Universidade do Minho

Um método de procura em padrão baseado em colónia de partículas para optimização não linear com limites nas variáveis

A. Ismael F. Vaz

Departamento de Produção e Sistemas

Escola de Engenharia, Universidade do Minho

aivaz@dps.uminho.pt

Luís Nunes Vicente

Departamento de Matemática,

Universidade de Coimbra

lnv@mat.uc.pt

Trabalhos parcialmente suportados pela FCT (POCI/MAT/58957/2004, POCI/MAT/59442/2004) e centro de investigação Algoritmi

Universidade do Minho

Conteúdo

- Introdução.
- Colónias de partículas (*Particle Swarm*).
- Procura em padrão (*Pattern search*).
- O algoritmo híbrido (`PSwarm`).
- Convergência.
- Resultados numéricos e conclusões.

Universidade do Minho

Conteúdo

- Introdução.
- Colónias de partículas (*Particle Swarm*).
- Procura em padrão (*Pattern search*).
- O algoritmo híbrido (`PSwarm`).
- Convergência.
- Resultados numéricos e conclusões.

Universidade do Minho

Conteúdo

- Introdução.
- Colónias de partículas (*Particle Swarm*).
- Procura em padrão (*Pattern search*).
- O algoritmo híbrido (`PSwarm`).
- Convergência.
- Resultados numéricos e conclusões.

Universidade do Minho

Conteúdo

- Introdução.
- Colónias de partículas (*Particle Swarm*).
- Procura em padrão (*Pattern search*).
- O algoritmo híbrido (`PSwarm`).
- Convergência.
- Resultados numéricos e conclusões.

Universidade do Minho

Conteúdo

Universidade do Minho

Conteúdo

- Introdução.
- Colónias de partículas (*Particle Swarm*).
- Procura em padrão (*Pattern search*).
- O algoritmo híbrido (`PSwarm`).
- Convergência.
- Resultados numéricos e conclusões.

Universidade do Minho

Introdução

Formulação

Pretende-se resolver problemas da forma

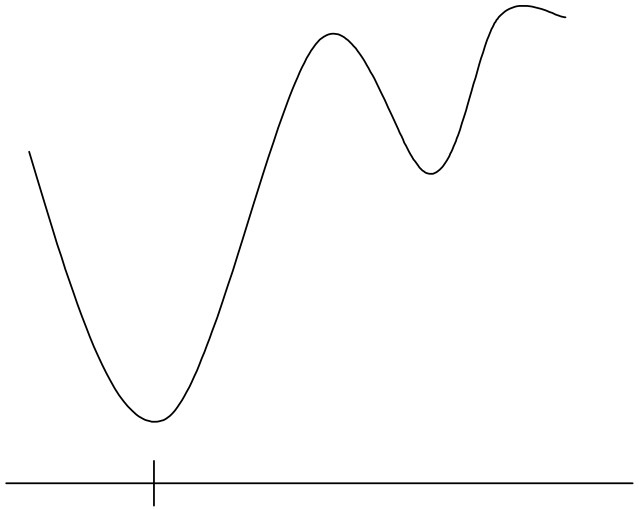
$$\min_{z \in \mathbb{R}^n} \; f(z)$$
$$\text{s.a} \quad \ell \leq z \leq u,$$

onde as desigualdades $\ell \leq z \leq u$ são componente a componente.

Para aplicar as colónias de partículas ou a procura em padrão não é necessário impor nenhum tipo de suavidade na função objectivo $f(z)$.

Para o estudo da convergência da procura em padrão e consequentemente do algoritmo híbrido, é necessário impor alguma suavidade em $f(z)$.

Optimização global



Em optimização global pretende-se determinar o óptimo global.

Não é possível distinguir entre um óptimo local e global através das condições de optimalidade.

Não confundir optimização global com convergência global ou local.

Universidade do Minho

Técnicas mais populares - Estocásticas

- Algoritmos genéticos/Estratégias evolutivas;
  - Baseado numa população (de pontos).
  - Aplica cruzamento (*crossover*) e mutação (*mutation*) de indivíduos para obter as melhores soluções.
- Arrefecimento simulado (*Simulated annealing*);
  - Baseado num único ponto.
  - Gera pseudo-aleatoriamente um novo ponto e usa uma função (arrefecimento) para o aceitar ou não.
- Colónias de partículas;
  - Baseado numa população (de pontos).
  - Usa o conhecimento adquirido por cada partícula e pela colónia.
- entre outras (colónias de formigas, etc..)

Universidade do Minho

Técnicas mais populares - Estocásticas

- Algoritmos genéticos/Estratégias evolutivas;
  - Baseado numa população (de pontos).
  - Aplica cruzamento (*crossover*) e mutação (*mutation*) de indivíduos para obter as melhores soluções.
- Arrefecimento simulado (*Simulated annealing*);
  - Baseado num único ponto.
  - Gera pseudo-aleatoriamente um novo ponto e usa uma função (arrefecimento) para o aceitar ou não.
- Colónias de partículas;
  - Baseado numa população (de pontos).
  - Usa o conhecimento adquirido por cada partícula e pela colónia.
- entre outras (colónias de formigas, etc..)

Universidade do Minho

Técnicas mais populares - Estocásticas

- Algoritmos genéticos/Estratégias evolutivas;
  - Baseado numa população (de pontos).
  - Aplica cruzamento (*crossover*) e mutação (*mutation*) de indivíduos para obter as melhores soluções.
- Arrefecimento simulado (*Simulated annealing*);
  - Baseado num único ponto.
  - Gera pseudo-aleatoriamente um novo ponto e usa uma função (arrefecimento) para o aceitar ou não.
- Colónias de partículas;
  - Baseado numa população (de pontos).
  - Usa o conhecimento adquirido por cada partícula e pela colónia.
- entre outras (colónias de formigas, etc..)

Universidade do Minho

Técnicas mais populares - Estocásticas

- Algoritmos genéticos/Estratégias evolutivas;
  - Baseado numa população (de pontos).
  - Aplica cruzamento (*crossover*) e mutação (*mutation*) de indivíduos para obter as melhores soluções.
- Arrefecimento simulado (*Simulated annealing*);
  - Baseado num único ponto.
  - Gera pseudo-aleatoriamente um novo ponto e usa uma função (arrefecimento) para o aceitar ou não.
- Colónias de partículas;
  - Baseado numa população (de pontos).
  - Usa o conhecimento adquirido por cada partícula e pela colónia.
- entre outras (colónias de formigas, etc..)

Universidade do Minho

Técnicas mais populares - Determinísticas

- Análise intervalar. Uso de aritmética intervalar para determinar um ou todos os óptimos globais.
- Optimização Lipschitz. Uso da constante de funções Lipschitz contínuas.
- Enumeração exaustiva (*Branch and bound*). Pode usar as duas estratégias anteriores para determinar limites da função.
- entre outras.

Universidade do Minho

Técnicas mais populares - Determinísticas

- Análise intervalar. Uso de aritmética intervalar para determinar um ou todos os óptimos globais.
- Optimização Lipschitz. Uso da constante de funções Lipschitz contínuas.
- Enumeração exaustiva (*Branch and bound*). Pode usar as duas estratégias anteriores para determinar limites da função.
- entre outras.

Universidade do Minho

Técnicas mais populares - Determinísticas

- Análise intervalar. Uso de aritmética intervalar para determinar um ou todos os óptimos globais.
- Optimização Lipschitz. Uso da constante de funções Lipschitz contínuas.
- Enumeração exaustiva (*Branch and bound*). Pode usar as duas estratégias anteriores para determinar limites da função.
- entre outras.

Universidade do Minho

Técnicas mais populares - Determinísticas

- Análise intervalar. Uso de aritmética intervalar para determinar um ou todos os óptimos globais.
- Optimização Lipschitz. Uso da constante de funções Lipschitz contínuas.
- Enumeração exaustiva (*Branch and bound*). Pode usar as duas estratégias anteriores para determinar limites da função.
- entre outras.

Universidade do Minho

Colónia de partículas

Universidade do Minho

O Paradigma da colónia de partículas (CP)

Os algoritmos baseados em colónia de partículas tentam imitar o comportamento social de uma população (colónia) de indivíduos (partículas).

O comportamento de um indivíduo é uma combinação da sua experiência passada (influência cognitiva) e da experiência da sociedade (influência social).

No contexto da optimização, uma partícula p, no instante t, é representada pela sua posição actual ($x^p(t)$), a sua melhor posição de sempre ($y^p(t)$) e uma velocidade de *viagem* ($v^p(t)$).

Nova posição e velocidade

A nova posição da partícula é actualizada por

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

onde $v^p(t+1)$ é a nova velocidade dada por

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

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

- $\iota(t)$ é o factor de inércia
- μ é o parâmetro *cognitivo* e ν é o parâmetro *social*
- $\omega_{1j}(t)$ e $\omega_{2j}(t)$ são números aleatórios obtidos da distribuição uniforme $(0, 1)$.

Nova posição e velocidade

A nova posição da partícula é actualizada por

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

onde $v^p(t+1)$ é a nova velocidade dada por

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

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

- $\iota(t)$ é o factor de inércia
- μ é o parâmetro *cognitivo* e ν é o parâmetro *social*
- $\omega_{1j}(t)$ e $\omega_{2j}(t)$ são números aleatórios obtidos da distribuição uniforme $(0, 1)$.

Nova posição e velocidade

A nova posição da partícula é actualizada por

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

onde $v^p(t+1)$ é a nova velocidade dada por

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

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

- $\iota(t)$ é o factor de inércia
- μ é o parâmetro *cognitivo* e ν é o parâmetro *social*
- $\omega_{1j}(t)$ e $\omega_{2j}(t)$ são números aleatórios obtidos da distribuição uniforme $(0, 1)$.

Nova posição e velocidade

A nova posição da partícula é actualizada por

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

onde $v^p(t+1)$ é a nova velocidade dada por

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

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

- $\iota(t)$ é o factor de inércia
- μ é o parâmetro *cognitivo* e ν é o parâmetro *social*
- $\omega_{1j}(t)$ e $\omega_{2j}(t)$ são números aleatórios obtidos da distribuição uniforme $(0, 1)$.

Universidade do Minho

A melhor partícula da colónia

$\hat{y}(t)$ é a posição da partícula com melhor valor de sempre da função objectivo, *i.e.*,

$$\hat{y}(t) = \arg\min_{a \in \mathcal{A}} f(a)$$

$$\mathcal{A} = \left\{ y^1(t), \ldots, y^s(t) \right\}.$$

onde s é o tamanho da população.

Universidade do Minho

A melhor partícula da colónia

$\hat{y}(t)$ é a posição da partícula com melhor valor de sempre da função objectivo, *i.e.*,

$$\hat{y}(t) = \arg\min_{a \in \mathcal{A}} f(a)$$

$$\mathcal{A} = \left\{ y^1(t), \ldots, y^s(t) \right\}.$$

onde s é o tamanho da população.

Do ponto de vista algorítmico apenas é necessário guardar o índice da partícula que obteve o melhor valor da função objectivo.

Tratamento das restrições

As restrições do tipo limite simples são tratadas pela projecção em $\Omega = \{x \in \mathbb{R}^n : \ell \leq x \leq u\}$, para todas as partículas $i = 1, \ldots, s$.

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

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

A projecção é aplicada as novas posições das partículas.

O algoritmo

1. Dado s e $v_{tol} > 0$. Inicializar $\{x^1(0), \ldots, x^s(0)\}$ e $\{v^1(0), \ldots, v^s(0)\}$.
2. $y^i(0) = x^i(0)$, $i = 1, \ldots, s$, e $\hat{y}(0) = \arg\min_{z \in \{y^1(0), \ldots, y^s(0)\}} f(z)$. $t = 0$.
3. $\hat{y}(t+1) = \hat{y}(t)$.
   Para $i = 1, \ldots, s$ fazer:
   - $\hat{x}^i(t) = proj_{\Omega}(x^i(t))$.
   - Se $f(\hat{x}^i(t)) < f(y^i(t))$ então
     - $y^i(t+1) = \hat{x}^i(t)$.
     - Se $f(y^i(t+1)) < f(\hat{y}(t+1))$ então $\hat{y}(t+1) = y^i(t+1)$.
   - Caso contrário fazer $y^i(t+1) = y^i(t)$.
4. Calcular $v^i(t+1)$ e $x^i(t+1)$, $i = 1, \ldots, s$.
5. Se $\|v^i(t+1)\| < v_{tol}$, para todo o $i = 1, \ldots, s$, então parar. Caso contrário, incrementar t em um e ir para o passo 3.

O algoritmo

1. Dado s e $v_{tol} > 0$. Inicializar $\{x^1(0), \ldots, x^s(0)\}$ e $\{v^1(0), \ldots, v^s(0)\}$.
2. $y^i(0) = x^i(0)$, $i = 1, \ldots, s$, e $\hat{y}(0) = \arg\min_{z \in \{y^1(0), \ldots, y^s(0)\}} f(z)$. $t = 0$.
3. $\hat{y}(t+1) = \hat{y}(t)$.
   Para $i = 1, \ldots, s$ fazer:
   - $\hat{x}^i(t) = proj_{\Omega}(x^i(t))$.
   - Se $f(\hat{x}^i(t)) < f(y^i(t))$ então
     - $y^i(t+1) = \hat{x}^i(t)$.
     - Se $f(y^i(t+1)) < f(\hat{y}(t+1))$ então $\hat{y}(t+1) = y^i(t+1)$.
   - Caso contrário fazer $y^i(t+1) = y^i(t)$.
4. Calcular $v^i(t+1)$ e $x^i(t+1)$, $i = 1, \ldots, s$.
5. Se $\|v^i(t+1)\| < v_{tol}$, para todo o $i = 1, \ldots, s$, então parar. Caso contrário, incrementar t em um e ir para o passo 3.

O algoritmo

1. Dado s e $v_{tol} > 0$. Inicializar $\{x^1(0), \ldots, x^s(0)\}$ e $\{v^1(0), \ldots, v^s(0)\}$.
2. $y^i(0) = x^i(0)$, $i = 1, \ldots, s$, e $\hat{y}(0) = \arg\min_{z \in \{y^1(0), \ldots, y^s(0)\}} f(z)$. $t = 0$.
3. $\hat{y}(t+1) = \hat{y}(t)$.
   Para $i = 1, \ldots, s$ fazer:
   - $\hat{x}^i(t) = proj_\Omega(x^i(t))$.
   - Se $f(\hat{x}^i(t)) < f(y^i(t))$ então
     - $y^i(t+1) = \hat{x}^i(t)$.
     - Se $f(y^i(t+1)) < f(\hat{y}(t+1))$ então $\hat{y}(t+1) = y^i(t+1)$.
   - Caso contrário fazer $y^i(t+1) = y^i(t)$.
4. Calcular $v^i(t+1)$ e $x^i(t+1)$, $i = 1, \ldots, s$.
5. Se $\|v^i(t+1)\| < v_{tol}$, para todo o $i = 1, \ldots, s$, então parar. Caso contrário, incrementar t em um e ir para o passo 3.

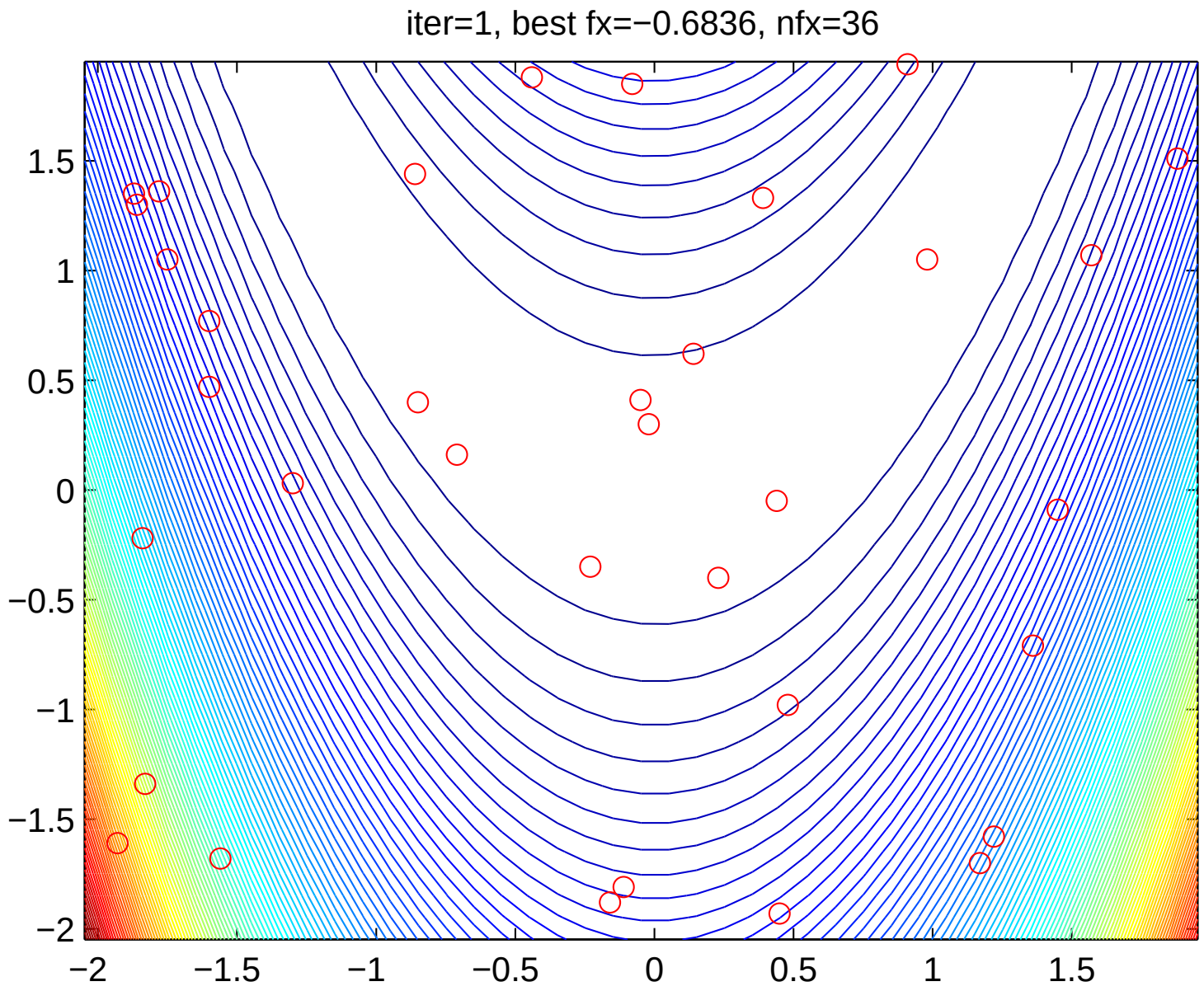
O algoritmo

1. Dado s e $v_{tol} > 0$. Inicializar $\{x^1(0), \ldots, x^s(0)\}$ e $\{v^1(0), \ldots, v^s(0)\}$.
2. $y^i(0) = x^i(0)$, $i = 1, \ldots, s$, e $\hat{y}(0) = \arg\min_{z \in \{y^1(0), \ldots, y^s(0)\}} f(z)$. $t = 0$.
3. $\hat{y}(t+1) = \hat{y}(t)$.
   Para $i = 1, \ldots, s$ fazer:
   - $\hat{x}^i(t) = proj_{\Omega}(x^i(t))$.
   - Se $f(\hat{x}^i(t)) < f(y^i(t))$ então
     - $y^i(t+1) = \hat{x}^i(t)$.
     - Se $f(y^i(t+1)) < f(\hat{y}(t+1))$ então $\hat{y}(t+1) = y^i(t+1)$.
   - Caso contrário fazer $y^i(t+1) = y^i(t)$.
4. Calcular $v^i(t+1)$ e $x^i(t+1)$, $i = 1, \ldots, s$.
5. Se $\|v^i(t+1)\| < v_{tol}$, para todo o $i = 1, \ldots, s$, então parar. Caso contrário, incrementar t em um e ir para o passo 3.

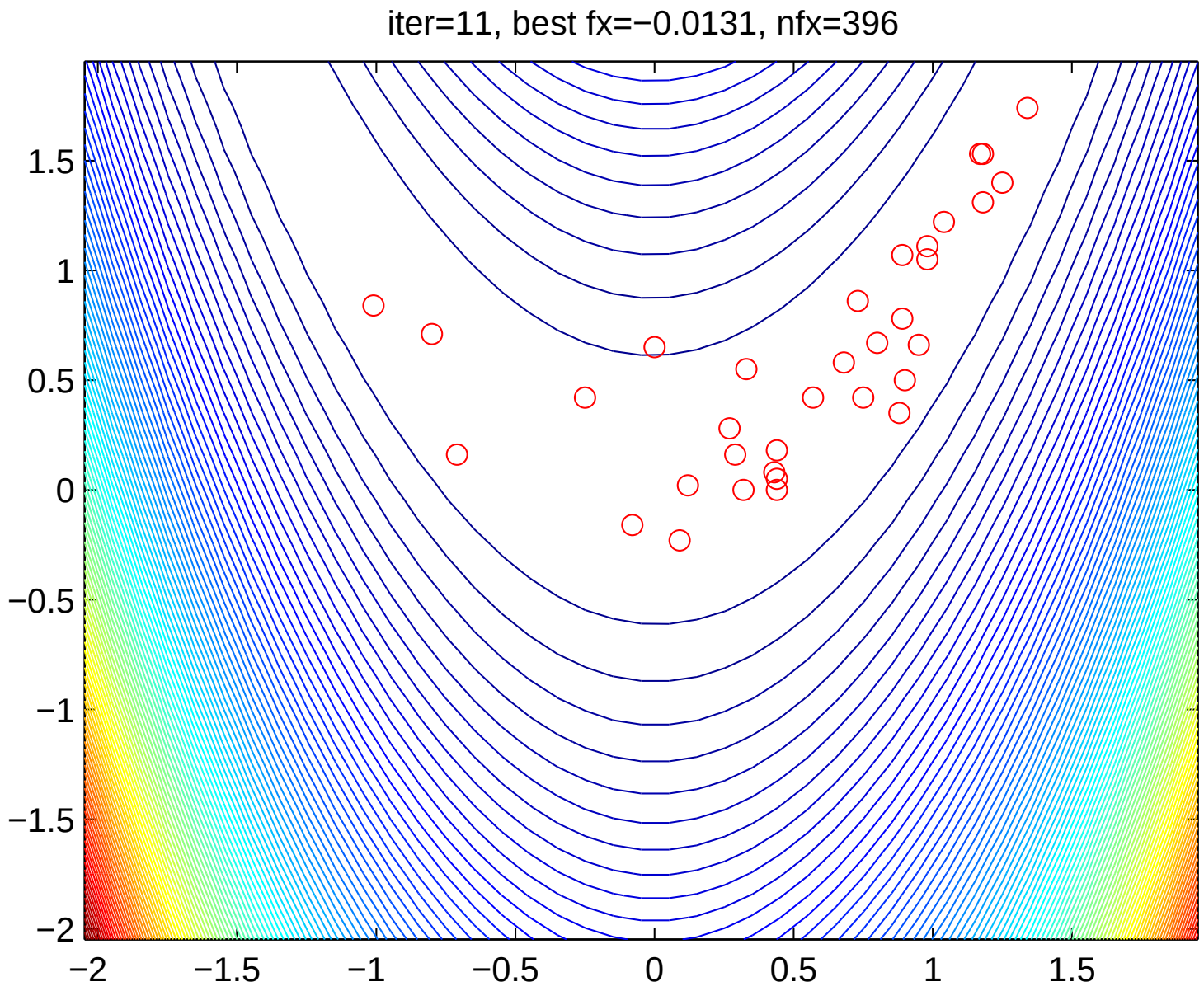
O algoritmo

1. Dado s e $v_{tol} > 0$. Inicializar $\{x^1(0), \ldots, x^s(0)\}$ e $\{v^1(0), \ldots, v^s(0)\}$.
2. $y^i(0) = x^i(0)$, $i = 1, \ldots, s$, e $\hat{y}(0) = \arg\min_{z \in \{y^1(0), \ldots, y^s(0)\}} f(z)$. $t = 0$.
3. $\hat{y}(t+1) = \hat{y}(t)$.
   Para $i = 1, \ldots, s$ fazer:
   - $\hat{x}^i(t) = proj_{\Omega}(x^i(t))$.
   - Se $f(\hat{x}^i(t)) < f(y^i(t))$ então
     - $y^i(t+1) = \hat{x}^i(t)$.
     - Se $f(y^i(t+1)) < f(\hat{y}(t+1))$ então $\hat{y}(t+1) = y^i(t+1)$.
   - Caso contrário fazer $y^i(t+1) = y^i(t)$.
4. Calcular $v^i(t+1)$ e $x^i(t+1)$, $i = 1, \ldots, s$.
5. Se $\|v^i(t+1)\| < v_{tol}$, para todo o $i = 1, \ldots, s$, então parar. Caso contrário, incrementar t em um e ir para o passo 3.

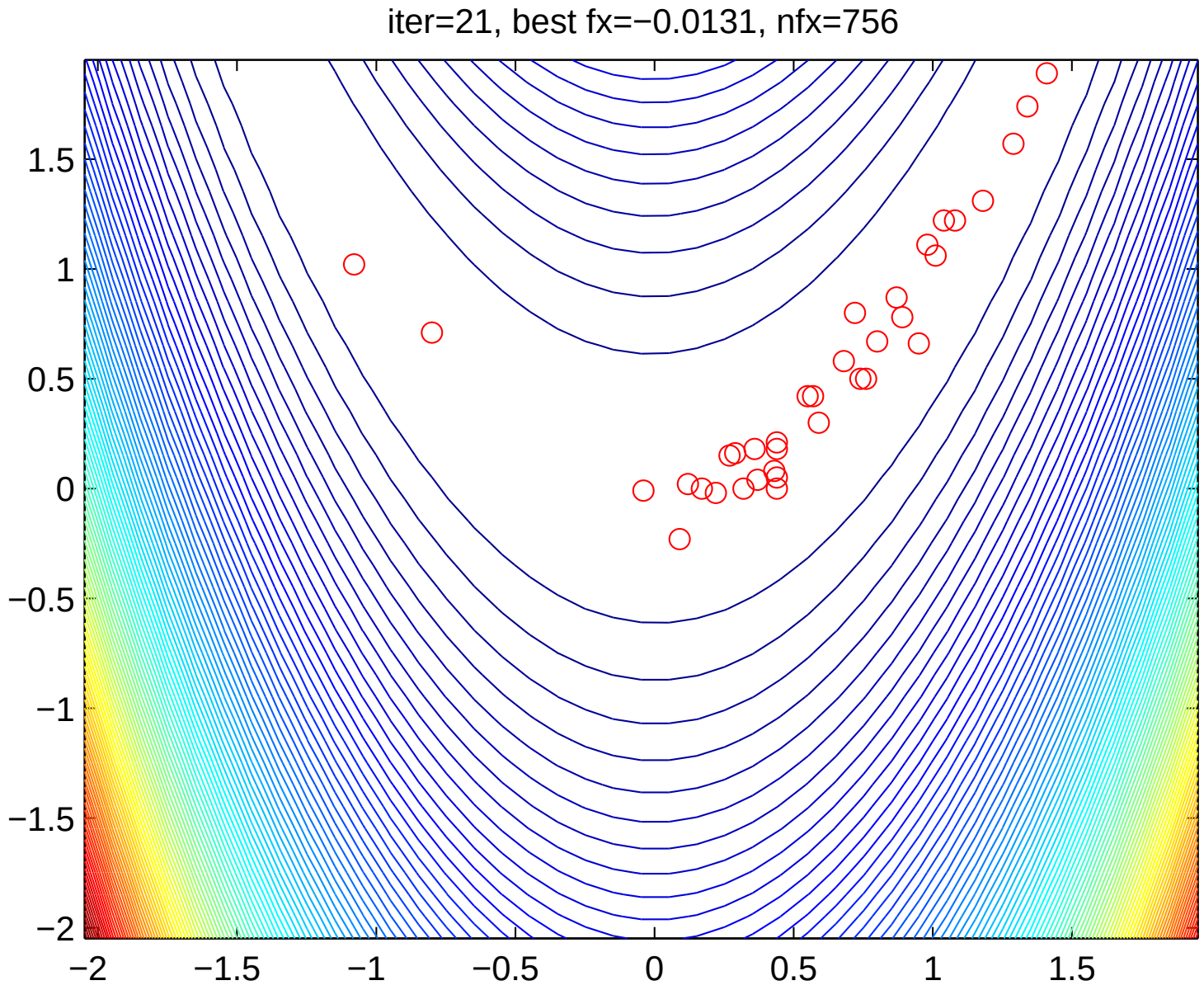
Exemplo com o problema `ir2.mod`



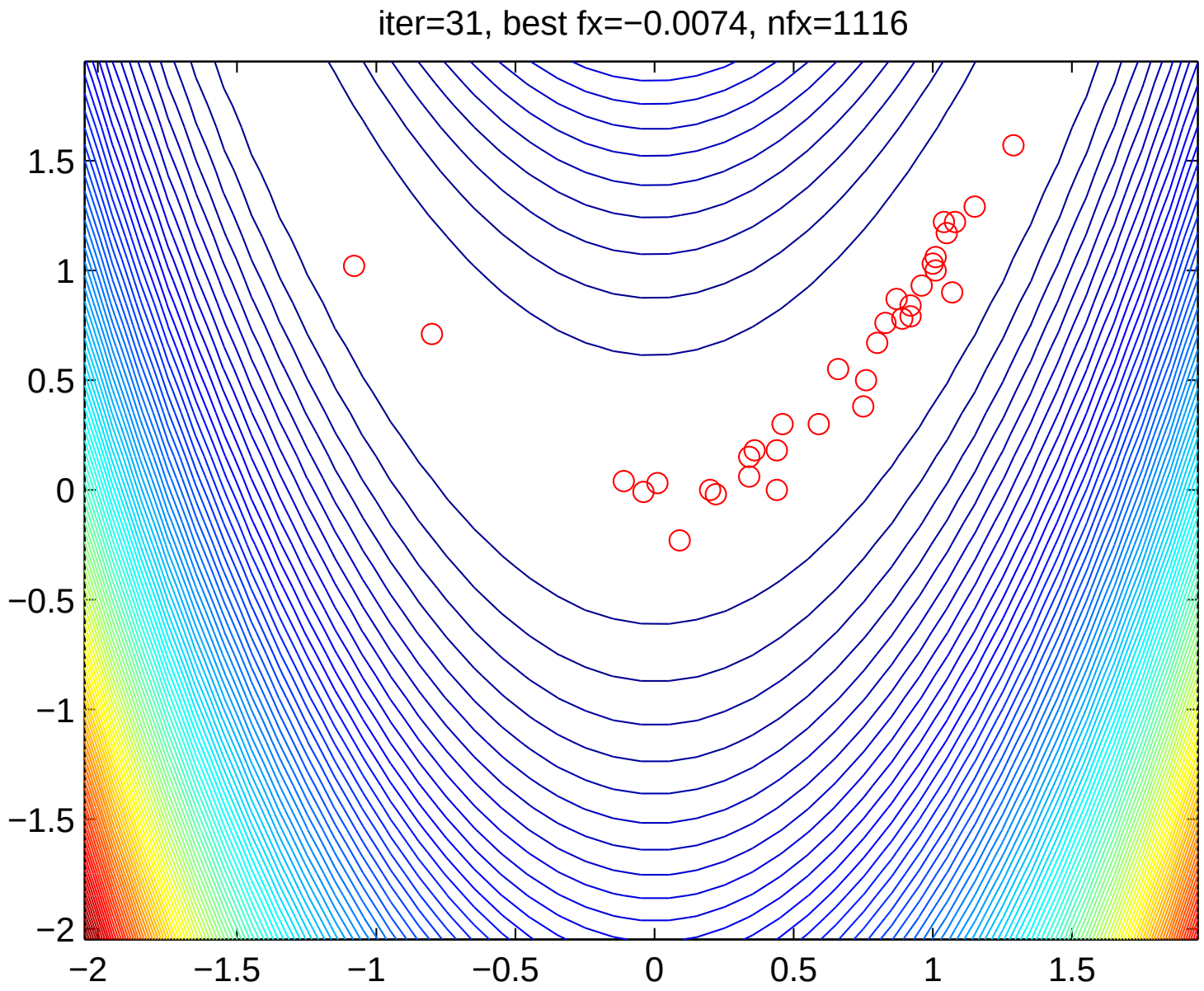
Exemplo



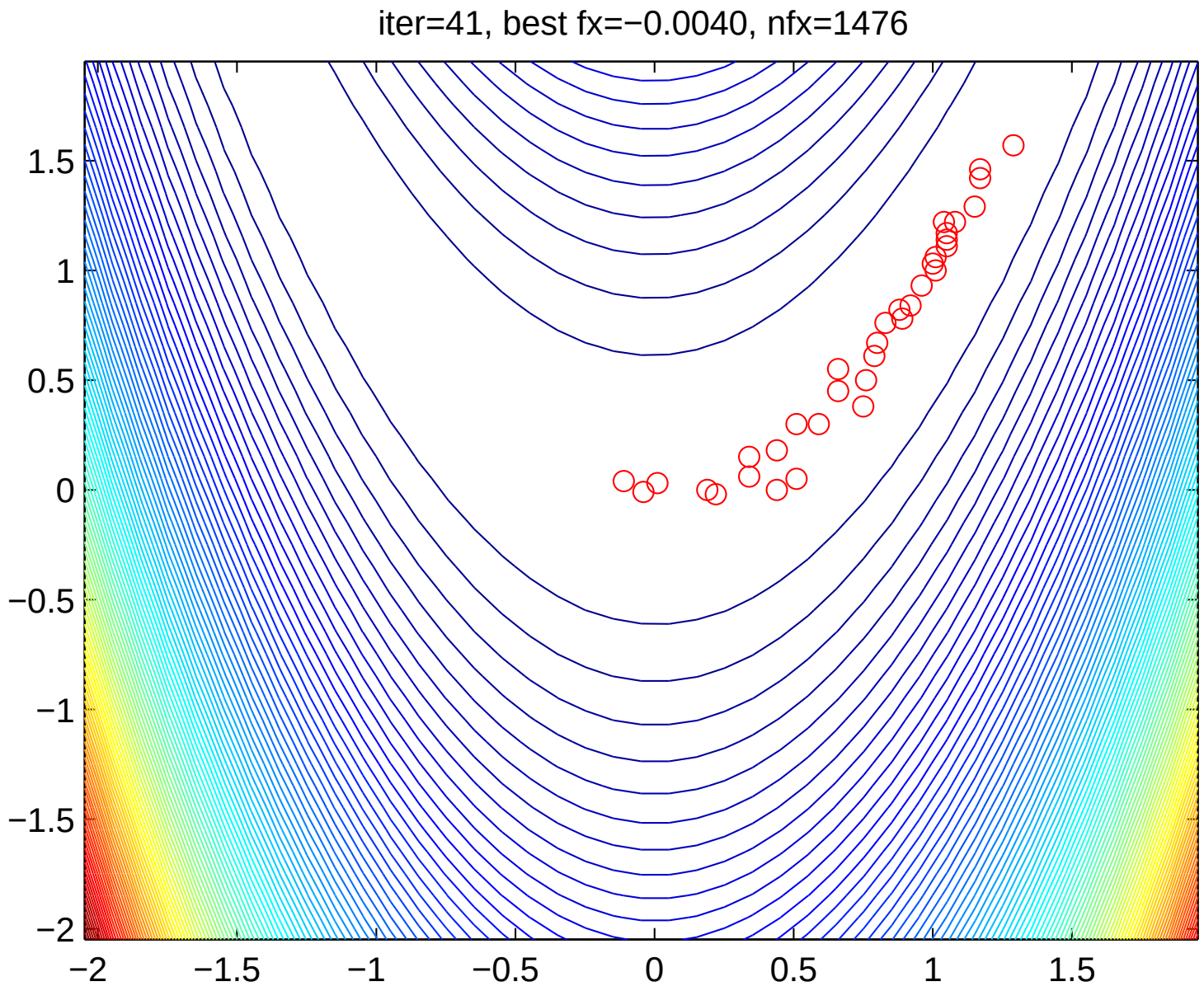
Exemplo



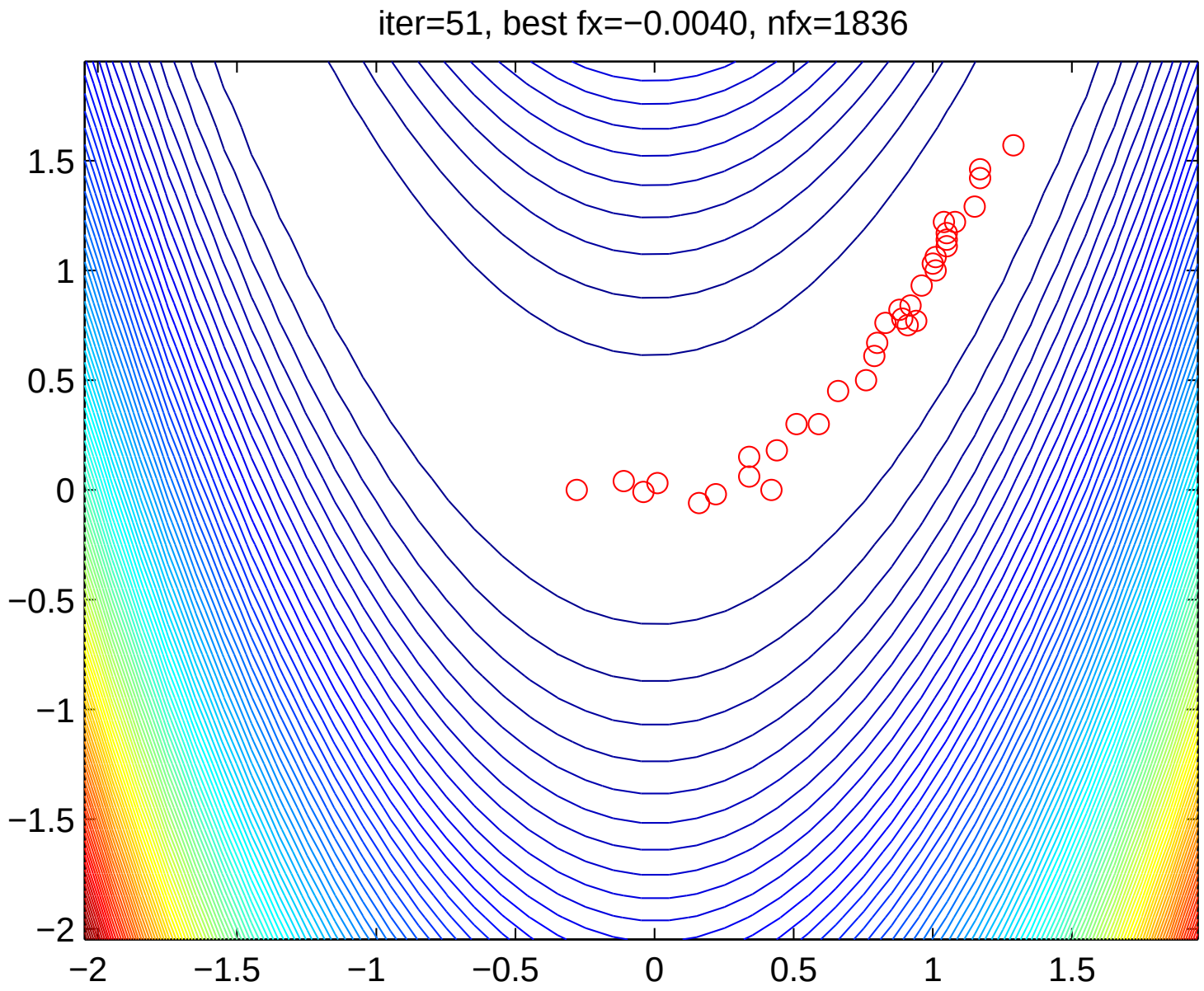
Exemplo



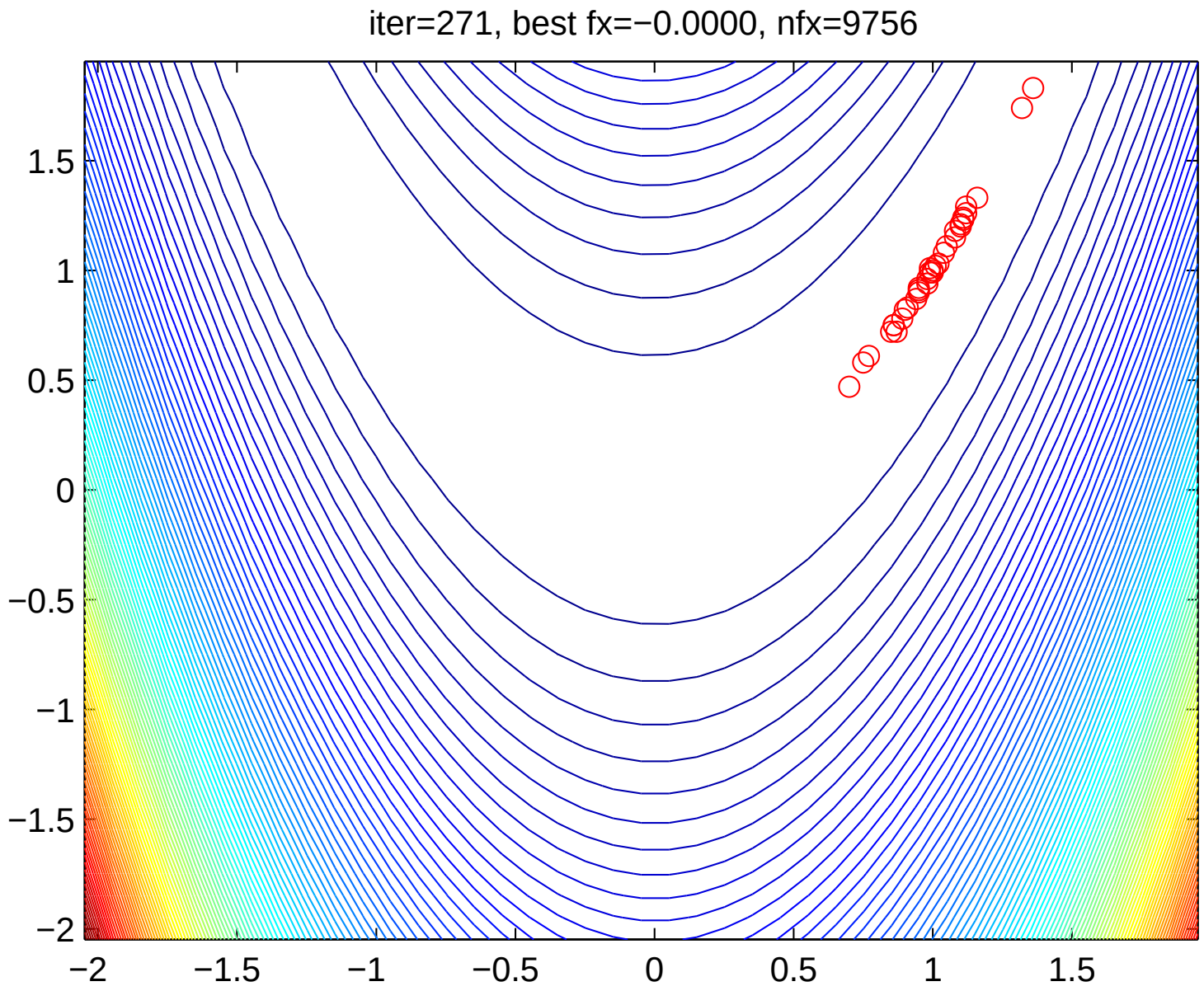
Exemplo



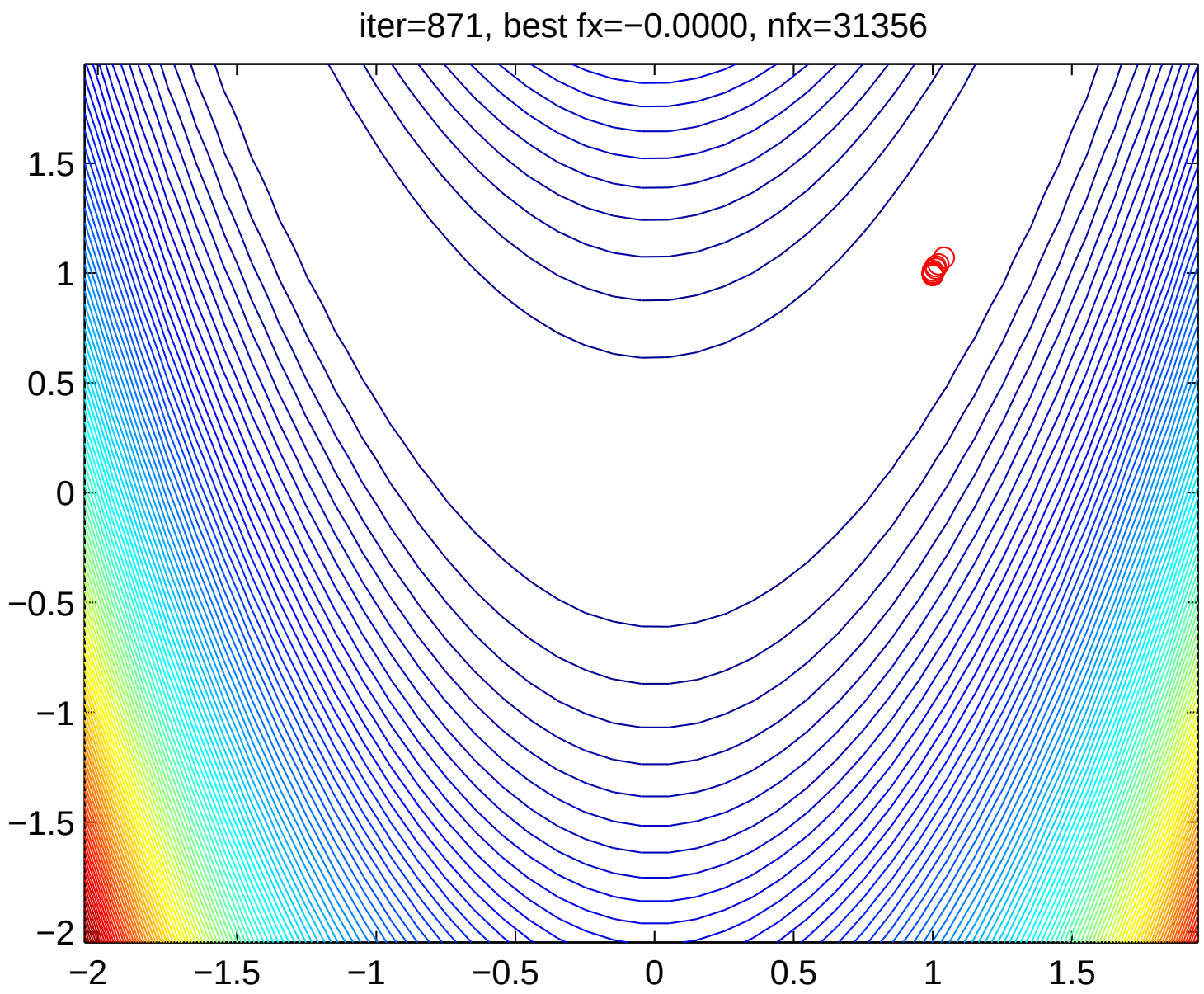
Exemplo



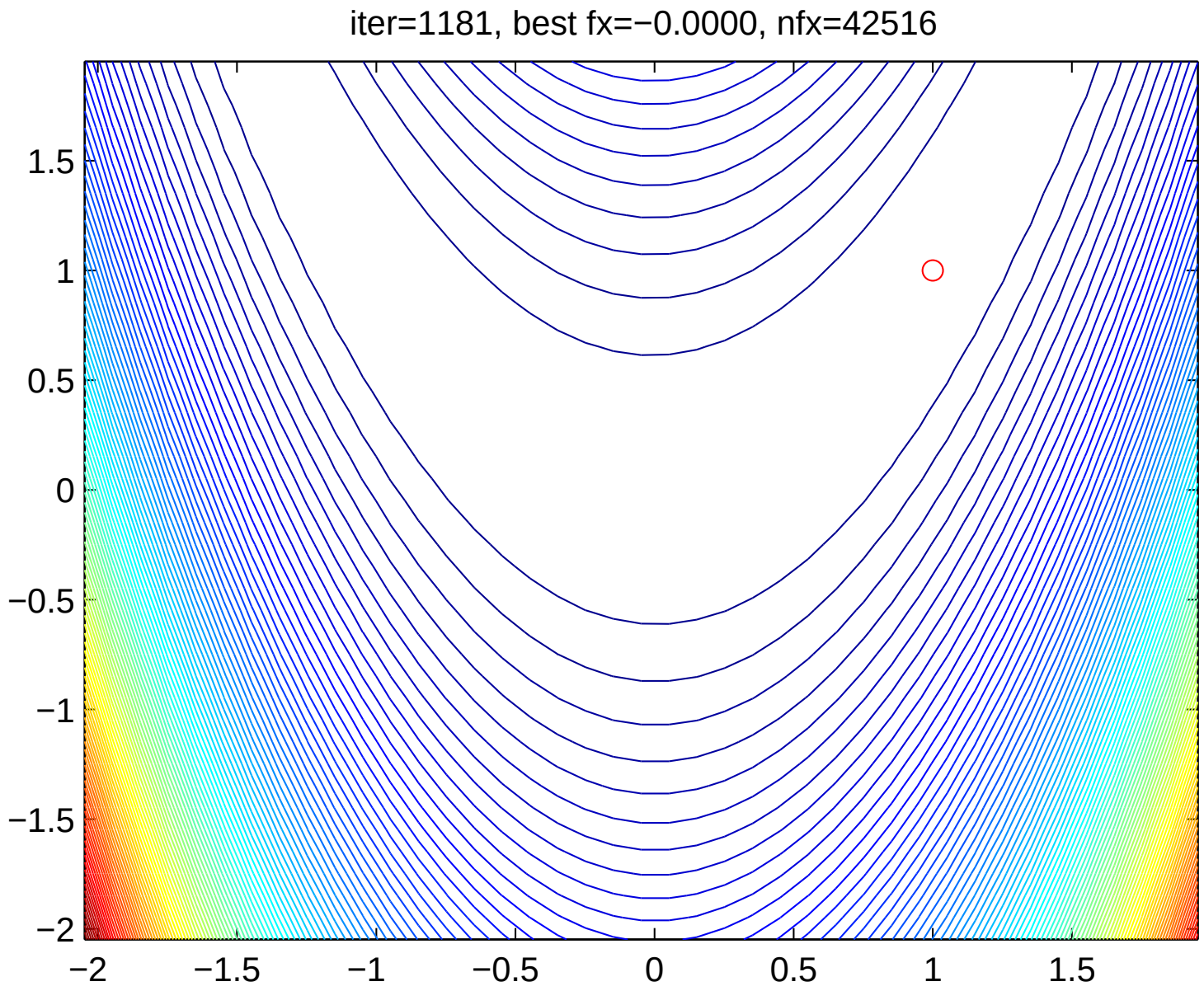
Exemplo



Exemplo



Exemplo



Universidade do Minho

Algumas propriedades

- Simples de implementar.
- Facilmente adaptável para variáveis discretas.
- Facilmente paralelizável.
- Para uma escolha correcta dos parâmetros o algoritmo termina ($\lim_{t \to +\infty} v(t) = 0$).
- Usa apenas cálculos da função objectivo (sem cálculo de derivadas ou aproximações).
- Convergência para um óptimo global, mas muito lento próximo do óptimo.
- Número elevado de cálculos da função objectivo.

Universidade do Minho

Algumas propriedades

- Simples de implementar.
- Facilmente adaptável para variáveis discretas.
- Facilmente paralelizável.
- Para uma escolha correcta dos parâmetros o algoritmo termina ($\lim_{t\to+\infty} v(t) = 0$).
- Usa apenas cálculos da função objectivo (sem cálculo de derivadas ou aproximações).
- Convergência para um óptimo global, mas muito lento próximo do óptimo.
- Número elevado de cálculos da função objectivo.

Universidade do Minho

Algumas propriedades

- Simples de implementar.
- Facilmente adaptável para variáveis discretas.
- Facilmente paralelizável.
- Para uma escolha correcta dos parâmetros o algoritmo termina ($\lim_{t \to +\infty} v(t) = 0$).
- Usa apenas cálculos da função objectivo (sem cálculo de derivadas ou aproximações).
- Convergência para um óptimo global, mas muito lento próximo do óptimo.
- Número elevado de cálculos da função objectivo.

Algumas propriedades

- Simples de implementar.
- Facilmente adaptável para variáveis discretas.
- Facilmente paralelizável.
- Para uma escolha correcta dos parâmetros o algoritmo termina ($\lim_{t \to +\infty} v(t) = 0$).
- Usa apenas cálculos da função objectivo (sem cálculo de derivadas ou aproximações).
- Convergência para um óptimo global, mas muito lento próximo do óptimo.
- Número elevado de cálculos da função objectivo.

Universidade do Minho

Algumas propriedades

- Simples de implementar.
- Facilmente adaptável para variáveis discretas.
- Facilmente paralelizável.
- Para uma escolha correcta dos parâmetros o algoritmo termina ($\lim_{t \to +\infty} v(t) = 0$).
- Usa apenas cálculos da função objectivo (sem cálculo de derivadas ou aproximações).
- Convergência para um óptimo global, mas muito lento próximo do óptimo.
- Número elevado de cálculos da função objectivo.

Universidade do Minho

Algumas propriedades

- Simples de implementar.
- Facilmente adaptável para variáveis discretas.
- Facilmente paralelizável.
- Para uma escolha correcta dos parâmetros o algoritmo termina ($\lim_{t\to+\infty} v(t) = 0$).
- Usa apenas cálculos da função objectivo (sem cálculo de derivadas ou aproximações).
- Convergência para um óptimo global, mas muito lento próximo do óptimo.
- Número elevado de cálculos da função objectivo.

Universidade do Minho

Algumas propriedades

- Simples de implementar.
- Facilmente adaptável para variáveis discretas.
- Facilmente paralelizável.
- Para uma escolha correcta dos parâmetros o algoritmo termina ($\lim_{t\to+\infty} v(t) = 0$).
- Usa apenas cálculos da função objectivo (sem cálculo de derivadas ou aproximações).
- Convergência para um óptimo global, mas muito lento próximo do óptimo.
- Número elevado de cálculos da função objectivo.

Universidade do Minho

Procura em padrão

Universidade do Minho

Introdução aos métodos directos

Os métodos de procura directa são uma classes importante de algoritmos de optimização que procuram minimizar uma função através da comparação dos valores da função objectivo num número finito de pontos.

Universidade do Minho

Introdução aos métodos directos

Os métodos de procura directa são uma classes importante de algoritmos de optimização que procuram minimizar uma função através da comparação dos valores da função objectivo num número finito de pontos.

Os métodos de procura directa não recorrem às derivadas da função objectivo nem as tentam aproximar.

Universidade do Minho

Introdução aos métodos directos

Os métodos de procura directa são uma classes importante de algoritmos de optimização que procuram minimizar uma função através da comparação dos valores da função objectivo num número finito de pontos.

Os métodos de procura directa não recorrem às derivadas da função objectivo nem as tentam aproximar.

O método de procura em padrão pertence à classe de métodos de procura directa, sendo a sua estrutura mais rígida.

Universidade do Minho

Introdução aos métodos directos

Os métodos de procura directa são uma classes importante de algoritmos de optimização que procuram minimizar uma função através da comparação dos valores da função objectivo num número finito de pontos.

Os métodos de procura directa não recorrem às derivadas da função objectivo nem as tentam aproximar.

O método de procura em padrão pertence à classe de métodos de procura directa, sendo a sua estrutura mais rígida.

Um dos conceitos mais importantes na definição da procura em padrão é a de bases geradoras positivas (*positive spanning sets*).

Universidade do Minho

Bases geradoras positivas

O que é uma base geradora (positiva) de $\mathbb{R}^n$?

Universidade do Minho

Bases geradoras positivas

O que é uma base geradora (positiva) de $\mathbb{R}^n$?

É um conjunto de vectores que geram todo o espaço ($\mathbb{R}^n$), *i.e.*, todos os pontos do espaço são uma combinação linear (com coeficientes positivos) dos vectores da base.

Bases geradoras positivas

O que é uma base geradora (positiva) de $\mathbb{R}^n$?

É um conjunto de vectores que geram todo o espaço ($\mathbb{R}^n$), *i.e.*, todos os pontos do espaço são uma combinação linear (com coeficientes positivos) dos vectores da base.

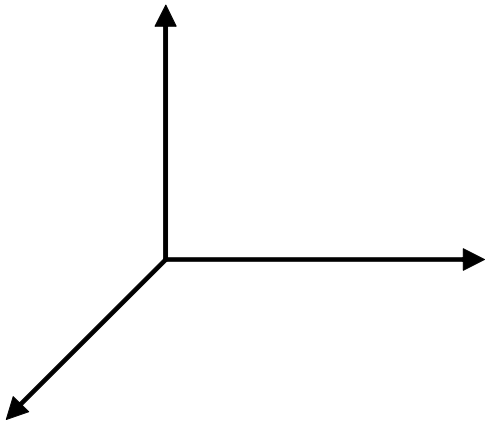
Exemplo para $\mathbb{R}^2$

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

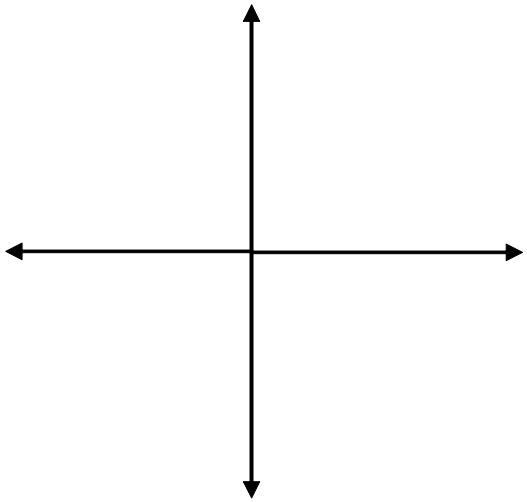
$$\begin{pmatrix} 5 \\ -2 \end{pmatrix} = 9 \begin{pmatrix} 1 \\ 0 \end{pmatrix} + 2 \begin{pmatrix} 0 \\ 1 \end{pmatrix} + 4 \begin{pmatrix} -1 \\ -1 \end{pmatrix}$$

$$= 8 \begin{pmatrix} 1 \\ 0 \end{pmatrix} + 1 \begin{pmatrix} 0 \\ 1 \end{pmatrix} + 3 \begin{pmatrix} -1 \\ -1 \end{pmatrix}$$

Tipos de bases



Base Minimal
com $n+1$ vectores
(3 no caso de $\mathbb{R}^2$).



Base Maximal
com $2n$ vectores
(4 no caso de $\mathbb{R}^2$).

Algumas definições

Seja

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

uma base positiva (maximal).

Universidade do Minho

Algumas definições

Seja

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

uma base positiva (maximal).

O método directo elementar baseado neste conjunto é conhecido por procura coordenada ou compasso (*coordinate or compass search*).

Universidade do Minho

Algumas definições

Seja

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

uma base positiva (maximal).

O método directo elementar baseado neste conjunto é conhecido por procura coordenada ou compasso (*coordinate or compass search*).

Dado um conjunto gerador D e o iterando corrente $y(t)$ define-se dois conjuntos de pontos: a grelha M_t e o conjunto de sondagem P_t. A grelha M_t é dada por

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

onde $\alpha(t) > 0$ é o parâmetro do tamanho da grelha. O conjunto de sondagem é dado por

$$P_t \ = \ \{ y(t) + \alpha(t) d, \ d \in D \}.$$

Universidade do Minho

Procura em padrão

O passo de procura (*search step*) executa uma procura finita na grelha M_t.

Universidade do Minho

Procura em padrão

O passo de procura (*search step*) executa uma procura finita na grelha M_t.

Se a fase de procura não obtiver sucesso na obtenção de um iterando melhor então segue-se uma fase de sondagem.

Universidade do Minho

Procura em padrão

O passo de procura (*search step*) executa uma procura finita na grelha M_t.

Se a fase de procura não obtiver sucesso na obtenção de um iterando melhor então segue-se uma fase de sondagem.

A fase de sondagem avalia a função nos pontos de P_t na procura de um ponto que tenha menor valor da função objectivo.
Se o passo de sondagem tiver sucesso o valor de $\alpha(t)$ pode ser expandido, caso contrário é contraído.

Universidade do Minho

Tratamento das restrições

No método procura coordenada é suficiente iniciar o algoritmo com uma aproximação inicial admissível ($y(0) \in \Omega$) e usar a função $\hat{f}$ como função objectivo.

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

O algoritmo

1. Dado $\alpha_{tol} > 0$, $D = D_{\oplus}$, $\alpha(0) > 0$ e $y(0)$ (admissível). $t = 0$.
2. **[Passo de procura]**
3. **[Passo de sondagem]**
   O passo de sondagem é ignorado se o passo de procura teve sucesso.
   - Se existe $d(t) \in D$ tal que $\hat{f}(y(t) + \alpha(t)d(t)) < \hat{f}(y(t))$ então
   - Caso contrário, $\hat{f}(y(t) + \alpha(t)d(t)) \geq \hat{f}(y(t))$ para todo o $d(t) \in D$ e
4. Se $\alpha(t+1) < \alpha_{tol}$ então pára-se. Caso contrário, $t = t + 1$ e vai-se para o Passo 2.

Universidade do Minho

O algoritmo

1. Dado $\alpha_{tol} > 0$, $D = D_{\oplus}$, $\alpha(0) > 0$ e $y(0)$ (admissível). $t = 0$.
2. **[Passo de procura]**
   Avalia-se f num conjunto finito de pontos de M_t. Se um ponto $z(t) \in M_t$ for encontrado tal que $\hat{f}(z(t)) < \hat{f}(y(t))$ então faz-se $y(t+1) = z(t)$, $\alpha(t+1) = \phi(t)\alpha(t)$ (expansão — sucesso).
3. **[Passo de sondagem]**
   O passo de sondagem é ignorado se o passo de procura teve sucesso.
   - Se existe $d(t) \in D$ tal que $\hat{f}(y(t) + \alpha(t)d(t)) < \hat{f}(y(t))$ então
   - Caso contrário, $\hat{f}(y(t) + \alpha(t)d(t)) \geq \hat{f}(y(t))$ para todo o $d(t) \in D$ e
4. Se $\alpha(t+1) < \alpha_{tol}$ então pára-se. Caso contrário, $t = t + 1$ e vai-se para o Passo 2.

O algoritmo

1. Dado $\alpha_{tol} > 0$, $D = D_{\oplus}$, $\alpha(0) > 0$ e $y(0)$ (admissível). $t = 0$.
2. **[Passo de procura]**
3. **[Passo de sondagem]**
   O passo de sondagem é ignorado se o passo de procura teve sucesso.
   - Se existe $d(t) \in D$ tal que $\hat{f}(y(t) + \alpha(t)d(t)) < \hat{f}(y(t))$ então
   - Caso contrário, $\hat{f}(y(t) + \alpha(t)d(t)) \geq \hat{f}(y(t))$ para todo o $d(t) \in D$ e
4. Se $\alpha(t+1) < \alpha_{tol}$ então pára-se. Caso contrário, $t = t + 1$ e vai-se para o Passo 2.

O algoritmo

1. Dado $\alpha_{tol} > 0$, $D = D_{\oplus}$, $\alpha(0) > 0$ e $y(0)$ (admissível). $t = 0$.
2. **[Passo de procura]**
3. **[Passo de sondagem]**
   O passo de sondagem é ignorado se o passo de procura teve sucesso.
   - Se existe $d(t) \in D$ tal que $\hat{f}(y(t) + \alpha(t)d(t)) < \hat{f}(y(t))$ então
     - $y(t+1) = y(t) + \alpha(t)d(t)$ (sucesso).
     - $\alpha(t+1) = \phi(t)\alpha(t)$ (expansão).
   - Caso contrário, $\hat{f}(y(t) + \alpha(t)d(t)) \geq \hat{f}(y(t))$ para todo o $d(t) \in D$ e
4. Se $\alpha(t+1) < \alpha_{tol}$ então pára-se. Caso contrário, $t = t + 1$ e vai-se para o Passo 2.

Universidade do Minho

O algoritmo

1. Dado $\alpha_{tol} > 0$, $D = D_{\oplus}$, $\alpha(0) > 0$ e $y(0)$ (admissível). $t = 0$.
2. **[Passo de procura]**
3. **[Passo de sondagem]**
   O passo de sondagem é ignorado se o passo de procura teve sucesso.
   - Se existe $d(t) \in D$ tal que $\hat{f}(y(t) + \alpha(t)d(t)) < \hat{f}(y(t))$ então
   - Caso contrário, $\hat{f}(y(t) + \alpha(t)d(t)) \geq \hat{f}(y(t))$ para todo o $d(t) \in D$ e
     - $y(t+1) = y(t)$ (insucesso).
     - $\alpha(t+1) = \theta(t)\alpha(t)$ (contracção).
4. Se $\alpha(t+1) < \alpha_{tol}$ então pára-se. Caso contrário, $t = t + 1$ e vai-se para o Passo 2.

O algoritmo

1. Dado $\alpha_{tol} > 0$, $D = D_{\oplus}$, $\alpha(0) > 0$ e $y(0)$ (admissível). $t = 0$.
2. **[Passo de procura]**
3. **[Passo de sondagem]**
   O passo de sondagem é ignorado se o passo de procura teve sucesso.
   - Se existe $d(t) \in D$ tal que $\hat{f}(y(t) + \alpha(t)d(t)) < \hat{f}(y(t))$ então
   - Caso contrário, $\hat{f}(y(t) + \alpha(t)d(t)) \geq \hat{f}(y(t))$ para todo o $d(t) \in D$ e
4. Se $\alpha(t+1) < \alpha_{tol}$ então pára-se. Caso contrário, $t = t + 1$ e vai-se para o Passo 2.

Universidade do Minho

O algoritmo híbrido

Universidade do Minho

Motivação

O algoritmo híbrido procura juntar o melhor dos dois algoritmos.

A capacidade das colónias de partículas em determinar o óptimo global.

A garantia de obtenção de um ponto estacionário da procura em padrão.

Ideia central: Aplicar o algoritmo da colónia de partículas no passo de procura e quando este não obtiver sucesso aplicar o passo de sondagem.

O algoritmo híbrido

1. Dados $\alpha_{tol} > 0$, $D = D_{\oplus}$, $\alpha(0) > 0$, s, $v_{tol} > 0$. Inicializar $\{x^1(0), \ldots, x^s(0)\}$ e $\{v^1(0), \ldots, v^s(0)\}$. $t = 0$.
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. **[Passo de procura]**
   $\hat{y}(t+1) = \hat{y}(t)$.
   Para $i = 1, \ldots, s$ faz-se:
   - $\hat{x}^i(t) = proj_{M_t}(x^i(t))$.
   - Se $\hat{f}(\hat{x}^i(t)) < \hat{f}(y^i(t))$ então
     - $y^i(t+1) = \hat{x}^i(t)$.
     - $f(y^i(t+1)) < f(\hat{y}(t+1))$ então
       - $\hat{y}(t+1) = y^i(t+1)$ (sucesso).
       - $\alpha(t+1) = \phi(t)\alpha(t)$ (expansão).
   - Caso contrário $y^i(t+1) = y^i(t)$.

O algoritmo híbrido

1. Dados $\alpha_{tol} > 0$, $D = D_{\oplus}$, $\alpha(0) > 0$, s, $v_{tol} > 0$. Inicializar $\{x^1(0), \ldots, x^s(0)\}$ e $\{v^1(0), \ldots, v^s(0)\}$. $t = 0$.
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. **[Passo de procura]**
   $\hat{y}(t+1) = \hat{y}(t)$.
   Para $i = 1, \ldots, s$ faz-se:
   - $\hat{x}^i(t) = proj_{M_t}(x^i(t))$.
   - Se $\hat{f}(\hat{x}^i(t)) < \hat{f}(y^i(t))$ então
     - $y^i(t+1) = \hat{x}^i(t)$.
     - $f(y^i(t+1)) < f(\hat{y}(t+1))$ então
       - $\hat{y}(t+1) = y^i(t+1)$ (sucesso).
       - $\alpha(t+1) = \phi(t)\alpha(t)$ (expansão).
   - Caso contrário $y^i(t+1) = y^i(t)$.

O algoritmo híbrido

1. Dados $\alpha_{tol} > 0$, $D = D_{\oplus}$, $\alpha(0) > 0$, s, $v_{tol} > 0$. Inicializar $\{x^1(0), \ldots, x^s(0)\}$ e $\{v^1(0), \ldots, v^s(0)\}$. $t = 0$.
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. **[Passo de procura]**
   $\hat{y}(t+1) = \hat{y}(t)$.
   Para $i = 1, \ldots, s$ faz-se:
   - $\hat{x}^i(t) = proj_{M_t}(x^i(t))$.
   - Se $\hat{f}(\hat{x}^i(t)) < \hat{f}(y^i(t))$ então
     - $y^i(t+1) = \hat{x}^i(t)$.
     - $f(y^i(t+1)) < f(\hat{y}(t+1))$ então
       - $\hat{y}(t+1) = y^i(t+1)$ (sucesso).
       - $\alpha(t+1) = \phi(t)\alpha(t)$ (expansão).
   - Caso contrário $y^i(t+1) = y^i(t)$.

O algoritmo híbrido

4. **[Poll Step]**
   Saltar o passo de sondagem se o passo de procura obteve sucesso.
   - Se existe $d(t) \in D$ tal que $\hat{f}(\hat{y}(t) + \alpha(t)d(t)) < \hat{f}(\hat{y}(t))$ então
     - $\hat{y}(t+1) = \hat{y}(t) + \alpha(t)d(t)$ (sucesso).
     - $\alpha(t+1) = \phi(t)\alpha(t)$ (expansão).
   - Caso contrário, $\hat{f}(\hat{y}(t) + \alpha(t)d(t)) \geq \hat{f}(\hat{y}(t))$ para todo o $d(t) \in D$, e
     - $\hat{y}(t+1) = \hat{y}(t)$ (insucesso).
     - $\alpha(t+1) = \theta(t)\alpha(t)$ (contracção).
5. Calcula-se $v^i(t+1)$ e $x^i(t+1)$, $i = 1, \ldots, s$.
6. Se $\alpha(t+1) < \alpha_{tol}$ e $\|v^i(t+1)\| < v_{tol}$, para todo $i = 1, \ldots, s$, então pára-se. Caso contrário, $t = t+1$ e vai-se para o Passo 3.

O algoritmo híbrido

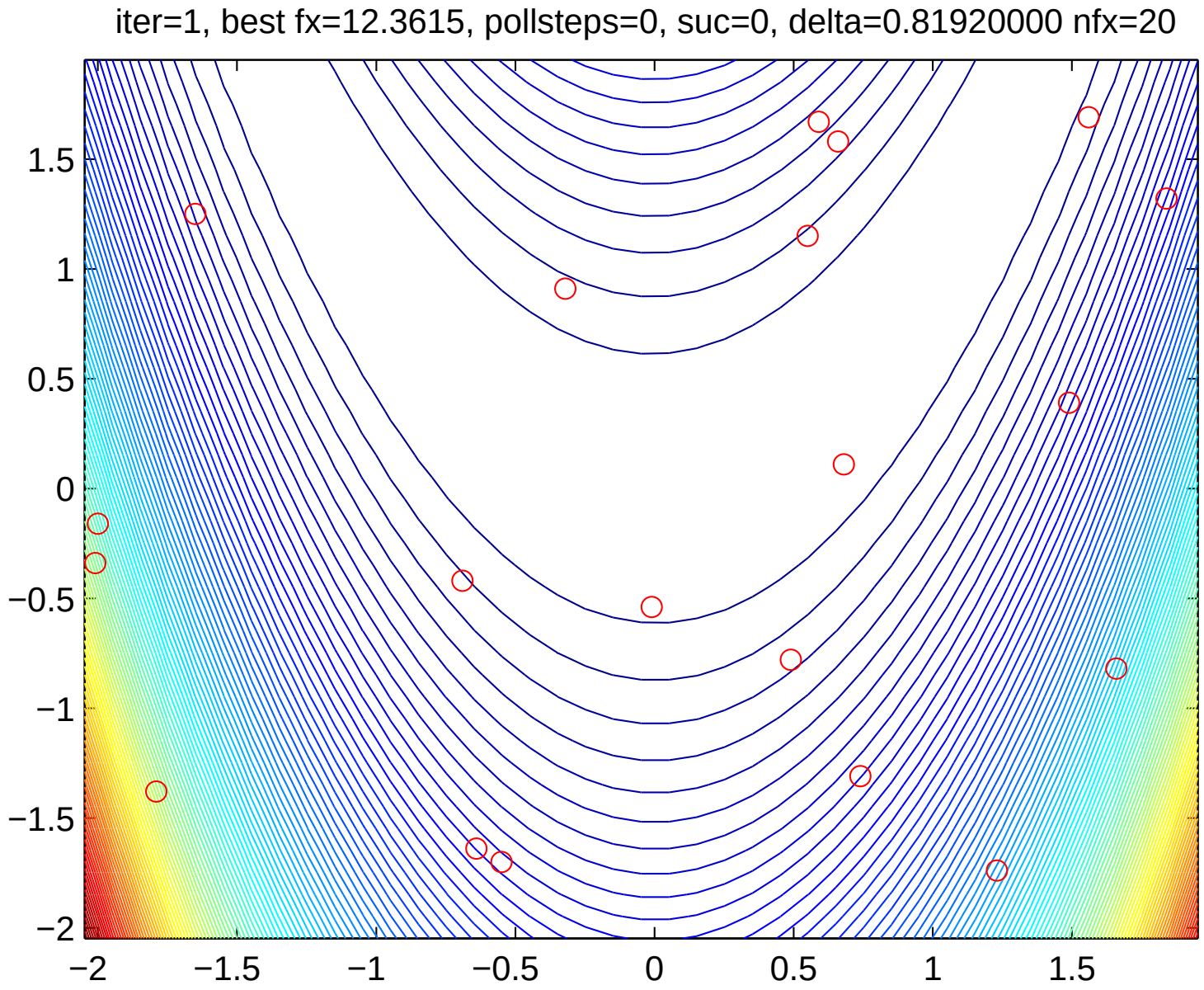
4. **[Poll Step]**
   Saltar o passo de sondagem se o passo de procura obteve sucesso.
   - Se existe $d(t) \in D$ tal que $\hat{f}(\hat{y}(t) + \alpha(t)d(t)) < \hat{f}(\hat{y}(t))$ então
     - $\hat{y}(t+1) = \hat{y}(t) + \alpha(t)d(t)$ (sucesso).
     - $\alpha(t+1) = \phi(t)\alpha(t)$ (expansão).
   - Caso contrário, $\hat{f}(\hat{y}(t) + \alpha(t)d(t)) \geq \hat{f}(\hat{y}(t))$ para todo o $d(t) \in D$, e
     - $\hat{y}(t+1) = \hat{y}(t)$ (insucesso).
     - $\alpha(t+1) = \theta(t)\alpha(t)$ (contracção).
5. Calcula-se $v^i(t+1)$ e $x^i(t+1)$, $i = 1, \ldots, s$.
6. Se $\alpha(t+1) < \alpha_{tol}$ e $\|v^i(t+1)\| < v_{tol}$, para todo $i = 1, \ldots, s$, então pára-se. Caso contrário, $t = t + 1$ e vai-se para o Passo 3.

O algoritmo híbrido

4. **[Poll Step]**
   Saltar o passo de sondagem se o passo de procura obteve sucesso.
   - Se existe $d(t) \in D$ tal que $\hat{f}(\hat{y}(t) + \alpha(t)d(t)) < \hat{f}(\hat{y}(t))$ então
     - $\hat{y}(t+1) = \hat{y}(t) + \alpha(t)d(t)$ (sucesso).
     - $\alpha(t+1) = \phi(t)\alpha(t)$ (expansão).
   - Caso contrário, $\hat{f}(\hat{y}(t) + \alpha(t)d(t)) \geq \hat{f}(\hat{y}(t))$ para todo o $d(t) \in D$, e
     - $\hat{y}(t+1) = \hat{y}(t)$ (insucesso).
     - $\alpha(t+1) = \theta(t)\alpha(t)$ (contracção).
5. Calcula-se $v^i(t+1)$ e $x^i(t+1)$, $i = 1, \ldots, s$.
6. Se $\alpha(t+1) < \alpha_{tol}$ e $\|v^i(t+1)\| < v_{tol}$, para todo $i = 1, \ldots, s$, então pára-se. Caso contrário, $t = t + 1$ e vai-se para o Passo 3.

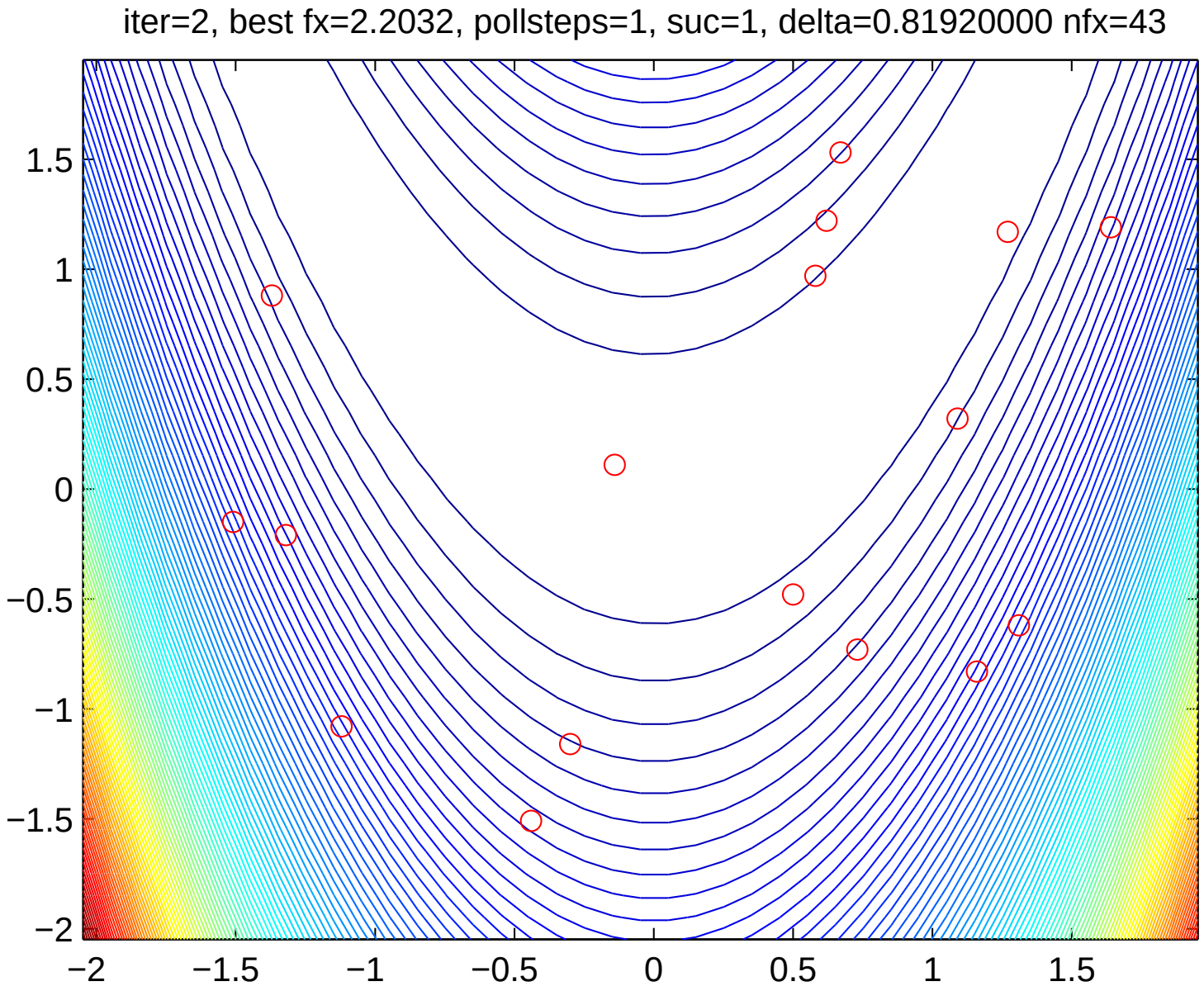
Universidade do Minho

Exemplo com o problema `ir2.mod`

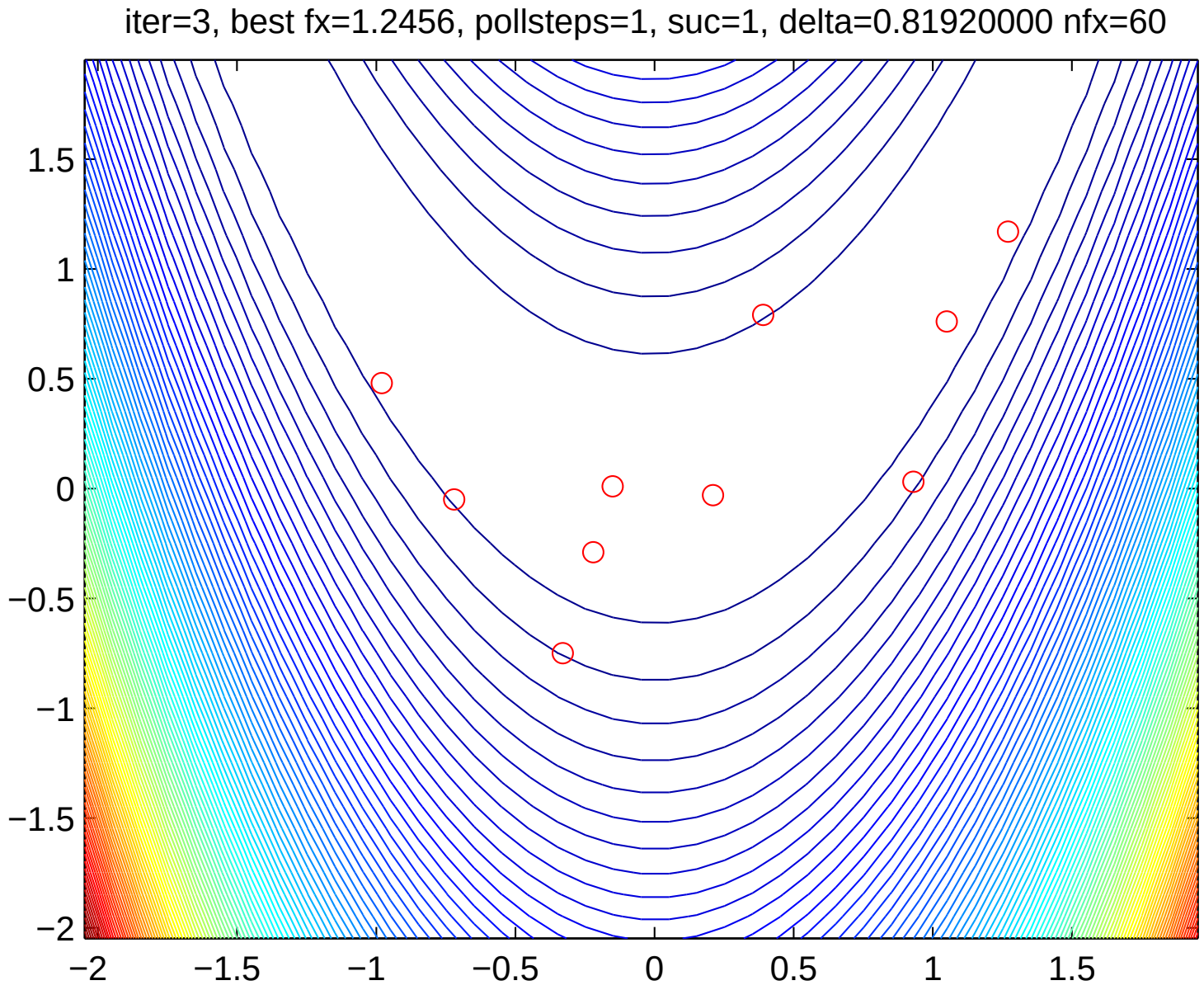


Universidade do Minho

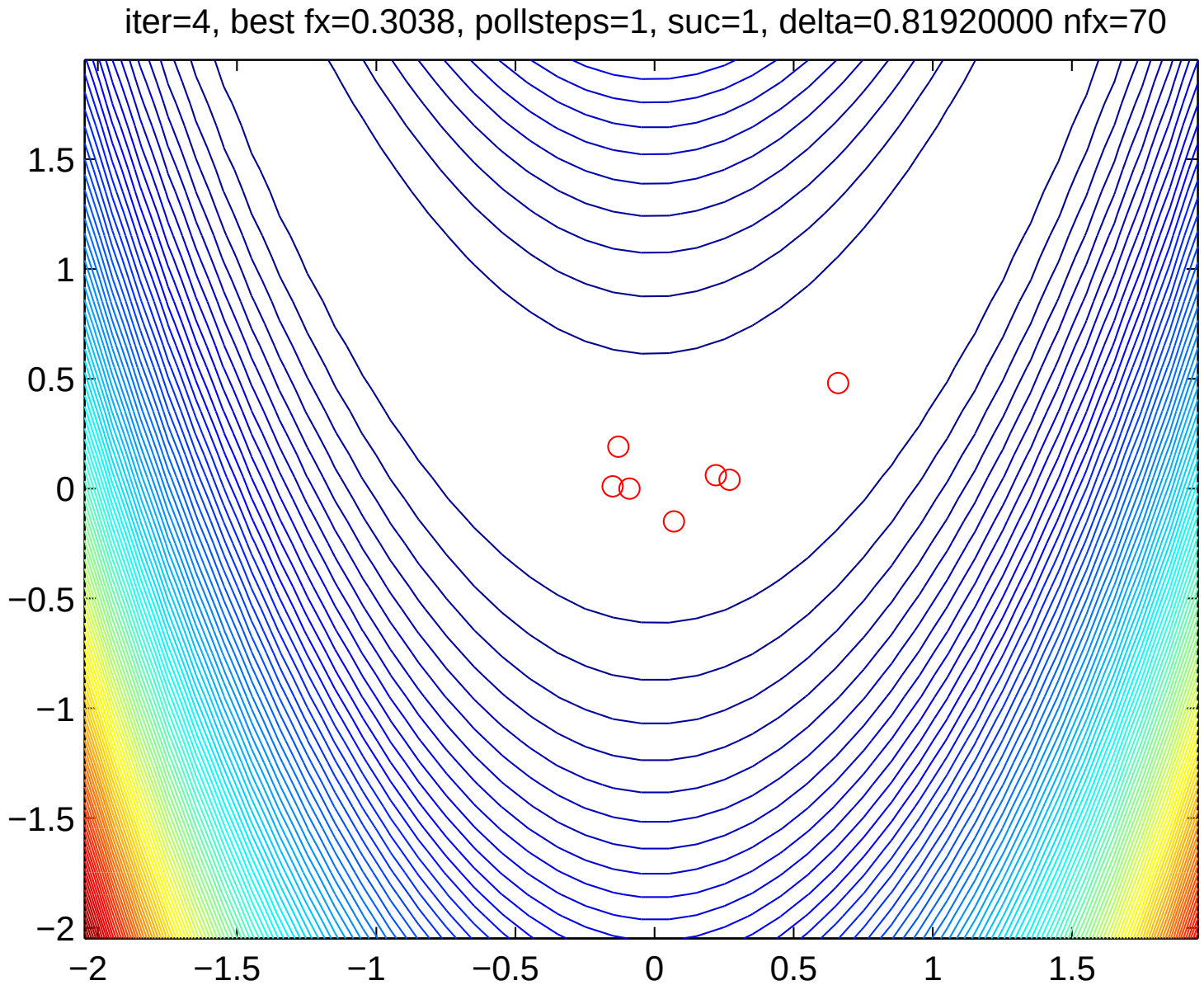
Exemplo com o problema `ir2.mod`



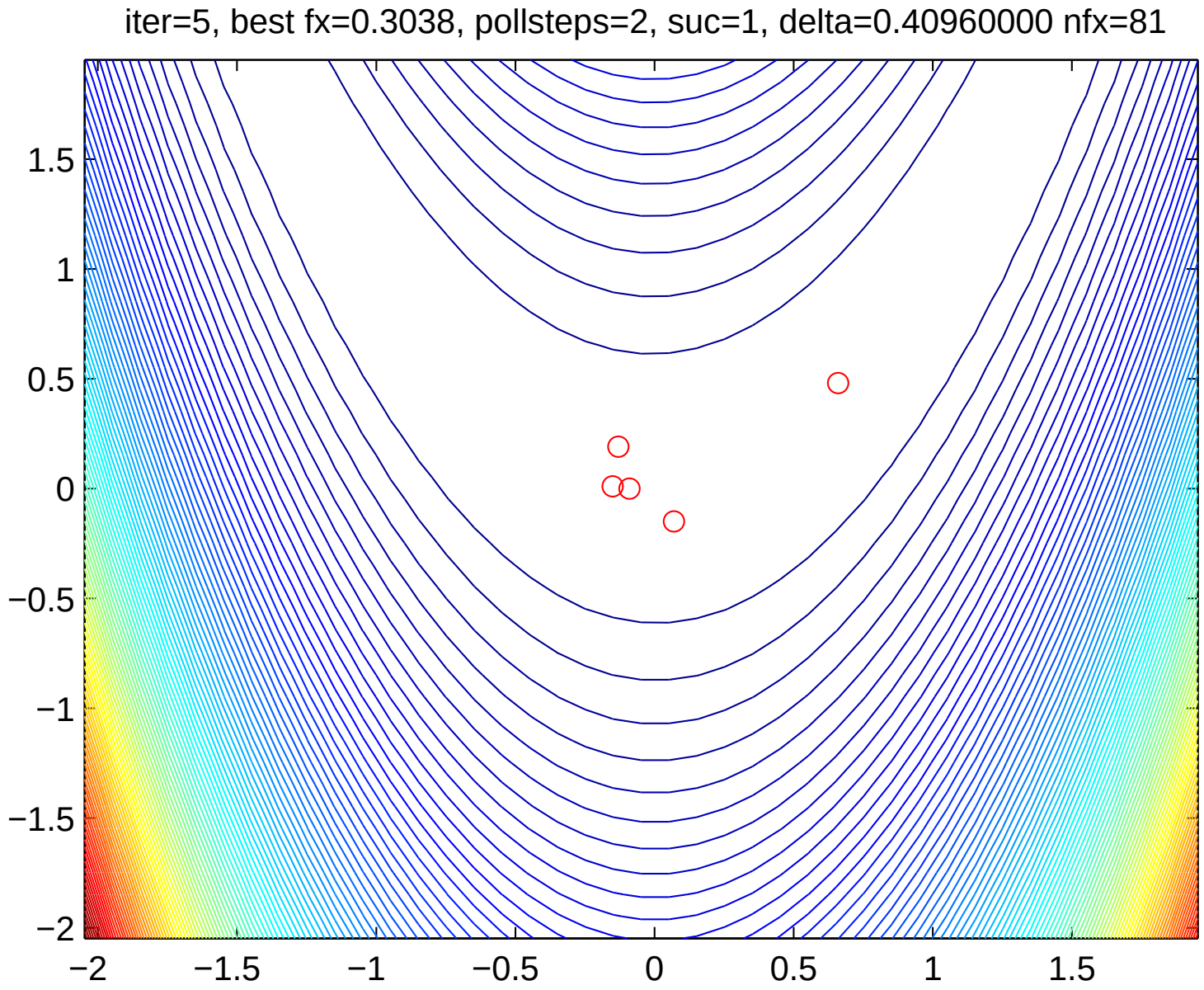
Exemplo com o problema `ir2.mod`



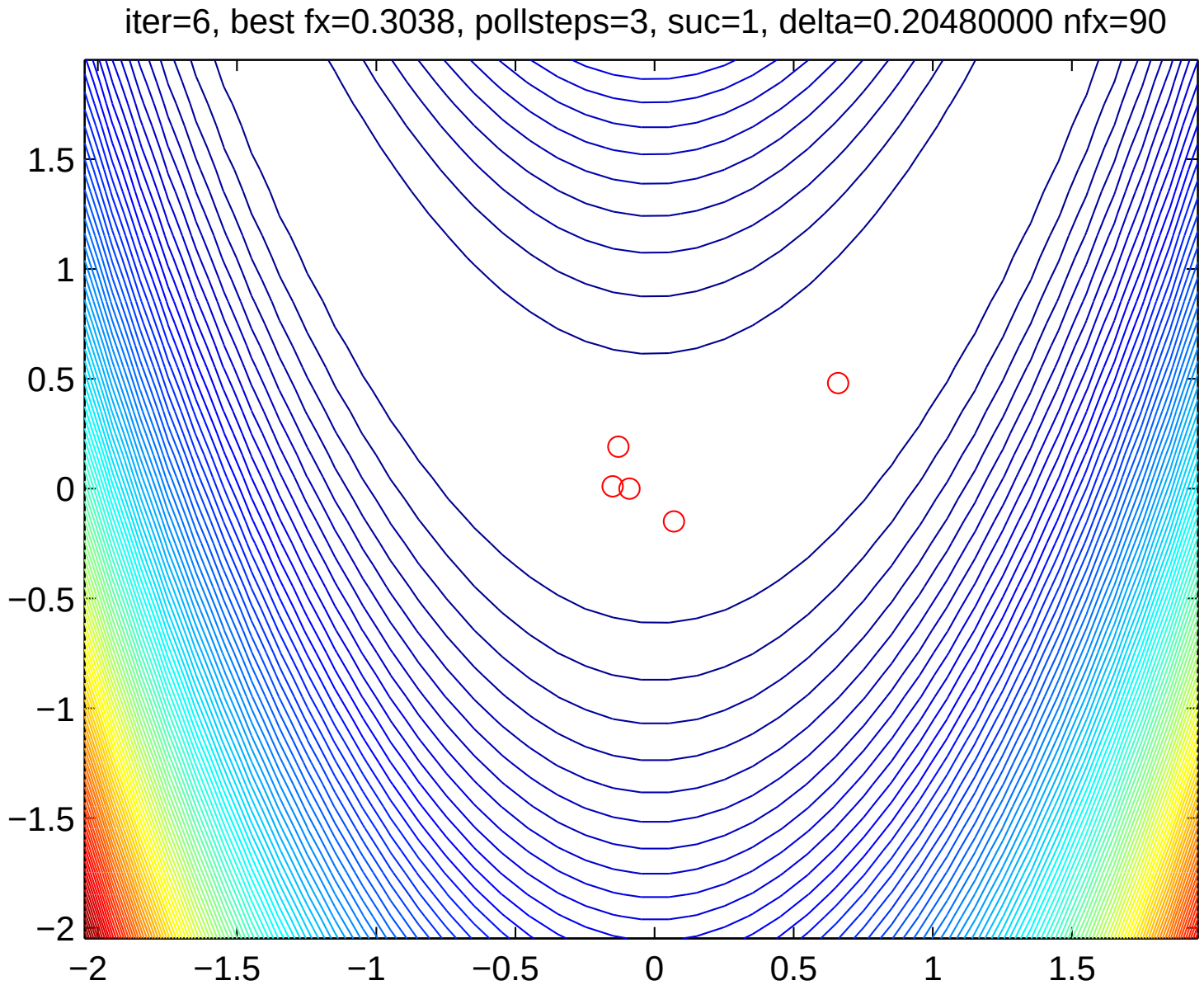
Exemplo com o problema `ir2.mod`



Exemplo com o problema `ir2.mod`

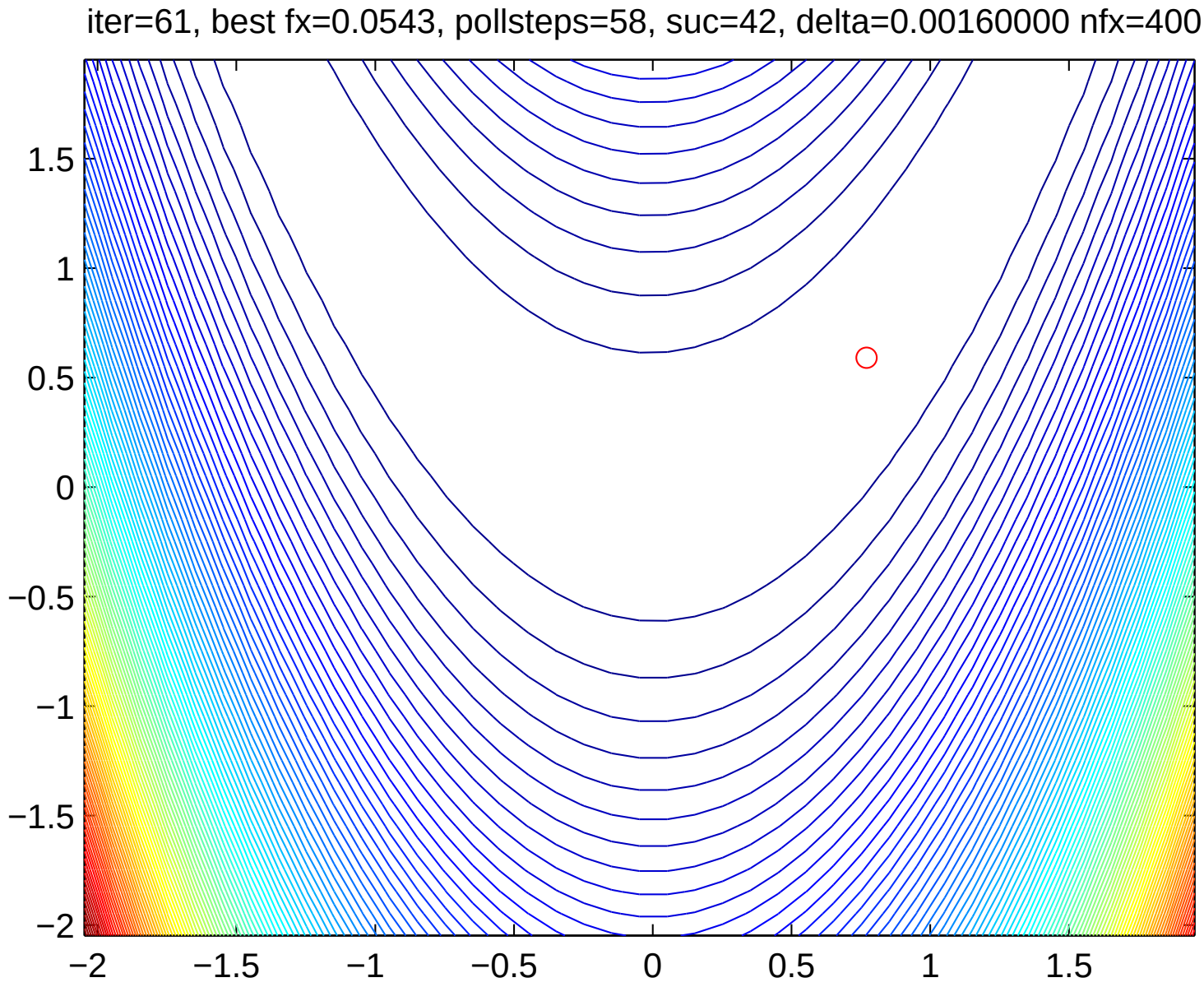


Exemplo com o problema `ir2.mod`



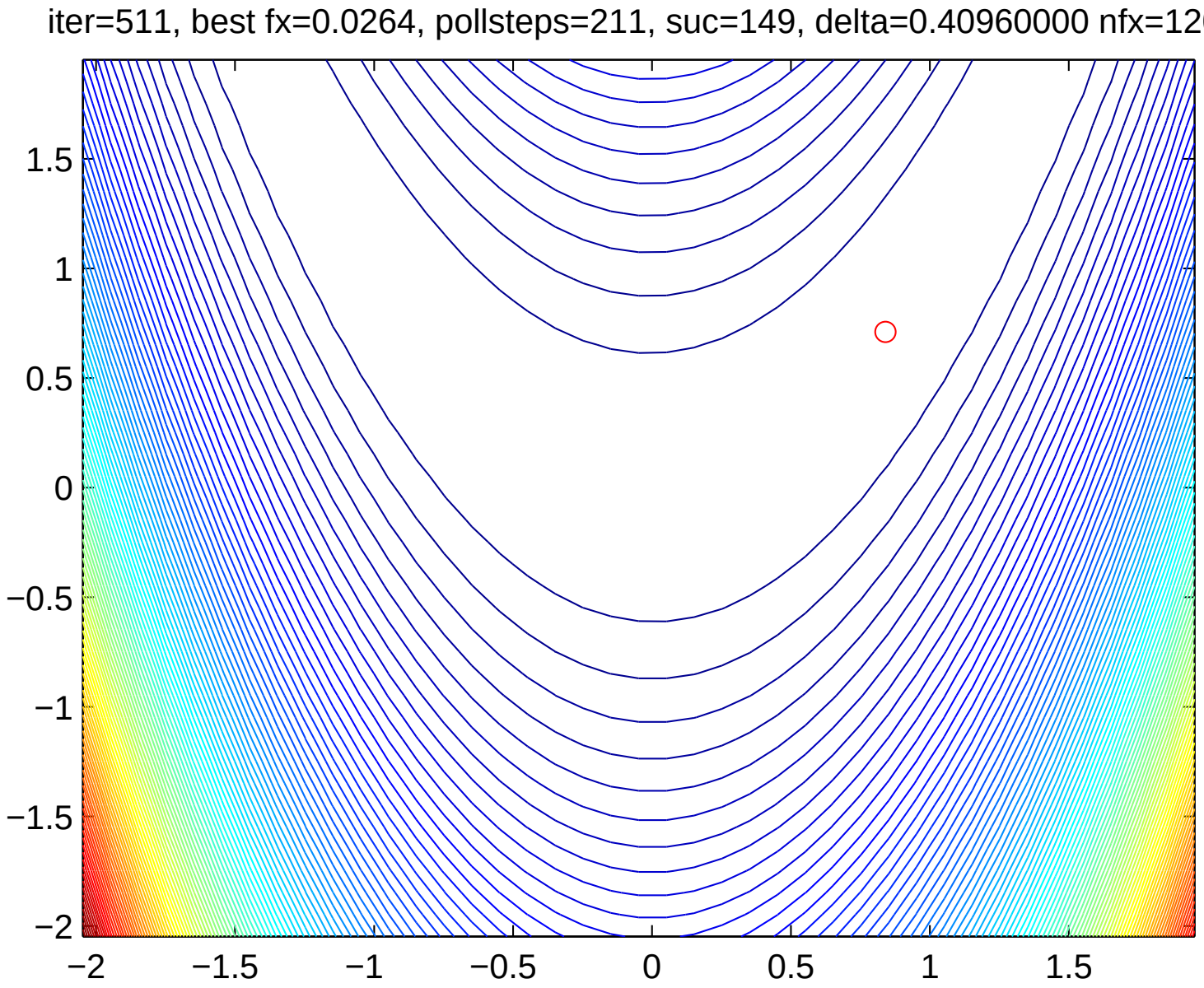
Universidade do Minho

Exemplo com o problema `ir2.mod`

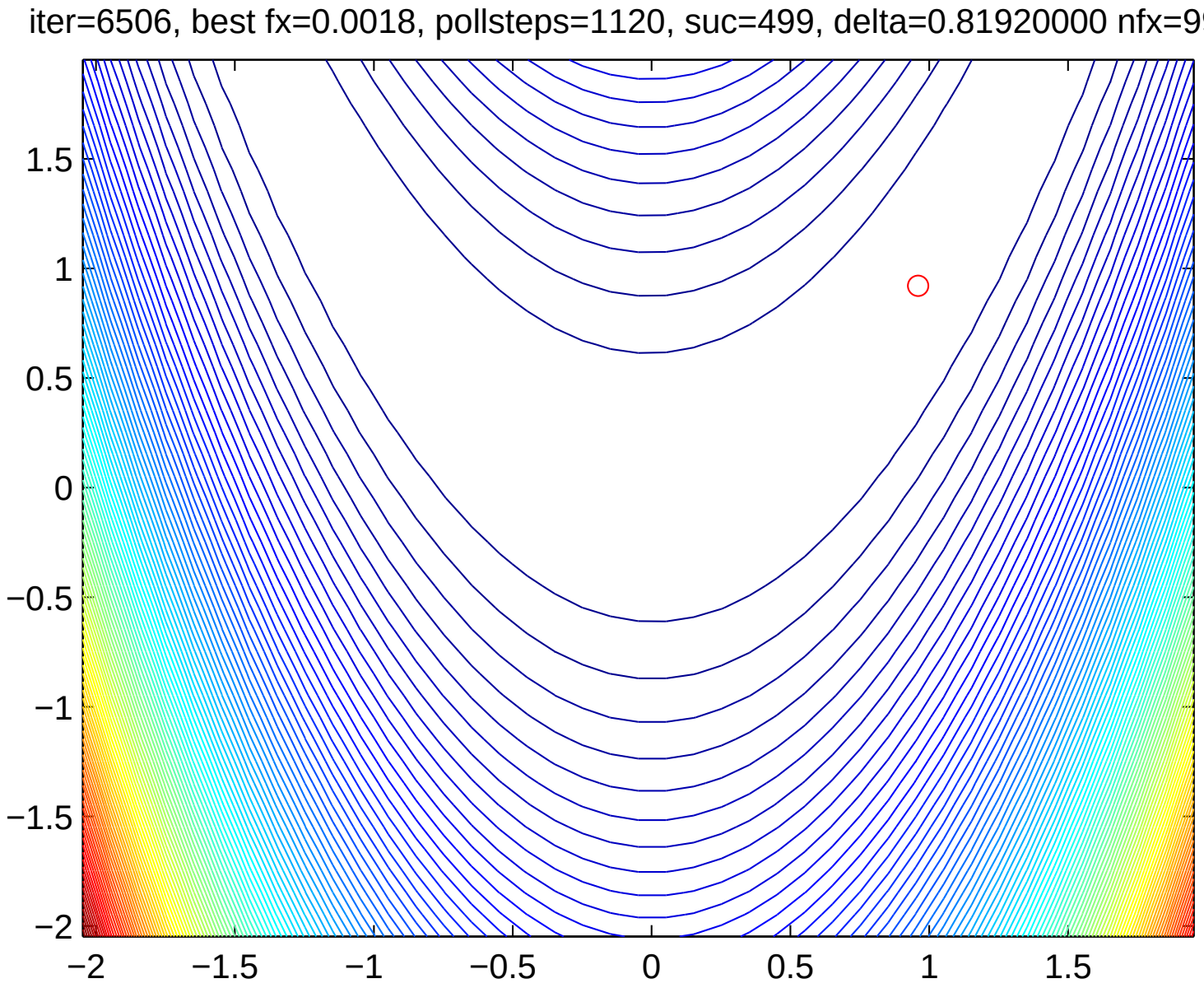


Universidade do Minho

Exemplo com o problema `ir2.mod`



Exemplo com o problema `ir2.mod`



Universidade do Minho

Convergência

Convergência global

Teorema 1 *Seja* $L(\hat{y}(0)) = \{z \in \mathbb{R}^n : f(z) \leq f(\hat{y}(0))\}$ *um conjunto compacto. Então, existe uma subsequência* $\{\hat{y}(t_k)\}$ *de iterandos produzida pelo algoritmo híbrido (com* $\alpha_{tol} = v_{tol} = 0$*) tal que*

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

para algum $\hat{y}_* \in \Omega$*. A subsequência* $\{t_k\}$ *consiste em iterações não sucedidas.*

Terminação finita

Teorema 2 *Suponha-se que para t suficientemente grande se tem que $\iota(t)$, $E(y^i(t))$, $i = 1, \ldots, s$, e $E(\hat{y}(t))$ são constantes e que $E(proj_{M_t}(x^i(t-1) + v^i(t))) = E(x^i(t-1) + v^i(t))$, $i = 1, \ldots, s$. Então, para uma escolha apropriada dos parâmetros de control da colónia de partículas,*

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

e o algoritmo híbrido termina (com 'probabilidade um') num número finito de iterações.

Universidade do Minho

Considerações acerca das hipóteses

Sendo o conjunto de nível $L(y(0))$ limitado a sequência monótona decrescente $\{f(y^i(t))\}$, $i = 1, \ldots, s$, e $\{f(\hat{y}(t))\}$ convergem. Então, é razoável supor que os valores esperados de $y^i(t)$, $i = 1, \ldots, s$, e $\hat{y}(t)$ também convergem.

Por outro lado, a diferença entre $proj_{M_t}(x^i(t-1) + v^i(t))$ e $x^i(t-1) + v^i(t)$, — e consequentemente dos seus valores esperados — é um múltiplo de $\alpha(t)$ para alguns tipos de grelhas. Esta situação ocorre com a procura coordenada, onde $D = D_{\oplus}$. Como a subsequência do parâmetro da grelha converge para zero, existe pelo menos a garantia de que a diferença entre $x^i(t-1) + v^i(t)$ e a sua projecção em M_t converge para zero ao longo da subsequência.

Universidade do Minho

Resultados numéricos

Problemas teste

Foram recolhidos 122 problemas teste da literatura da optimização global.

12 problemas de grande dimensão (número de variáveis entre 100 e 300). Restantes de pequena (< 10) e média dimensão (< 30).

Grande parte das funções são diferenciáveis, mas multimodais.

Todos os problemas com limites simples em todas as variáveis (necessário para o passo de procura — colónia de partículas).

Os problemas teste foram codificados em AMPL (*A Modeling Language for Mathematical Programming*).

Os problemas teste estão disponíveis em `http://www.norg.uminho.pt/aivaz` (em *software*).

Universidade do Minho

Os diversos solvers

- PSwarm (*Pattern Swarm* ou *Pattern Search warm*);
  - Programação em linguagem C.
  - Interface com o AMPL.
  - Estocástico.
  - Baseado numa população de pontos.
  - Baseado apenas na avaliação da função sem estimar derivadas.
- PGAPack (*Parallel Genetic Algorithm pack*);
  - Programação em linguagem C.
  - Construção da interface com o AMPL.
  - Estocástico.
  - Baseado numa população de pontos.
  - Baseado apenas na avaliação da função sem estimar derivadas.

Os diversos solvers

- PSwarm (*Pattern Swarm* ou *Pattern Search warm*);
  - Programação em linguagem C.
  - Interface com o AMPL.
  - Estocástico.
  - Baseado numa população de pontos.
  - Baseado apenas na avaliação da função sem estimar derivadas.
- PGAPack (*Parallel Genetic Algorithm pack*);
  - Programação em linguagem C.
  - Construção da interface com o AMPL.
  - Estocástico.
  - Baseado numa população de pontos.
  - Baseado apenas na avaliação da função sem estimar derivadas.

Os diversos solver

- ASA (*Adaptive Simulated Annealing*);
  - Programação em linguagem C.
  - Construção da interface com o AMPL.
  - Estocástico.
  - Baseado num único iterando.
  - Baseado apenas na avaliação da função. Estima derivadas para actualizar parâmetros.
- `Direct` (*DIviding RECTangles*);
  - Linguagem MATLAB.
  - Construção da interface com o AMPL (apoiada pela interface AMPL-MATLAB).
  - Determinístico.
  - Algoritmo baseado na optimização Lipschitziana. Considera o domínio dividido em rectângulos.
  - Baseado apenas na avaliação da função sem estimar derivadas.

Os diversos solver

- ASA (*Adaptive Simulated Annealing*);
  - Programação em linguagem C.
  - Construção da interface com o AMPL.
  - Estocástico.
  - Baseado num único iterando.
  - Baseado apenas na avaliação da função. Estima derivadas para actualizar parâmetros.
- `Direct` (*DIviding RECTangles*);
  - Linguagem MATLAB.
  - Construção da interface com o AMPL (apoiada pela interface AMPL-MATLAB).
  - Determinístico.
  - Algoritmo baseado na optimização Lipschitziana. Considera o domínio dividido em rectângulos.
  - Baseado apenas na avaliação da função sem estimar derivadas.

Os diversos solver

- MCS (*Multilevel Coordinate Search*;
  - Linguagem MATLAB.
  - Construção da interface com o AMPL (apoiada pela interface AMPL-MATLAB).
  - Determinístico.
  - Algoritmo divide o espaço de procura em pequenas caixas (à imagem do Direct).
  - Baseado apenas na avaliação da função. Estima derivadas.

Universidade do Minho

O que comparar?

Frequentemente compara-se

- Número de iterações (algoritmos baseados em populações e de ponto único).
- Tempo de execução (critérios de paragem, linguagens de programação).
- Número de cálculos da função objectivo (critério de paragem).
- Obtenção do mínimo global (o menor valor da função objectivo).

Universidade do Minho

O que comparar?

Frequentemente compara-se

- Número de iterações (algoritmos baseados em populações e de ponto único).
- Tempo de execução (critérios de paragem, linguagens de programação).
- Número de cálculos da função objectivo (critério de paragem).
- Obtenção do mínimo global (o menor valor da função objectivo).

O que comparar?

Frequentemente compara-se

- Número de iterações (algoritmos baseados em populações e de ponto único).
- Tempo de execução (critérios de paragem, linguagens de programação).
- Número de cálculos da função objectivo (critério de paragem).
- Obtenção do mínimo global (o menor valor da função objectivo).

Universidade do Minho

O que comparar?

Frequentemente compara-se

- Número de iterações (algoritmos baseados em populações e de ponto único).
- Tempo de execução (critérios de paragem, linguagens de programação).
- Número de cálculos da função objectivo (critério de paragem).
- Obtenção do mínimo global (o menor valor da função objectivo).

Universidade do Minho

O que comparar?

Frequentemente compara-se

- Número de iterações (algoritmos baseados em populações e de ponto único).
- Tempo de execução (critérios de paragem, linguagens de programação).
- Número de cálculos da função objectivo (critério de paragem).
- Obtenção do mínimo global (o menor valor da função objectivo).

Usado: O valor da função objectivo no melhor pontos encontrado (solução), impondo um máximo para o número de cálculos da função objectivo.

Universidade do Minho

O que comparar?

Frequentemente compara-se

- Número de iterações (algoritmos baseados em populações e de ponto único).
- Tempo de execução (critérios de paragem, linguagens de programação).
- Número de cálculos da função objectivo (critério de paragem).
- Obtenção do mínimo global (o menor valor da função objectivo).

Usado: O valor da função objectivo no melhor pontos encontrado (solução), impondo um máximo para o número de cálculos da função objectivo.

Foram efectuados 30 runs para os *solver* estocásticos.

Universidade do Minho

Como comparar?

Perfis de desempenho (*Performance profiles* – Dolan e Moré, 2003).

Uma vantagem dos perfis de desempenho é que podem ser apresentados numa figura, desenhando para cada *solver* uma função distribuição acumulada $\rho(\tau)$ que representa uma razão de desempenho.

A razão de desempenho é definida por $r_{p,s} = \frac{t_{p,s}}{\min\{t_{p,s} : s \in \mathcal{S}\}}$, onde $\mathcal{P}$ é o conjunto de problemas teste, $p \in \mathcal{P}$, $\mathcal{S}$ é o conjunto de *solvers*, $s \in \mathcal{S}$, e $t_{p,s}$ é o valor obtido pelo *solver* s para o problema teste p.

$r_{p,s}$ foi adaptado para este caso em particular. Evita os possíveis valores não positivos de $t_{p,s}$.

Seja $\rho_s(\tau) = \frac{1}{n_p} \mathit{size}\{p \in \mathcal{P} : r_{p,s} \leq \tau\}$, onde n_p é o número de problemas teste.

Universidade do Minho

Como comparar?

$\rho_s(1) = \frac{1}{n_p} size\{p \in \mathcal{P} : r_{p,s} \leq 1\}$ é a probabilidade de o *solver* ganhar aos restantes. Maior valor de $\rho_s(1)$ significa que terá maiores probabilidades de ganhar (ser o melhor).

Por outro lado, os *solver* com maior valor de $\rho_s(\tau)$, para valores grande são os mais robustos. Se $\rho_s(\tau) = 1$ então o solver s resolve todos os problemas.

$t_{p,s}$ =(melhor/média/pior) valor da função obtido para o problema p pelo *solver* s (para todos os *runs* se o solver for estocástico).

Universidade do Minho

Parâmetros usados

PSwarm

$\alpha_{tol} = 10^{-5}$, $\nu = \mu = 0.5$, $\phi(t) = 2$, $\theta(t) = 0.5$,
$\alpha(0) = \max_{j=1,..,n}(u_j - \ell_j)/c$ com $c = 5$ e $s = 20$.

A projecção na grelha ($\hat{x}^i(t) = proj_{M_t}(x^i(t))$) não foi implementada.

O parâmetro de inércia ι é linearmente interpolado entre 0.9 e 0.4, *i.e.*, $\iota(t) = 0.9 - (0.5/t_{max})t$, onde t_{max} é o número máximo de iterações permitidas.

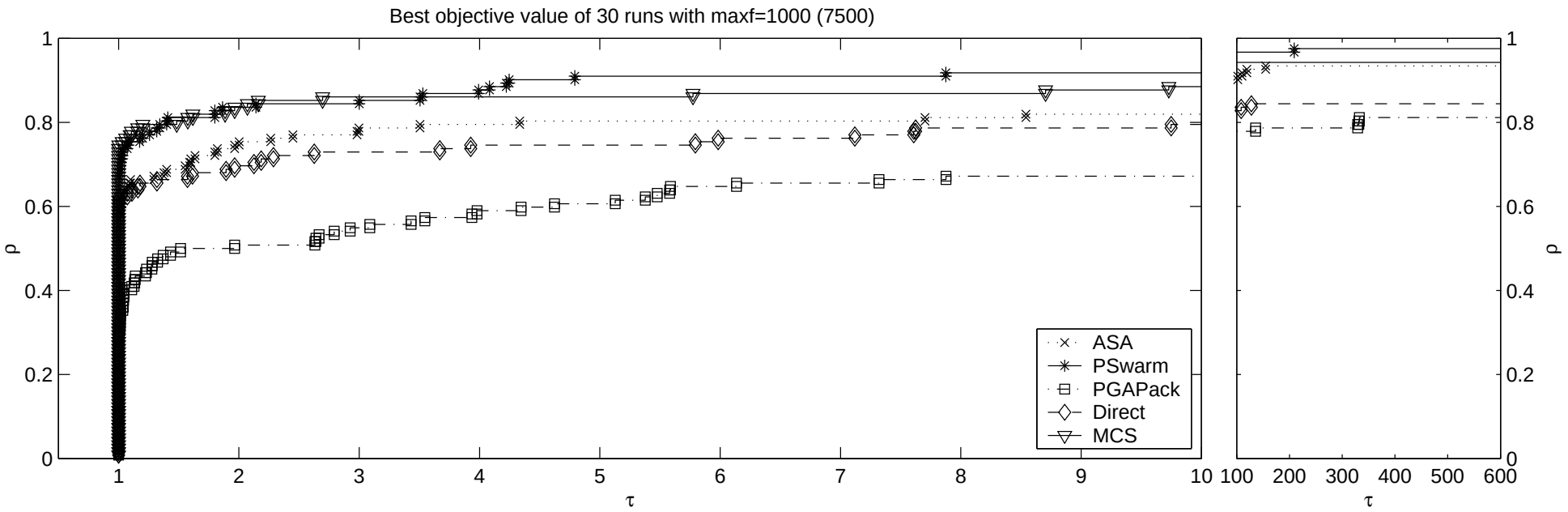
A população inicial é obtida gerando s pontos aleatórios segundo a distribuição uniforme $U(\ell, u)$, ficando $x_j^i(0) \sim U(\ell_j, u_j)$, $j = 1, \ldots, n$, para todas as partículas $i = 1, \ldots, s$ (aproximações iniciais admissíveis).

PGAPack

População de tamanho 200.

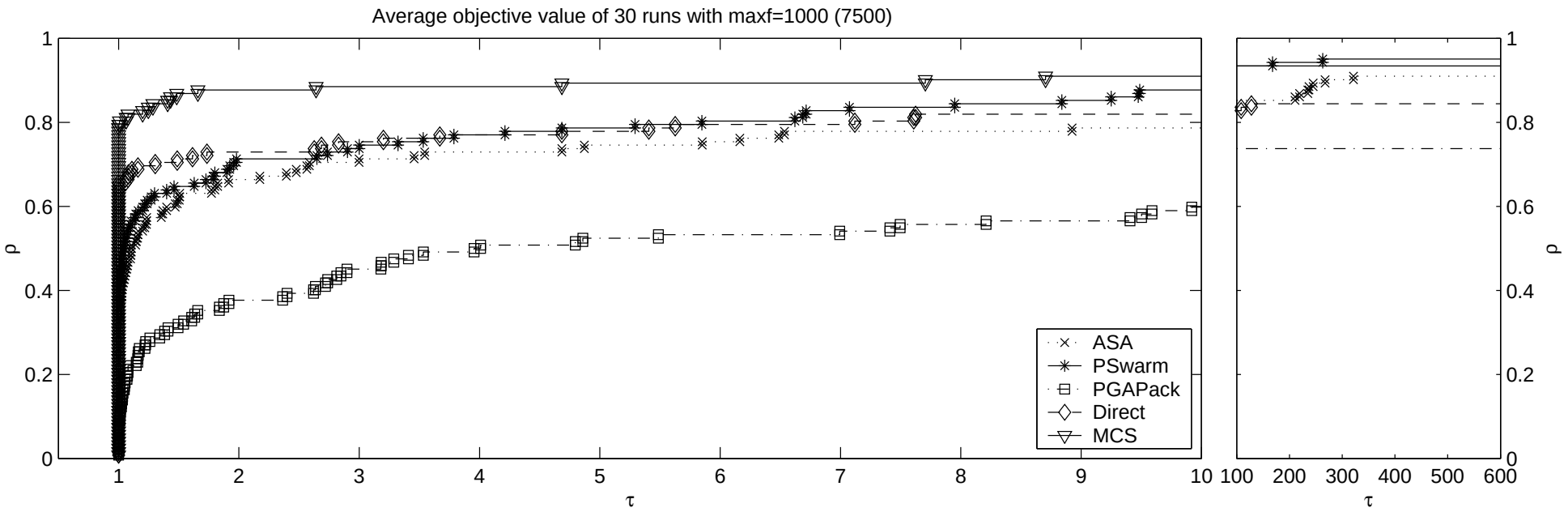
Profiles

Universidade do Minho



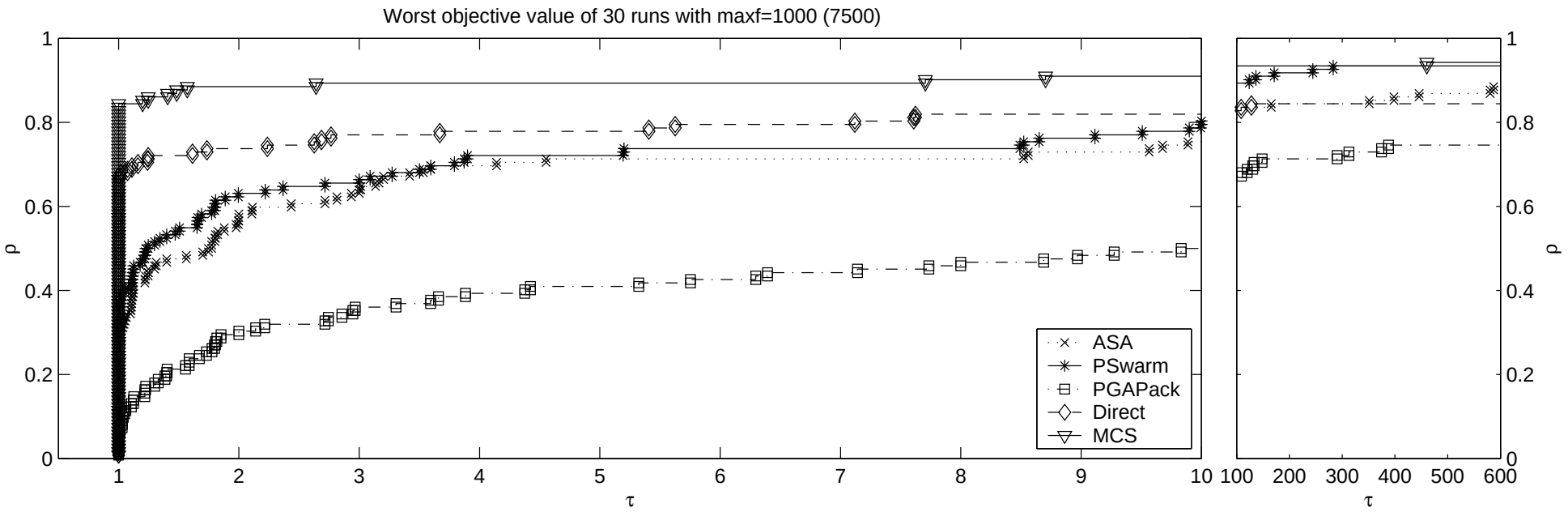
Profiles

Universidade do Minho



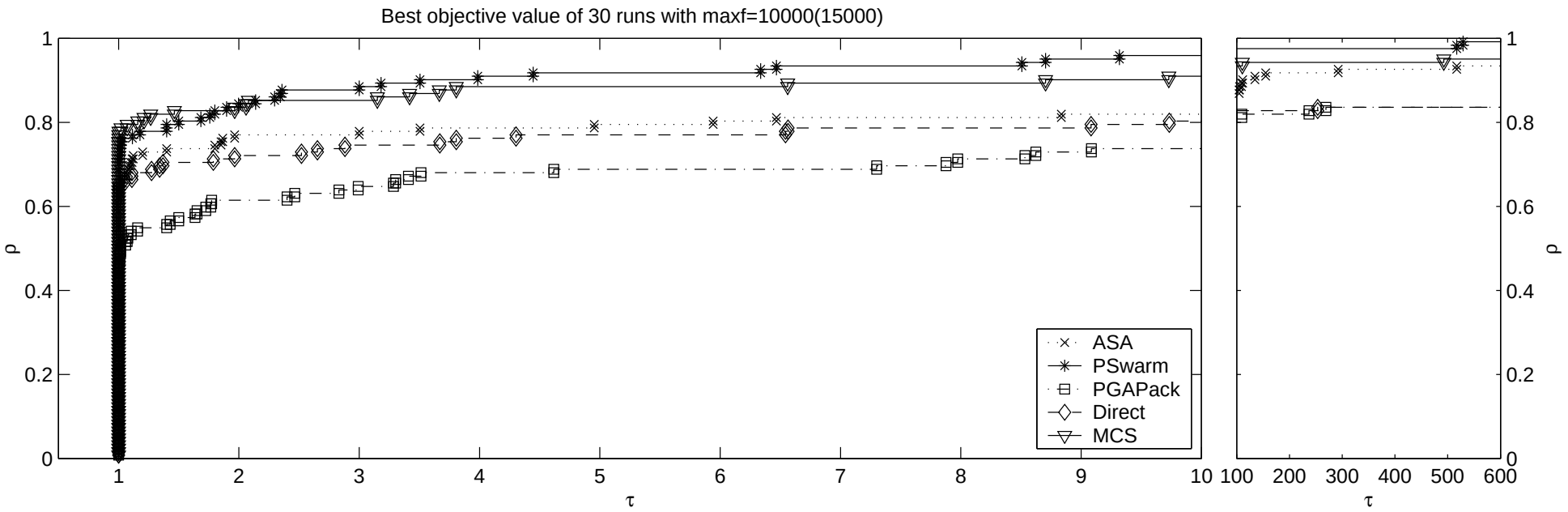
Profiles

Universidade do Minho



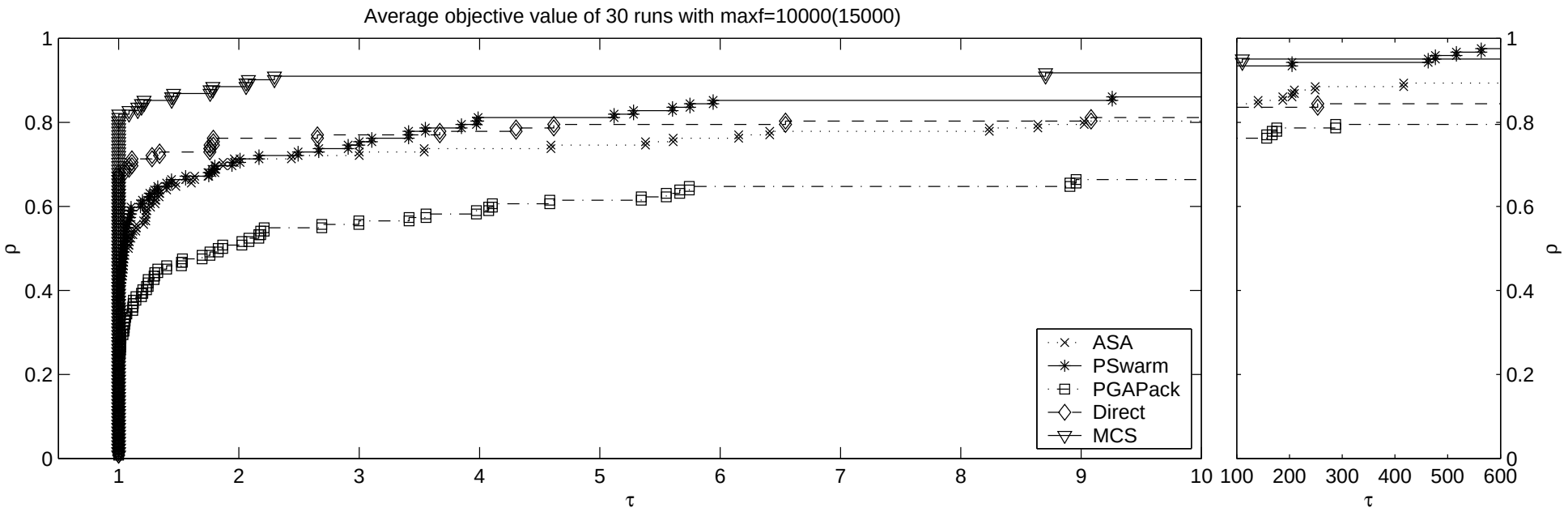
Profiles

Universidade do Minho



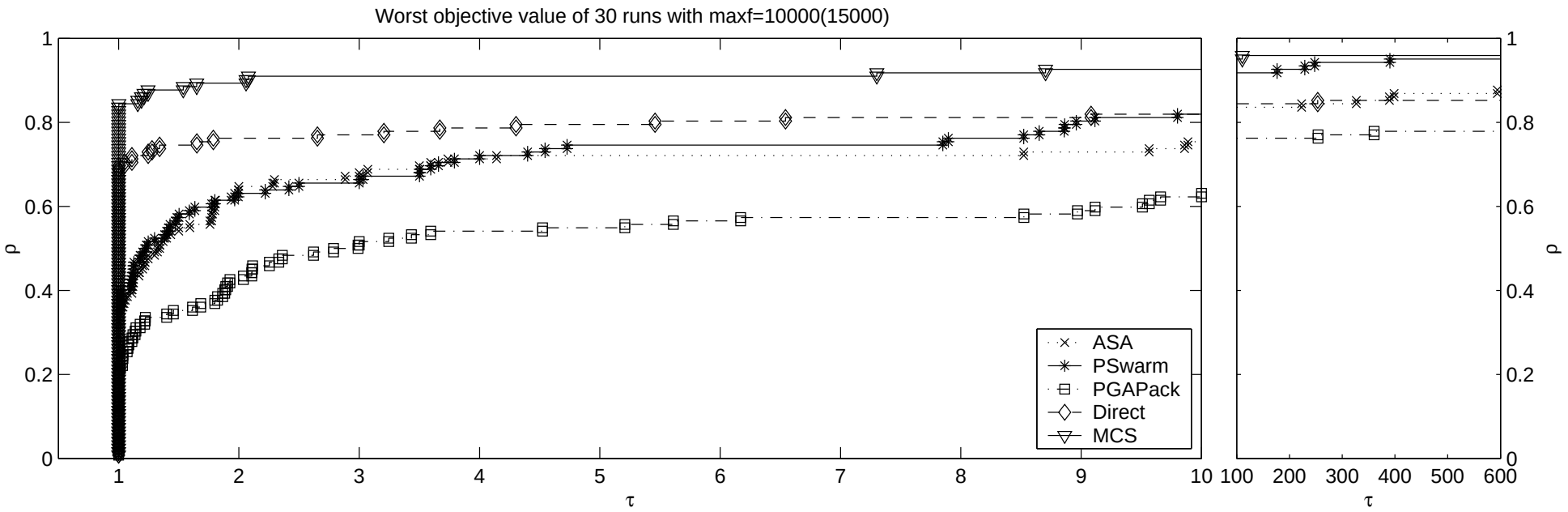
Profiles

Universidade do Minho



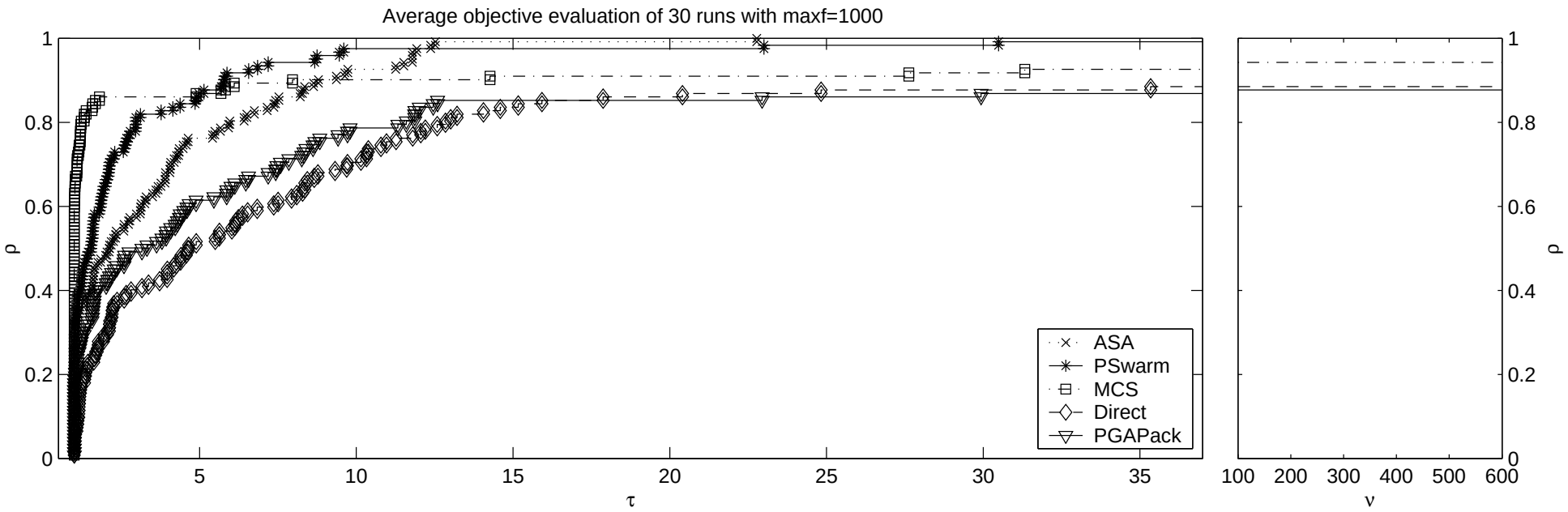
Profiles

Universidade do Minho



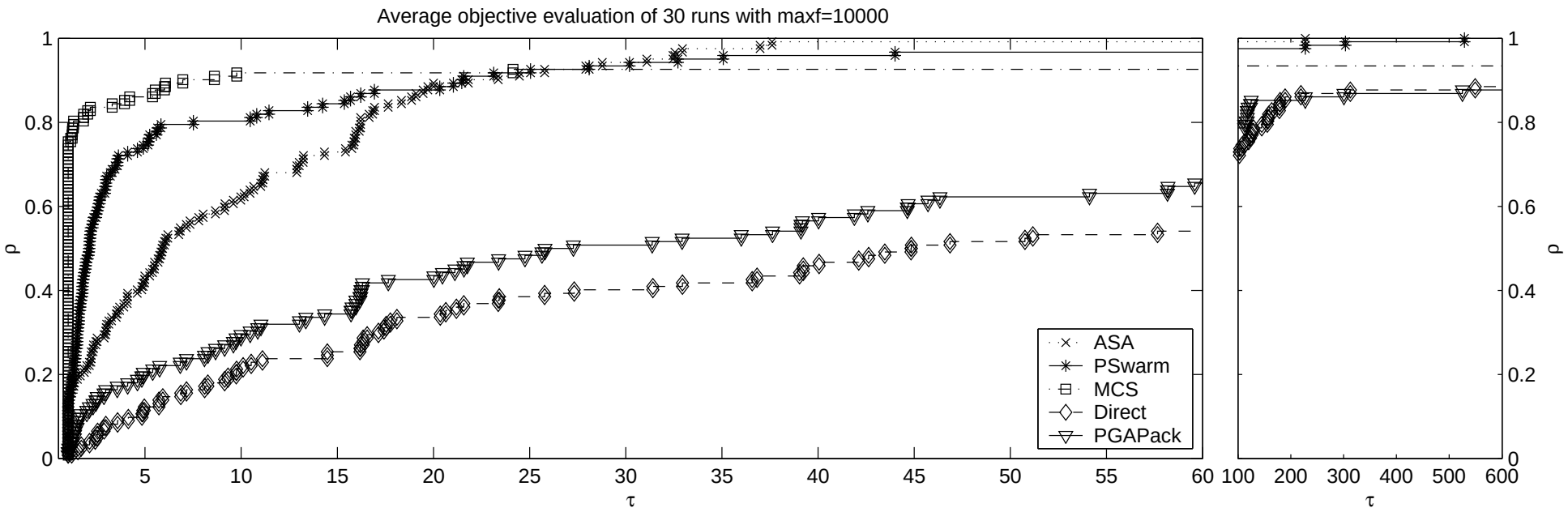
Média de cálculos da função objectivo

Universidade do Minho



Média de cálculos da função objectivo

Universidade do Minho



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

Universidade do Minho

Conclusões e trabalho futuro

Universidade do Minho

Conclusões e trabalho futuro

Conclusões

- Desenvolvimento de um algoritmo híbrido para optimização global.
- Análise das condições de terminação do algoritmo.
- PSwarm demonstrou ser um algoritmo robusto e competitivo.

Trabalho futuro

- Versão paralela, uma vez que as colónias de partículas e a procura em padrão são facilmente paralelizáveis, também o é o algoritmo híbrido.
- Tratamento de restrições mais gerais (lineares e não lineares).

Conclusões e trabalho futuro

Conclusões

- Desenvolvimento de um algoritmo híbrido para optimização global.
- Análise das condições de terminação do algoritmo.
- PSwarm demonstrou ser um algoritmo robusto e competitivo.

Trabalho futuro

- Versão paralela, uma vez que as colónias de partículas e a procura em padrão são facilmente paralelizáveis, também o é o algoritmo híbrido.
- Tratamento de restrições mais gerais (lineares e não lineares).

Universidade do Minho

Conclusões e trabalho futuro

Conclusões

- Desenvolvimento de um algoritmo híbrido para optimização global.
- Análise das condições de terminação do algoritmo.
- PSwarm demonstrou ser um algoritmo robusto e competitivo.

Trabalho futuro

- Versão paralela, uma vez que as colónias de partículas e a procura em padrão são facilmente paralelizáveis, também o é o algoritmo híbrido.
- Tratamento de restrições mais gerais (lineares e não lineares).

Universidade do Minho

Conclusões e trabalho futuro

Conclusões

- Desenvolvimento de um algoritmo híbrido para optimização global.
- Análise das condições de terminação do algoritmo.
- PSwarm demonstrou ser um algoritmo robusto e competitivo.

Trabalho futuro

- Versão paralela, uma vez que as colónias de partículas e a procura em padrão são facilmente paralelizáveis, também o é o algoritmo híbrido.
- Tratamento de restrições mais gerais (lineares e não lineares).

Conclusões e trabalho futuro

Conclusões

- Desenvolvimento de um algoritmo híbrido para optimização global.
- Análise das condições de terminação do algoritmo.
- PSwarm demonstrou ser um algoritmo robusto e competitivo.

Trabalho futuro

- Versão paralela, uma vez que as colónias de partículas e a procura em padrão são facilmente paralelizáveis, também o é o algoritmo híbrido.
- Tratamento de restrições mais gerais (lineares e não lineares).

Universidade do Minho

Fim

email: aivaz@dps.uminho.pt

Web http://www.norg.uminho.pt/

email: lnv@mat.uc.pt

Web http://www.mat.uc.pt/∼lnv