5
−1
−1.5
−2
−2
−1.5
−1
−0.5
0
0.5
1
1.5

Example


iter=6506, best fx=0.0018, pollsteps=1120, suc=499, delta=0.81920000 nfx=9997
1.5
1
0.5
0
−0.5
−1
−1.5
−2
−2
−1.5
−1
−0.5
0
0.5
1
1.5

Global convergence

Theorem

Let $L(\hat{y}(0)) = \{z \in \mathbb{R}^n : f(z) \leq f(\hat{y}(0))\}$ *be a bounded set. Then, there exists a subsequence* $\{\hat{y}(t_k)\}$ *of the iterates produced by the hybrid algorithm (with* $\alpha_{tol} = v_{tol} = 0$*) such that*

$$\lim_{k \longrightarrow +\infty} \hat{y}(t_k) \;=\; \hat{y}_* \quad \text{and} \quad \lim_{k \longrightarrow +\infty} \alpha(t_k) \;=\; 0,$$

for some $\hat{y}_* \in \Omega$ *and such that the subsequence* $\{t_k\}$ *consists of unsuccessful iterations.*

Finite termination

Theorem

Suppose that for t sufficiently large one has that $\iota(t)$, $E(y^i(t))$, $i = 1, \ldots, s$, and $E(\hat{y}(t))$ are constant and that $E(proj_{M_t}(x^i(t-1) + v^i(t))) = E(x^i(t-1) + v^i(t))$, $i = 1, \ldots, s$. Then, for an appropriate choice of the control parameters for particle swarm,

$$\lim_{t \longrightarrow +\infty} E(v_j^i(t)) \; = \; 0, \quad i = 1, \ldots, s, \; j = 1, \ldots, n.$$

and the hybrid algorithm will stop almost surely in a finite number of iterations.

Some considerations

Being the level set $L(y(0))$ bounded the strict decreasing sequences $\{f(y^i(t))\}$, $i = 1, \ldots, s$, and $\{f(\hat{y}(t))\}$ converge. Thus, it is reasonable to suppose that the expected values of $y^i(t)$, $i = 1, \ldots, s$, and $\hat{y}(t)$ also converge.

On the other hand, the difference between $proj_{M_t}(x^i(t-1) + v^i(t))$ and $x^i(t-1) + v^i(t)$ — and thus between their expected values — is a multiple of $\alpha(t)$ for some choices of D. This situation occurs in coordinate search, where $D = D_\oplus$. Since there is a subsequence of the mesh size parameters that converges to zero, there is at least the guarantee that the expected difference between $x^i(t-1) + v^i(t)$ and its projection onto M_t converges to zero in that subsequence.

Outline

1 Introduction

2 Particle swarm

3 Coordinate search

4 The hybrid algorithm

5 Numerical results with a set of test problems

6 Parameter estimation in Astrophysics

Test problems

- 122 problems were collected from the global optimization literature.
- 12 problems of large dimension (between 100 and 300 variables). The others are small (< 10) and medium size (< 30).
- Majority of objective functions are differentiable, but non-convex.
- All problems have simple bounds on the variables (needed for the search step — particle swarm).
- The test problems were coded in AMPL (*A Modeling Language for Mathematical Programming*).
- Test problems available on `http://www.norg.uminho.pt/aivaz` (under *software*).

Test problems

- 122 problems were collected from the global optimization literature.
- 12 problems of large dimension (between 100 and 300 variables). The others are small (< 10) and medium size (< 30).
- Majority of objective functions are differentiable, but non-convex.
- All problems have simple bounds on the variables (needed for the search step — particle swarm).
- The test problems were coded in AMPL (*A Modeling Language for Mathematical Programming*).
- Test problems available on `http://www.norg.uminho.pt/aivaz` (under *software*).

Test problems

- 122 problems were collected from the global optimization literature.
- 12 problems of large dimension (between 100 and 300 variables). The others are small (< 10) and medium size (< 30).
- Majority of objective functions are differentiable, but non-convex.
- All problems have simple bounds on the variables (needed for the search step — particle swarm).
- The test problems were coded in AMPL (*A Modeling Language for Mathematical Programming*).
- Test problems available on `http://www.norg.uminho.pt/aivaz` (under *software*).

Test problems

- 122 problems were collected from the global optimization literature.
- 12 problems of large dimension (between 100 and 300 variables). The others are small (< 10) and medium size (< 30).
- Majority of objective functions are differentiable, but non-convex.
- All problems have simple bounds on the variables (needed for the search step — particle swarm).
- The test problems were coded in AMPL (*A Modeling Language for Mathematical Programming*).
- Test problems available on `http://www.norg.uminho.pt/aivaz` (under *software*).

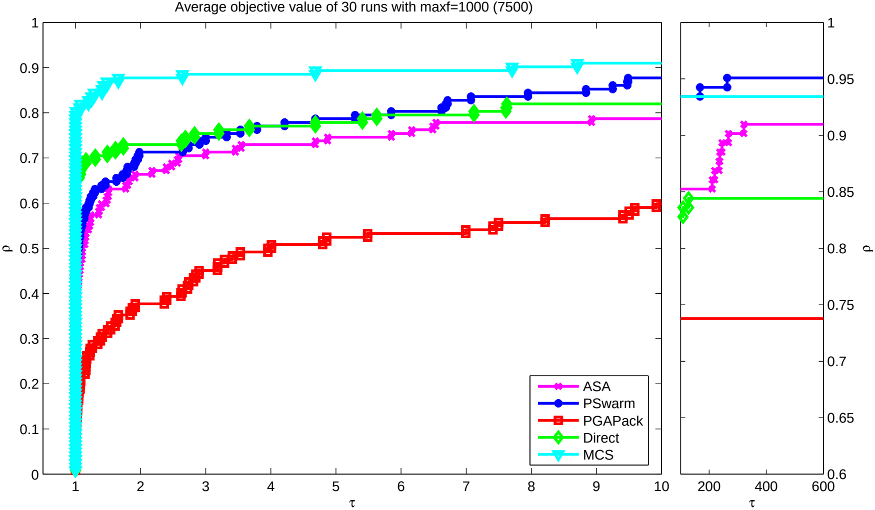
Test problems

- 122 problems were collected from the global optimization literature.
- 12 problems of large dimension (between 100 and 300 variables). The others are small (< 10) and medium size (< 30).
- Majority of objective functions are differentiable, but non-convex.
- All problems have simple bounds on the variables (needed for the search step — particle swarm).
- The test problems were coded in AMPL (*A Modeling Language for Mathematical Programming*).
- Test problems available on `http://www.norg.uminho.pt/aivaz` (under *software*).

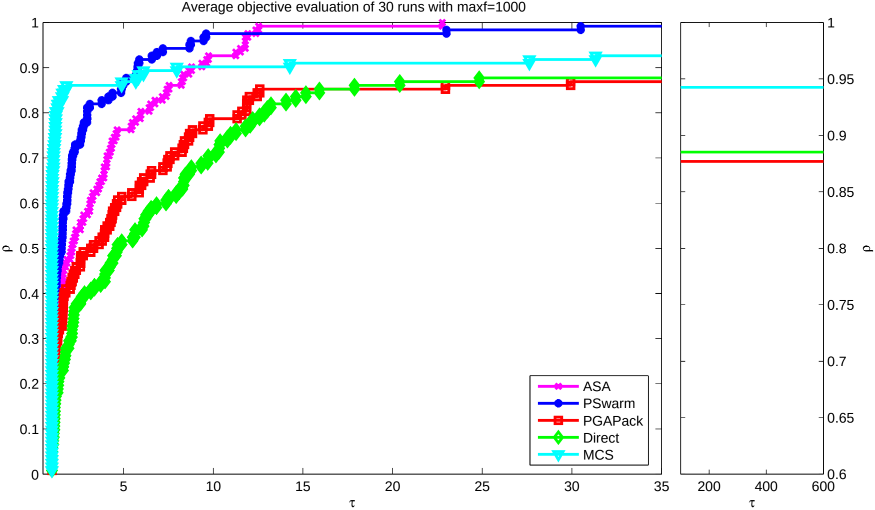
Test problems

- 122 problems were collected from the global optimization literature.
- 12 problems of large dimension (between 100 and 300 variables). The others are small (< 10) and medium size (< 30).
- Majority of objective functions are differentiable, but non-convex.
- All problems have simple bounds on the variables (needed for the search step — particle swarm).
- The test problems were coded in AMPL (*A Modeling Language for Mathematical Programming*).
- Test problems available on `http://www.norg.uminho.pt/aivaz` (under *software*).

Average objective value



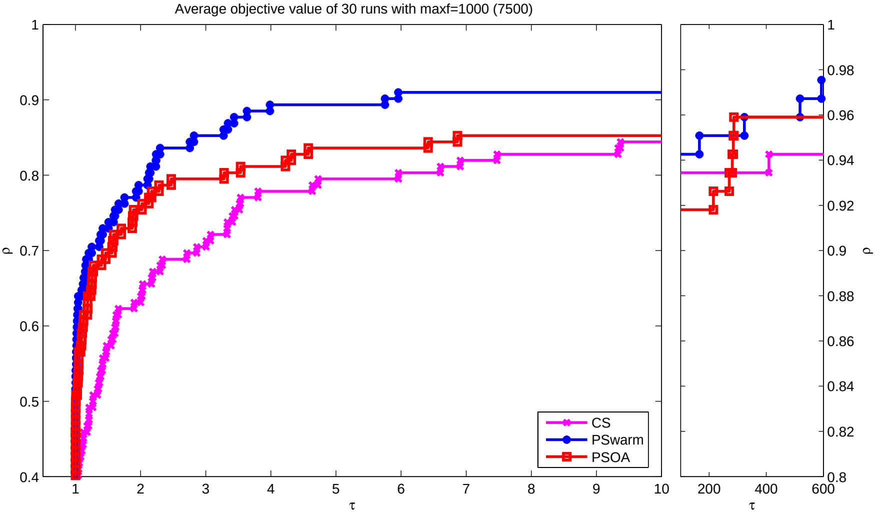
Average of objective function evaluations



Average number of objective function evaluations

$maxf$	ASA	PGAPack	PSwarm	Direct	MCS
1000	857	1009*	686	1107*	1837*
10000	5047	10009*	3603	11517*	4469

Coordinate search *vs* Particle swarm *vs* PSwarm



Outline

1. Introduction
2. Particle swarm
3. Coordinate search
4. The hybrid algorithm
5. Numerical results with a set of test problems
6. Parameter estimation in Astrophysics

The problem

Objective

To determine a set of stellar parameters (that define the star internal structure and evolution) from observable information.

Set of parameters to be determined

- M — stellar mass (relative to Sun mass $M_\odot$).
- X — abundance of hydrogen ($\%$).
- Y — abundance of helium ($\%$).
- Z — abundance of other elements ($Z = 100\% - X - Y$).
- t — star age (in Gyr = 1000 million years).
- two other parameters.

The problem

Objective

To determine a set of stellar parameters (that define the star internal structure and evolution) from observable information.

Set of parameters to be determined

- M — stellar mass (relative to Sun mass $M_\odot$).
- X — abundance of hydrogen (%).
- Y — abundance of helium (%).
- Z — abundance of other elements ($Z = 100\% - X - Y$).
- t — star age (in Gyr = 1000 million years).
- two other parameters.

The problem

Objective

To determine a set of stellar parameters (that define the star internal structure and evolution) from observable information.

Set of parameters to be determined

- M — stellar mass (relative to Sun mass $M_\odot$).
- X — abundance of hydrogen (%).
- Y — abundance of helium (%).
- Z — abundance of other elements ($Z = 100\% - X - Y$).
- t — star age (in Gyr = 1000 million years).
- two other parameters.

The problem

Objective

To determine a set of stellar parameters (that define the star internal structure and evolution) from observable information.

Set of parameters to be determined

- M — stellar mass (relative to Sun mass $M_\odot$).
- X — abundance of hydrogen (%).
- Y — abundance of helium (%).
- Z — abundance of other elements ($Z = 100\% - X - Y$).
- t — star age (in Gyr = 1000 million years).
- two other parameters.

The problem

Objective

To determine a set of stellar parameters (that define the star internal structure and evolution) from observable information.

Set of parameters to be determined

- M — stellar mass (relative to Sun mass $M_\odot$).
- X — abundance of hydrogen ($\%$).
- Y — abundance of helium ($\%$).
- Z — abundance of other elements ($Z = 100\% - X - Y$).
- t — star age (in Gyr = 1000 million years).
- two other parameters.

The problem

Objective

To determine a set of stellar parameters (that define the star internal structure and evolution) from observable information.

Set of parameters to be determined

- M — stellar mass (relative to Sun mass $M_\odot$).
- X — abundance of hydrogen ($\%$).
- Y — abundance of helium ($\%$).
- Z — abundance of other elements ($Z = 100\% - X - Y$).
- t — star age (in Gyr $=$ 1000 million years).
- two other parameters.

The problem

Objective

To determine a set of stellar parameters (that define the star internal structure and evolution) from observable information.

Set of parameters to be determined

- M — stellar mass (relative to Sun mass $M_\odot$).
- X — abundance of hydrogen ($\%$).
- Y — abundance of helium ($\%$).
- Z — abundance of other elements ($Z = 100\% - X - Y$).
- t — star age (in Gyr = 1000 million years).
- two other parameters.

The problem

Objective

To determine a set of stellar parameters (that define the star internal structure and evolution) from observable information.

Set of parameters to be determined

- M — stellar mass (relative to Sun mass $M_\odot$).
- X — abundance of hydrogen (%).
- Y — abundance of helium (%).
- Z — abundance of other elements ($Z = 100\% - X - Y$).
- t — star age (in Gyr = 1000 million years).
- two other parameters.

The problem

Observable data from spectrum analysis

- t_{eff} — stellar surface temperature.
- lum — total stellar luminosity.
- $\left(\frac{Z}{X}\right)$ — relation between the abundance of other elements and hydrogen.
- g — surface gravity (less accurate).

Parameters and observable data for Sun

$M = 1$ and $t = 4.6$Gyr, with $t_{eff} = 5777$, $lum = 1$ and $Z/X = 0.0245$.

This information is only available for Sun.

The problem

Observable data from spectrum analysis

- t_{eff} — stellar surface temperature.
- lum — total stellar luminosity.
- $\left(\frac{Z}{X}\right)$ — relation between the abundance of other elements and hydrogen.
- g — surface gravity (less accurate).

Parameters and observable data for Sun

$M = 1$ and $t = 4.6$Gyr, with $t_{eff} = 5777$, $lum = 1$ and $Z/X = 0.0245$.

This information is only available for Sun.

The problem

Observable data from spectrum analysis

- t_{eff} — stellar surface temperature.
- lum — total stellar luminosity.
- $\left(\frac{Z}{X}\right)$ — relation between the abundance of other elements and hydrogen.
- g — surface gravity (less accurate).

Parameters and observable data for Sun

$M = 1$ and $t = 4.6$Gyr, with $t_{eff} = 5777$, $lum = 1$ and $Z/X = 0.0245$.

This information is only available for Sun.

The problem

Observable data from spectrum analysis

- t_{eff} — stellar surface temperature.
- lum — total stellar luminosity.
- $\left(\frac{Z}{X}\right)$ — relation between the abundance of other elements and hydrogen.
- g — surface gravity (less accurate).

Parameters and observable data for Sun

$M = 1$ and $t = 4.6$Gyr, with $t_{eff} = 5777$, $lum = 1$ and $Z/X = 0.0245$.

This information is only available for Sun.

The problem

Observable data from spectrum analysis

- t_{eff} — stellar surface temperature.
- lum — total stellar luminosity.
- $\left(\frac{Z}{X}\right)$ — relation between the abundance of other elements and hydrogen.
- g — surface gravity (less accurate).

Parameters and observable data for Sun

$M = 1$ and $t = 4.6\text{Gyr}$, with $t_{eff} = 5777$, $lum = 1$ and $Z/X = 0.0245$.

This information is only available for Sun.

The problem

Observable data from spectrum analysis

- t_{eff} — stellar surface temperature.
- lum — total stellar luminosity.
- $\left(\frac{Z}{X}\right)$ — relation between the abundance of other elements and hydrogen.
- g — surface gravity (less accurate).

Parameters and observable data for Sun

$M = 1$ and $t = 4.6$Gyr, with $t_{eff} = 5777$, $lum = 1$ and $Z/X = 0.0245$.

This information is only available for Sun.

The optimization problem

The optimization problem

$$\min_{M,t,X,Y} \left(\frac{t_{eff} - t_{eff,obs}}{\delta t_{eff,obs}}\right)^2 + \left(\frac{lum - lum_{obs}}{\delta lum_{obs}}\right)^2 + \left(\frac{\frac{1-X-Y}{X} - \left(\frac{Z}{X}\right)_{obs}}{\delta\left(\frac{Z}{X}\right)_{obs}}\right)^2 + \left(\frac{g - g_{obs}}{\delta g_{obs}}\right)^2$$

Given M, t, fixing X, Y, and the two other parameters the parameters t_{eff}, lum and g are computed by simulating (CESAM code) a system of differentiable equations.

The equations of internal structure are five: conservation of mass and energy, hydrostatic equilibrium, energy transport, production and destruction of chemical elements by thermonuclear reactions.

The optimization problem

The optimization problem

$$\min_{M,t,X,Y} \left(\frac{t_{eff}-t_{eff,obs}}{\delta t_{eff,obs}}\right)^2+\left(\frac{lum-lum_{obs}}{\delta lum_{obs}}\right)^2 + \left(\frac{\frac{1-X-Y}{X}-\left(\frac{Z}{X}\right)_{obs}}{\delta\left(\frac{Z}{X}\right)_{obs}}\right)^2+\left(\frac{g-g_{obs}}{\delta g_{obs}}\right)^2$$

Given M, t, fixing X, Y, and the two other parameters the parameters t_{eff}, lum and g are computed by simulating (CESAM code) a system of differentiable equations.

The equations of internal structure are five: conservation of mass and energy, hydrostatic equilibrium, energy transport, production and destruction of chemical elements by thermonuclear reactions.

The optimization problem

The optimization problem

$$\min_{M,t,X,Y} \left(\frac{t_{eff} - t_{eff,obs}}{\delta t_{eff,obs}}\right)^2 + \left(\frac{lum - lum_{obs}}{\delta lum_{obs}}\right)^2 + \left(\frac{\frac{1-X-Y}{X} - \left(\frac{Z}{X}\right)_{obs}}{\delta\left(\frac{Z}{X}\right)_{obs}}\right)^2 + \left(\frac{g - g_{obs}}{\delta g_{obs}}\right)^2$$

Given M, t, fixing X, Y, and the two other parameters the parameters t_{eff}, lum and g are computed by simulating (CESAM code) a system of differentiable equations.

The equations of internal structure are five: conservation of mass and energy, hydrostatic equilibrium, energy transport, production and destruction of chemical elements by thermonuclear reactions.

Numerical results

Getting t_{eff}, lum and g – CESAM

t_{eff}, lum and g are computed by CESAM (Fortran 77 code), which is viewed as a black box function for the optimization process.

Optimization solver – PSwarm

PSwarm (C code).
Solver used with default options.

Linking PSwarm and CESAM

Optimization solver communicates with CESAM by input and output files.

Numerical results

Getting t_{eff}, lum and g – CESAM

t_{eff}, lum and g are computed by CESAM (Fortran 77 code), which is viewed as a black box function for the optimization process.

Optimization solver – `PSwarm`

PSwarm (C code).
Solver used with default options.

Linking PSwarm and CESAM

Optimization solver communicates with CESAM by input and output files.

Numerical results

Getting t_{eff}, lum and g – CESAM

t_{eff}, lum and g are computed by CESAM (Fortran 77 code), which is viewed as a black box function for the optimization process.

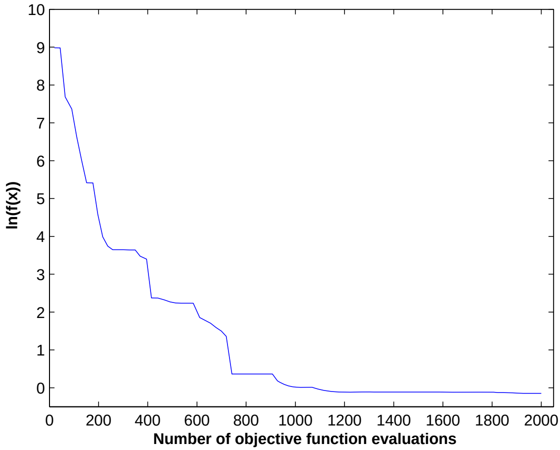
Optimization solver – PSwarm

PSwarm (C code).
Solver used with default options.

Linking PSwarm and CESAM

Optimization solver communicates with CESAM by input and output files.

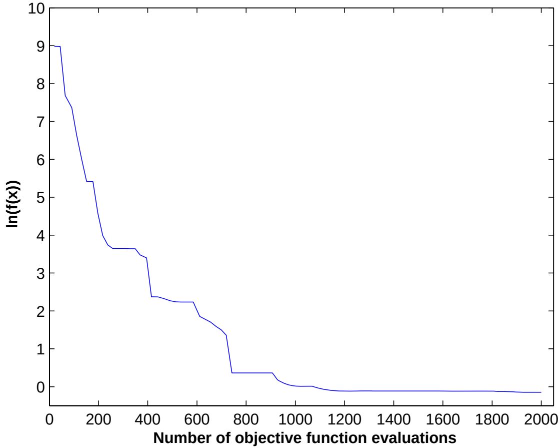
Numerical results - DH37124 star



- $f(\hat{y}(1)) \approx 7960$,
- $f(\hat{y}(100)) \approx 0.86$,
- 18 hours and 54 minutes real time (68070s),
- The CESAM call for the solution takes (printing to terminal)
  - real 1m28.359s
  - user 40.811s
  - sys 0.718s

Solution: $M = 0.81$, $t = 5.48$, $X = 0.66$, $Y = 0.33$, $a = 0.81$, $ov = 0.48$

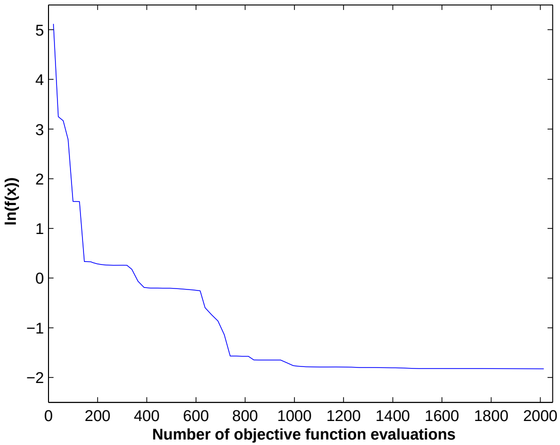
Numerical results - DH37124 star



- $f(\hat{y}(1)) \approx 7960$,
- $f(\hat{y}(100)) \approx 0.86$,
- 18 hours and 54 minutes real time (68070s),
- The CESAM call for the solution takes (printing to terminal)
  - real 1m28.359s
  - user 40.811s
  - sys 0.718s

Solution: $M = 0.81$, $t = 5.48$, $X = 0.66$, $Y = 0.33$, $a = 0.81$, $ov = 0.48$

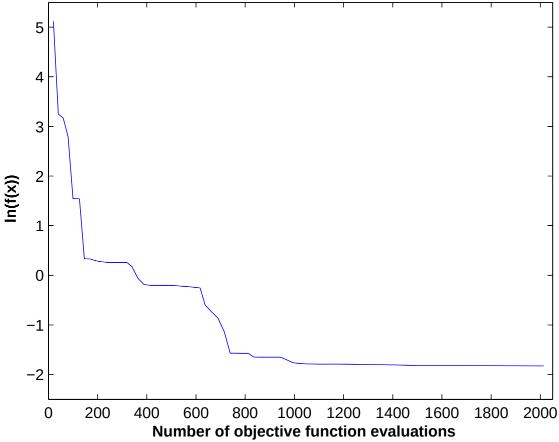
Numerical results - HD46375 star



- $f(\hat{y}(1)) \approx 167$,
- $f(\hat{y}(100)) \approx 0.16$,
- 21 hours and 54 minutes real time (78839s),
- The CESAM call for the solution takes (printing to terminal)
  - real 39.605s
  - user 37.719s
  - sys 0.450s

Solution: $M = 0.93$, $t = 4.87$, $X = 0.62$, $Y = 0.35$, $a = 1.08$, $ov = 0.50$

Numerical results - HD46375 star



- $f(\hat{y}(1)) \approx 167$,
- $f(\hat{y}(100)) \approx 0.16$,
- 21 hours and 54 minutes real time (78839s),
- The CESAM call for the solution takes (printing to terminal)
  - real 39.605s
  - user 37.719s
  - sys 0.450s

Solution: $M = 0.93$, $t = 4.87$, $X = 0.62$, $Y = 0.35$, $a = 1.08$, $ov = 0.50$

Numerical results

Parallel approach

- Each objective function evaluation takes around 1 minute to compute (on a desktop computer). One day for a full algorithm run (serial).
- We tested 5 fake stars (in order to validate the approach) and 10 real stars.
- For each star we performed 28 runs. (28*15=420 days!).
- A parallel version was implemented using MPI-2. The Centopeia (University of Coimbra) and SeARCH (University of Minho) parallel platforms were used to obtain the numerical results.
- About one day for 10 runs (parallel in 8 processors) — 42 particles with a maximum of 2000 o.f. evaluations.

Numerical results

Parallel approach

- Each objective function evaluation takes around 1 minute to compute (on a desktop computer). One day for a full algorithm run (serial).
- We tested 5 fake stars (in order to validate the approach) and 10 real stars.
- For each star we performed 28 runs. (28*15=420 days!).
- A parallel version was implemented using MPI-2. The Centopeia (University of Coimbra) and SeARCH (University of Minho) parallel platforms were used to obtain the numerical results.
- About one day for 10 runs (parallel in 8 processors) — 42 particles with a maximum of 2000 o.f. evaluations.

Numerical results

Parallel approach

- Each objective function evaluation takes around 1 minute to compute (on a desktop computer). One day for a full algorithm run (serial).
- We tested 5 fake stars (in order to validate the approach) and 10 real stars.
- For each star we performed 28 runs. (28*15=420 days!).
- A parallel version was implemented using MPI-2. The Centopeia (University of Coimbra) and SeARCH (University of Minho) parallel platforms were used to obtain the numerical results.
- About one day for 10 runs (parallel in 8 processors) — 42 particles with a maximum of 2000 o.f. evaluations.

Numerical results

Parallel approach

- Each objective function evaluation takes around 1 minute to compute (on a desktop computer). One day for a full algorithm run (serial).
- We tested 5 fake stars (in order to validate the approach) and 10 real stars.
- For each star we performed 28 runs. (28*15=420 days!).
- A parallel version was implemented using MPI-2. The Centopeia (University of Coimbra) and SeARCH (University of Minho) parallel platforms were used to obtain the numerical results.
- About one day for 10 runs (parallel in 8 processors) — 42 particles with a maximum of 2000 o.f. evaluations.

Numerical results

Parallel approach

- Each objective function evaluation takes around 1 minute to compute (on a desktop computer). One day for a full algorithm run (serial).
- We tested 5 fake stars (in order to validate the approach) and 10 real stars.
- For each star we performed 28 runs. (28*15=420 days!).
- A parallel version was implemented using MPI-2. The Centopeia (University of Coimbra) and SeARCH (University of Minho) parallel platforms were used to obtain the numerical results.
- About one day for 10 runs (parallel in 8 processors) — 42 particles with a maximum of 2000 o.f. evaluations.

Numerical results

Parallel approach

- Each objective function evaluation takes around 1 minute to compute (on a desktop computer). One day for a full algorithm run (serial).
- We tested 5 fake stars (in order to validate the approach) and 10 real stars.
- For each star we performed 28 runs. (28*15=420 days!).
- A parallel version was implemented using MPI-2. The Centopeia (University of Coimbra) and SeARCH (University of Minho) parallel platforms were used to obtain the numerical results.
- About one day for 10 runs (parallel in 8 processors) — 42 particles with a maximum of 2000 o.f. evaluations.

Numerical results

Average obtained results (in Red) *vs* the real data.

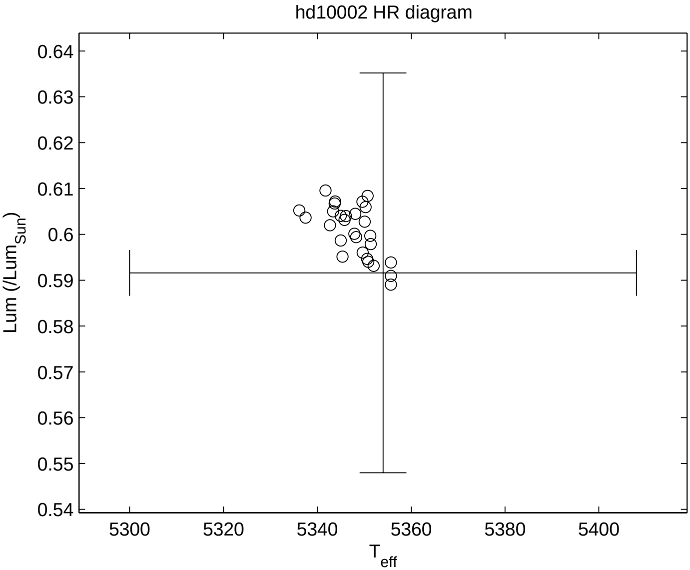
Star	M	t (Myr)	X	Y	α	ov	o.f. (average)
Sun	1.00	4600	0.715	0.268	1.63	0.00	
Sun	0.96	4691	0.68	0.31	1.55	0.265	0.272511931
fake1	0.85	1600	0.70	0.29	1.9	0.0	
fake1	0.84	2989	0.69	0.30	2.0	0.36	0.846046483
fake2	1.30	850	0.72	0.25	1.0	0.25	
fake2	1.20	4403	0.70	0.27	1.27	0.33	0.250562107
fake3	1.00	5000	0.68	0.30	0.7	0.15	
fake3	1.00	5499	0.68	0.30	0.72	0.28	0.209947500
fake4	0.70	5000	0.66	0.33	2.0	0.0	
fake4	0.71	3786	0.66	0.33	2.0	0.26	0.040181857
fake5	1.10	2500	0.62	0.36	1.4	0.3	
fake5	1.10	2956	0.62	0.36	1.57	0.22	0.232024714

Numerical results

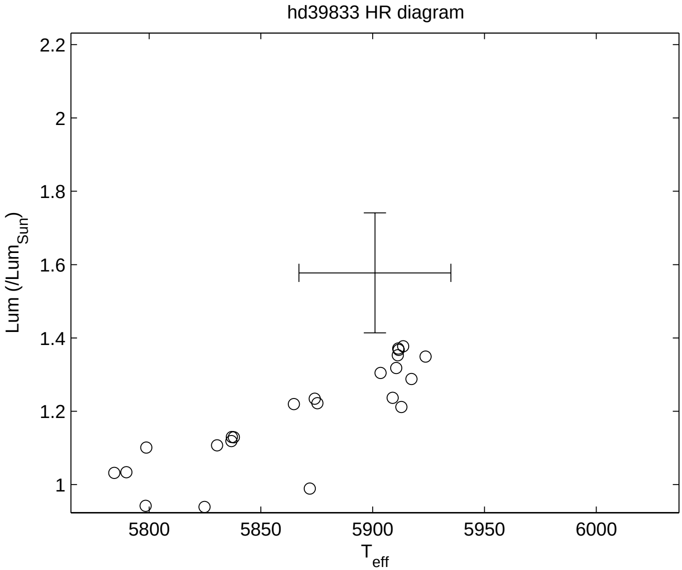
Average obtained results for real stars.

Star	M	t (Myr)	X	Y	α	ov	o.f. (average)
hd10002	0.87	5455	0.62	0.35	1.39	0.22	0.454073286
hd11226	1.12	3524	0.67	0.30	1.63	0.29	1.449135786
hd19994	1.28	2539	0.63	0.34	1.37	0.22	1.242964393
hd30177	1.02	5381	0.62	0.34	1.48	0.23	0.215747107
hd39833	1.24	1787	0.74	0.23	2.18	0.36	4.535001821
hd40979	1.08	3286	0.63	0.35	1.76	0.26	0.083869821
hd72659	1.18	4064	0.71	0.27	1.47	0.28	0.905840517
hd74868	1.26	2081	0.64	0.33	1.74	0.28	0.310089143
hd76700	1.15	4964	0.64	0.32	1.64	0.28	0.303584679
hd117618	1.09	4248	0.69	0.29	1.72	0.30	0.581501536

HR diagram with hd10002



HR diagram with hd39833



Future (present) work

Future (present) work

- Extend `PSwarm` to more general constrained optimization problems. Write a MATLAB code.
- Solve a range of astrophysics parameter estimation problems related to a number of different stars.

Future (present) work

Future (present) work

- Extend `PSwarm` to more general constrained optimization problems. Write a MATLAB code.
- Solve a range of astrophysics parameter estimation problems related to a number of different stars.

The end

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