Adapt-MEMPSODE: A Memetic Algorithm with Adaptive Selection of Local Searches

Costas Voglis

Department of Computer Science
University of Ioannina
Greece

July 6, 2013

Presentation outline

1. Introduction
   - Memetic global optimization
   - Adaptive memetic global optimization
   - This year's contribution (BBOB 2013)
2. adapt-MEMPSODE description
   - Swarm Intelligent components
   - Memetic strategy
   - Local searches
   - Adaptive selection
3. Many local searches?
   - Local search mapping
   - Local search on a track
4. Experimental results
   - Experimental procedure
   - Results

Contents

1. Introduction
   - Memetic global optimization
   - Adaptive memetic global optimization
   - This year's contribution (BBOB 2013)
2. adapt-MEMPSODE description
   - Swarm Intelligent components
   - Memetic strategy
   - Local searches
   - Adaptive selection
3. Many local searches?
   - Local search mapping
   - Local search on a track
4. Experimental results
   - Experimental procedure
   - Results

Memetic global optimization

Memetic algorithms

Two parts

1. *exploration* ensures that every part of the domain will be covered and
2. *exploitation* concentrates the search effort in a close neighbourhood of the so far best detected positions.

MEMPSODE software

MEMPSODE global optimization software tool integrates Particle Swarm Optimization and Differential Evolution with well established efficient local search procedures

Adaptive Memetic algorithms

Memetic algorithms that select from a pool of local searches

Memetic global optimization

Memetic algorithms

Two parts

1. *exploration* ensures that every part of the domain will be covered and
2. *exploitation* concentrates the search effort in a close neighbourhood of the so far best detected positions.

MEMPSODE software

MEMPSODE global optimization software tool integrates Particle Swarm Optimization and Differential Evolution with well established efficient local search procedures

Adaptive Memetic algorithms

Memetic algorithms that select from a pool of local searches

Memetic global optimization

Memetic algorithms

Two parts

1. *exploration* ensures that every part of the domain will be covered and
2. *exploitation* concentrates the search effort in a close neighbourhood of the so far best detected positions.

MEMPSODE software

MEMPSODE global optimization software tool integrates Particle Swarm Optimization and Differential Evolution with well established efficient local search procedures

Adaptive Memetic algorithms

Memetic algorithms that select from a pool of local searches

Classification of Adaptive MAs

Adaptive MAs can be classified [Ong et al., 2006]:

According to the adaptation type:

1. *S*tatic: No form of feedback is considered during the search
2. *A*daptive: A specific feedback is considered
3. *S*elf-Adaptive: The local searches co-evolve by evolutionary operators

According to the feedback type:

1. *Q*ualitative: Simple binary (*good* or *bad*) feedback
2. *Q*uantitative: Numeric feedback

According to the feedback level:

1. *E*xternal: Problem-specific knowledge from past experience is exploited
2. *L*ocal: A recent part of the trace is used
3. *G*lobal: The complete progress of the algorithm is used

Classification of Adaptive MAs

Adaptive MAs can be classified [Ong et al., 2006]:

According to the adaptation type:

1. *S*tatic: No form of feedback is considered during the search
2. *A*daptive: A specific feedback is considered
3. *S*elf-Adaptive: The local searches co-evolve by evolutionary operators

According to the feedback type:

1. *Q*ualitative: Simple binary (*good* or *bad*) feedback
2. *Q*uantitative: Numeric feedback

According to the feedback level:

1. *E*xternal: Problem-specific knowledge from past experience is exploited
2. *L*ocal: A recent part of the trace is used
3. *G*lobal: The complete progress of the algorithm is used

Classification of Adaptive MAs

Adaptive MAs can be classified [Ong et al., 2006]:

According to the adaptation type:

1. *S*tatic: No form of feedback is considered during the search
2. *A*daptive: A specific feedback is considered
3. *S*elf-Adaptive: The local searches co-evolve by evolutionary operators

According to the feedback type:

1. *Q*ualitative: Simple binary (*good* or *bad*) feedback
2. *Q*uantitative: Numeric feedback

According to the feedback level:

1. *E*xternal: Problem-specific knowledge from past experience is exploited
2. *L*ocal: A recent part of the trace is used
3. *G*lobal: The complete progress of the algorithm is used

Classification of Adaptive MAs

Adaptive MAs can be classified [Ong et al., 2006]:

According to the adaptation type:

1. *S*tatic: No form of feedback is considered during the search
2. *A*daptive: A specific feedback is considered
3. *S*elf-Adaptive: The local searches co-evolve by evolutionary operators

According to the feedback type:

1. *Q*ualitative: Simple binary (*good* or *bad*) feedback
2. *Q*uantitative: Numeric feedback

According to the feedback level:

1. *E*xternal: Problem-specific knowledge from past experience is exploited
2. *L*ocal: A recent part of the trace is used
3. *G*lobal: The complete progress of the algorithm is used

Classification of Adaptive MAs

Adaptive MAs can be classified [Ong et al., 2006]:

According to the adaptation type:

1. *S*tatic: No form of feedback is considered during the search
2. *A*daptive: A specific feedback is considered
3. *S*elf-Adaptive: The local searches co-evolve by evolutionary operators

According to the feedback type:

1. *Q*ualitative: Simple binary (*good* or *bad*) feedback
2. *Q*uantitative: Numeric feedback

According to the feedback level:

1. *E*xternal: Problem-specific knowledge from past experience is exploited
2. *L*ocal: A recent part of the trace is used
3. *G*lobal: The complete progress of the algorithm is used

Classification of Adaptive MAs

Adaptive MAs can be classified [Ong et al., 2006]:

According to the adaptation type:

1. *S*tatic: No form of feedback is considered during the search
2. *A*daptive: A specific feedback is considered
3. *S*elf-Adaptive: The local searches co-evolve by evolutionary operators

According to the feedback type:

1. *Q*ualitative: Simple binary (*good* or *bad*) feedback
2. *Q*uantitative: Numeric feedback

According to the feedback level:

1. *E*xternal: Problem-specific knowledge from past experience is exploited
2. *L*ocal: A recent part of the trace is used
3. *G*lobal: The complete progress of the algorithm is used

Classification of Adaptive MAs

Adaptive MAs can be classified [Ong et al., 2006]:

According to the adaptation type:

1. *S*tatic: No form of feedback is considered during the search
2. *A*daptive: A specific feedback is considered
3. *S*elf-Adaptive: The local searches co-evolve by evolutionary operators

According to the feedback type:

1. *Q*ualitative: Simple binary (*good* or *bad*) feedback
2. *Q*uantitative: Numeric feedback

According to the feedback level:

1. *E*xternal: Problem-specific knowledge from past experience is exploited
2. *L*ocal: A recent part of the trace is used
3. *G*lobal: The complete progress of the algorithm is used

Classification of Adaptive MAs

Adaptive MAs can be classified [Ong et al., 2006]:

According to the adaptation type:

1. *S*tatic: No form of feedback is considered during the search
2. *A*daptive: A specific feedback is considered
3. *S*elf-Adaptive: The local searches co-evolve by evolutionary operators

According to the feedback type:

1. *Q*ualitative: Simple binary (*good* or *bad*) feedback
2. *Q*uantitative: Numeric feedback

According to the feedback level:

1. *E*xternal: Problem-specific knowledge from past experience is exploited
2. *L*ocal: A recent part of the trace is used
3. *G*lobal: The complete progress of the algorithm is used

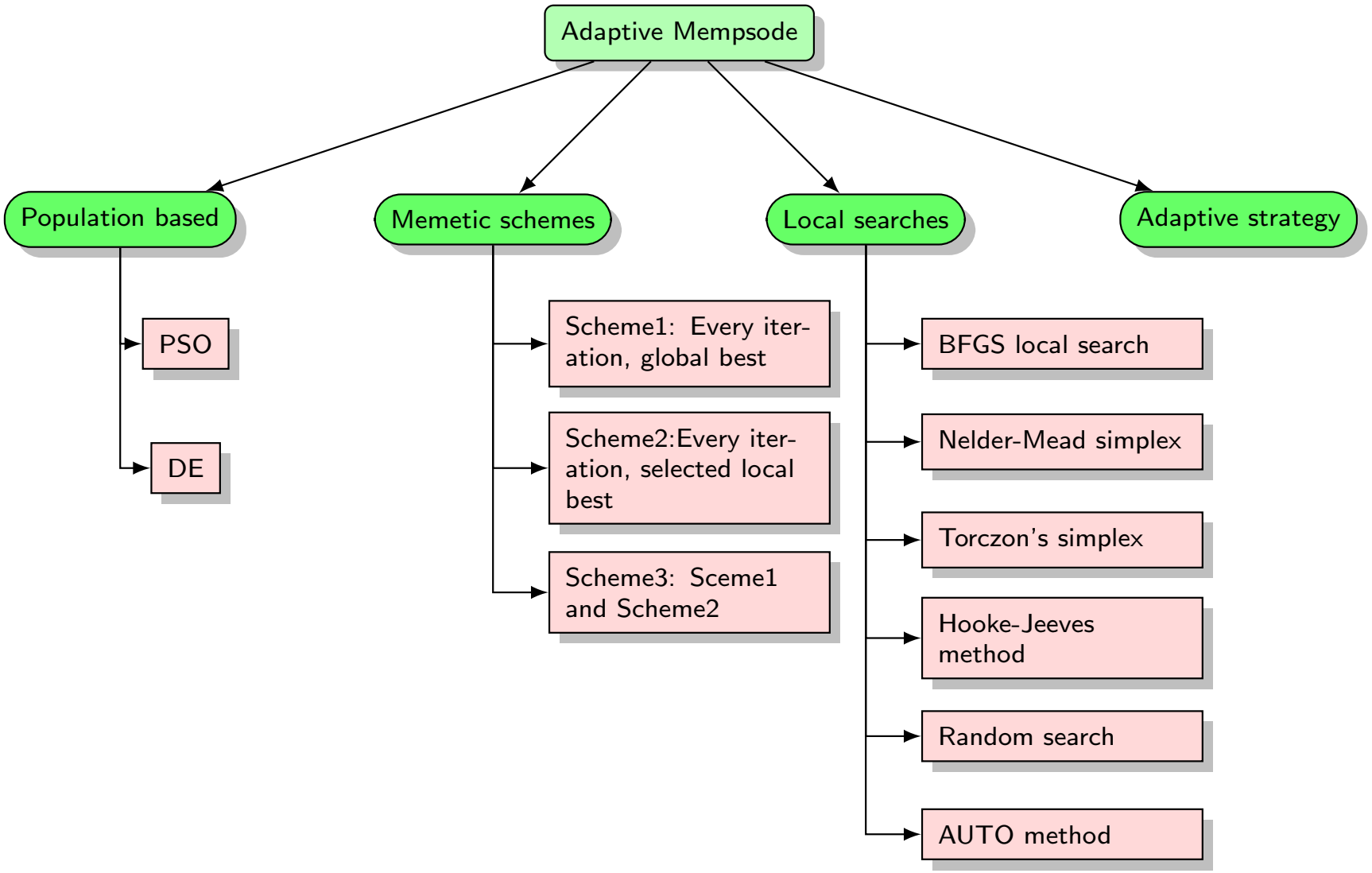
Adaptive MEMPSODE

- Adaptive extension of MEMPSODE:
  1. Adaptation type: *A*daptive
  2. Feedback level: *Q*uantitative
  3. Feedback level: *G*lobal
- Stochastic (roulette) selection from available local searches
- More that 20% improvement than last year's entry

Contents

1. Introduction
   - Memetic global optimization
   - Adaptive memetic global optimization
   - This year's contribution (BBOB 2013)
2. adapt-MEMPSODE description
   - Swarm Intelligent components
   - Memetic strategy
   - Local searches
   - Adaptive selection
3. Many local searches?
   - Local search mapping
   - Local search on a track
4. Experimental results
   - Experimental procedure
   - Results

Players


Adaptive Mempsode
Population based
Memetic schemes
Local searches
Adaptive strategy
PSO
DE
Scheme1: Every iteration, global best
Scheme2:Every iteration, selected local best
Scheme3: Sceme1 and Scheme2
BFGS local search
Nelder-Mead simplex
Torczon's simplex
Hooke-Jeeves method
Random search
AUTO method

MEMPSODE in a nutshell

First presented in [Voglis et al., 2012]

- Global optimization software tool
- Implements a memetic algorithm
  - Synergy: local and population based global optimization
  - Exploration phase: PSO or DE
  - Exploitation phase: Merlin local optimization environment
- Standalone executable or user callable interface
- Fully parameterizable
- Accompanied by many test functions

MEMPSODE in a nutshell

First presented in [Voglis et al., 2012]

- Global optimization software tool
- Implements a memetic algorithm
  - Synergy: local and population based global optimization
  - Exploration phase: PSO or DE
  - Exploitation phase: Merlin local optimization environment
- Standalone executable or user callable interface
- Fully parameterizable
- Accompanied by many test functions

local

Deterministic sequential algorith aiming a local minimum

MEMPSODE in a nutshell

First presented in [Voglis et al., 2012]

- Global optimization software tool
- Implements a memetic algorithm
  - Synergy: local and population based global optimization
  - Exploration phase: PSO or DE
  - Exploitation phase: Merlin local optimization environment
- Standalone executable or user callable interface
- Fully parameterizable
- Accompanied by many test functions

population based

Stochastic models that draw the inspiration from physical systems

MEMPSODE in a nutshell

First presented in [Voglis et al., 2012]

- Global optimization software tool
- Implements a memetic algorithm
  - Synergy: local and population based global optimization
  - Exploration phase: PSO or DE
  - Exploitation phase: Merlin local optimization environment
- Standalone executable or user callable interface
- Fully parameterizable
- Accompanied by many test functions

Merlin

Steepest descent, conjugate-gradient, Quasi–Newt methods etc.

MEMPSODE in a nutshell

First presented in [Voglis et al., 2012]

- Global optimization software tool
- Implements a memetic algorithm
  - Synergy: local and population based global optimization
  - Exploration phase: PSO or DE
  - Exploitation phase: Merlin local optimization environment
- Standalone executable or user callable interface
- Fully parameterizable
- Accompanied by many test functions

Standalone

```
mempsode -a pso -d 20 -s 30 -f 10000 -l
...
```

MEMPSODE in a nutshell

First presented in [Voglis et al., 2012]

- Global optimization software tool
- Implements a memetic algorithm
  - Synergy: local and population based global optimization
  - Exploration phase: PSO or DE
  - Exploitation phase: Merlin local optimization environment
- Standalone executable or user callable interface
- Fully parameterizable
- Accompanied by many test functions

User callable

```
init_mempsode();
set_mempsode_iparam("algorithm", 1);
set_mempsode_iparam("dimension", 9);
set_mempsode_iparam("swarm-size", 100);
set_mempsode_iparam("max-fun-evals",
50000);
mempsode();
get_mempsode_min("minval", &val);
```

Unified PSO

- Generalization of PSO
- Combination of local and global best velocity updates
  - Local best: ring topology, each particle communicates only with its mates with adjacent indices
  - Global best: the whole swarm constitutes a single neighborhood, new information (best position) is immediately advertised to every single particle
- Update equations, $u \in [0, 1]$ is a unification factor.

Global best and local best velocity updates

$$G_{ij}^{(t+1)} = \chi \left[v_{ij}^{(t)} + c_1 \mathcal{R}_1 \left(p_{ij}^{(t)} - x_{ij}^{(t)} \right) + c_2 \mathcal{R}_2 \left(p_{gj}^{(t)} - x_{ij}^{(t)} \right) \right],$$

$$L_{ij}^{(t+1)} = \chi \left[v_{ij}^{(t)} + c_1 \mathcal{R}_1 \left(p_{ij}^{(t)} - x_{ij}^{(t)} \right) + c_2 \mathcal{R}_2 \left(p_{g_i j}^{(t)} - x_{ij}^{(t)} \right) \right]$$

Position update

$$\begin{aligned} U_{ij}^{(t+1)} &= u\, G_{ij}^{(t+1)} + (1-u)\, L_{ij}^{(t+1)}, \\ x_{ij}^{(t+1)} &= x_{ij}^{(t)} + U_{ij}^{(t+1)}, i = 1, 2, \ldots, N, \qquad j = 1, 2, \ldots, n. \end{aligned}$$

Unified PSO

- Generalization of PSO
- Combination of local and global best velocity updates
  - Local best: ring topology, each particle communicates only with its mates with adjacent indices
  - Global best: the whole swarm constitutes a single neighborhood, new information (best position) is immediately advertised to every single particle
- Update equations, $u \in [0,1]$ is a unification factor.

Global best and local best velocity updates

$$G_{ij}^{(t+1)} = \chi \left[v_{ij}^{(t)} + c_1 \mathcal{R}_1 \left(p_{ij}^{(t)} - x_{ij}^{(t)} \right) + c_2 \mathcal{R}_2 \left(p_{gj}^{(t)} - x_{ij}^{(t)} \right) \right],$$

$$L_{ij}^{(t+1)} = \chi \left[v_{ij}^{(t)} + c_1 \mathcal{R}_1 \left(p_{ij}^{(t)} - x_{ij}^{(t)} \right) + c_2 \mathcal{R}_2 \left(p_{g_i j}^{(t)} - x_{ij}^{(t)} \right) \right]$$

Position update

$$\begin{aligned} U_{ij}^{(t+1)} &= u\, G_{ij}^{(t+1)} + (1-u)\, L_{ij}^{(t+1)}, \\ x_{ij}^{(t+1)} &= x_{ij}^{(t)} + U_{ij}^{(t+1)}, i = 1, 2, \ldots, N, \qquad j = 1, 2, \ldots, n. \end{aligned}$$

Unified PSO

- Generalization of PSO
- Combination of local and global best velocity updates
  - Local best: ring topology, each particle communicates only with its mates with adjacent indices
  - Global best: the whole swarm constitutes a single neighborhood, new information (best position) is immediately advertised to every single particle
- Update equations, $u \in [0,1]$ is a unification factor.

Global best and local best velocity updates

$$G_{ij}^{(t+1)} = \chi \left[v_{ij}^{(t)} + c_1 \mathcal{R}_1 \left(p_{ij}^{(t)} - x_{ij}^{(t)} \right) + c_2 \mathcal{R}_2 \left(p_{gj}^{(t)} - x_{ij}^{(t)} \right) \right],$$

$$L_{ij}^{(t+1)} = \chi \left[v_{ij}^{(t)} + c_1 \mathcal{R}_1 \left(p_{ij}^{(t)} - x_{ij}^{(t)} \right) + c_2 \mathcal{R}_2 \left(p_{g_i j}^{(t)} - x_{ij}^{(t)} \right) \right]$$

Position update

$$\begin{aligned} U_{ij}^{(t+1)} &= u\, G_{ij}^{(t+1)} + (1-u)\, L_{ij}^{(t+1)}, \\ x_{ij}^{(t+1)} &= x_{ij}^{(t)} + U_{ij}^{(t+1)}, i = 1, 2, \ldots, N, \qquad j = 1, 2, \ldots, n. \end{aligned}$$

Unified PSO

- Generalization of PSO
- Combination of local and global best velocity updates
  - Local best: ring topology, each particle communicates only with its mates with adjacent indices
  - Global best: the whole swarm constitutes a single neighborhood, new information (best position) is immediately advertised to every single particle
- Update equations, $u \in [0, 1]$ is a unification factor.

Global best and local best velocity updates

$$G_{ij}^{(t+1)} = \chi \left[v_{ij}^{(t)} + c_1 \mathcal{R}_1 \left(p_{ij}^{(t)} - x_{ij}^{(t)} \right) + c_2 \mathcal{R}_2 \left(p_{gj}^{(t)} - x_{ij}^{(t)} \right) \right],$$

$$L_{ij}^{(t+1)} = \chi \left[v_{ij}^{(t)} + c_1 \mathcal{R}_1 \left(p_{ij}^{(t)} - x_{ij}^{(t)} \right) + c_2 \mathcal{R}_2 \left(p_{g_i j}^{(t)} - x_{ij}^{(t)} \right) \right]$$

Position update

$$\begin{aligned} U_{ij}^{(t+1)} &= u\, G_{ij}^{(t+1)} + (1-u)\, L_{ij}^{(t+1)}, \\ x_{ij}^{(t+1)} &= x_{ij}^{(t)} + U_{ij}^{(t+1)}, i = 1, 2, \ldots, N, \qquad j = 1, 2, \ldots, n. \end{aligned}$$

Unified PSO

- Generalization of PSO
- Combination of local and global best velocity updates
  - Local best: ring topology, each particle communicates only with its mates with adjacent indices
  - Global best: the whole swarm constitutes a single neighborhood, new information (best position) is immediately advertised to every single particle
- Update equations, $u \in [0, 1]$ is a unification factor.

Global best and local best velocity updates

$$G_{ij}^{(t+1)} = \chi \left[v_{ij}^{(t)} + c_1 \mathcal{R}_1 \left(p_{ij}^{(t)} - x_{ij}^{(t)} \right) + c_2 \mathcal{R}_2 \left(p_{gj}^{(t)} - x_{ij}^{(t)} \right) \right],$$

$$L_{ij}^{(t+1)} = \chi \left[v_{ij}^{(t)} + c_1 \mathcal{R}_1 \left(p_{ij}^{(t)} - x_{ij}^{(t)} \right) + c_2 \mathcal{R}_2 \left(p_{g_i j}^{(t)} - x_{ij}^{(t)} \right) \right]$$

Position update

$$U_{ij}^{(t+1)} = u\, G_{ij}^{(t+1)} + (1-u)\, L_{ij}^{(t+1)},$$

$$x_{ij}^{(t+1)} = x_{ij}^{(t)} + U_{ij}^{(t+1)}, i = 1, 2, \ldots, N, \qquad j = 1, 2, \ldots, n.$$

Memetic algorithms

- Hybrid schemes that exploit the advantages of both approaches
- Answer the *fundamental memetic questions* [Parsopoulos and Vrahatis, 2010], are:
  (a) *Where* should the LS procedures be applied.
  (b) *When* should the LS procedures be applied.
  (c) *How much of the budget* should the LS procedures spend.
- We implement the memetic strategies presented in [Petalas et al., 2007] for PSO:
  S1: LS is applied only on the overall best position, p_g, of the swarm.
  S2: LS is applied on each localy best position, p_i, $i = 1, 2, \ldots, N$, with a prescribed fixed probability, $\rho \in (0, 1]$.
  S3: LS is applied both on the best position, p_g, as well as on some randomly selected localy best positions, p_i, $i \in \{1, 2, \ldots, N\}$.

Memetic algorithms

- Hybrid schemes that exploit the advantages of both approaches
- Answer the *fundamental memetic questions* [Parsopoulos and Vrahatis, 2010], are:
  (a) *Where* should the LS procedures be applied.
  (b) *When* should the LS procedures be applied.
  (c) *How much of the budget* should the LS procedures spend.
- We implement the memetic strategies presented in [Petalas et al., 2007] for PSO:
  S1: LS is applied only on the overall best position, p_g, of the swarm.
  S2: LS is applied on each localy best position, p_i, $i = 1, 2, \ldots, N$, with a prescribed fixed probability, $\rho \in (0, 1]$.
  S3: LS is applied both on the best position, p_g, as well as on some randomly selected localy best positions, p_i, $i \in \{1, 2, \ldots, N\}$.

Memetic algorithms

- Hybrid schemes that exploit the advantages of both approaches
- Answer the *fundamental memetic questions* [Parsopoulos and Vrahatis, 2010], are:
  (a) *Where* should the LS procedures be applied.
  (b) *When* should the LS procedures be applied.
  (c) *How much of the budget* should the LS procedures spend.
- We implement the memetic strategies presented in [Petalas et al., 2007] for PSO:

S1: LS is applied only on the overall best position, p_g, of the swarm.
S2: LS is applied on each localy best position, p_i, $i = 1, 2, \ldots, N$, with a prescribed fixed probability, $\rho \in (0, 1]$.
S3: LS is applied both on the best position, p_g, as well as on some randomly selected localy best positions, p_i, $i \in \{1, 2, \ldots, N\}$.

Merlin Optimization Environment

`http://merlin.cs.uoi.gr`

- General purpose optimization package
- Deterministic - sequential algorithms for single local optimum
- Command line mode and user callable subroutine mode
- *Gradient–based algorithms*
  - BFGS and DFP with line search
  - BFGS with trust region
  - Levenberg–Marquardt
  - Powell's TOLMIN algorithm (BFGS with different matrix factorization)
  - Conjugate Gradients
- *Gradient–free algorithms*
  - Nonlinear Simplex method of Nelder and Mead.
  - Alternating directions algorithm.
- *Custom algorithms*
  - Hooke and Jeeves
  - Random search
  - Torczon's MDS

Merlin Optimization Environment

`http://merlin.cs.uoi.gr`

- General purpose optimization package
- Deterministic - sequential algorithms for single local optimum
- Command line mode and user callable subroutine mode
- *Gradient–based algorithms*
  - BFGS and DFP with line search
  - BFGS with trust region
  - Levenberg–Marquardt
  - Powell's TOLMIN algorithm (BFGS with different matrix factorization)
  - Conjugate Gradients
- *Gradient–free algorithms*
  - Nonlinear Simplex method of Nelder and Mead.
  - Alternating directions algorithm.
- *Custom algorithms*
  - Hooke and Jeeves
  - Random search
  - Torczon's MDS

Merlin Optimization Environment

`http://merlin.cs.uoi.gr`

- General purpose optimization package
- Deterministic - sequential algorithms for single local optimum
- Command line mode and user callable subroutine mode
- *Gradient–based algorithms*
  - BFGS and DFP with line search
  - BFGS with trust region
  - Levenberg–Marquardt
  - Powell's TOLMIN algorithm (BFGS with different matrix factorization)
  - Conjugate Gradients
- *Gradient–free algorithms*
  - Nonlinear Simplex method of Nelder and Mead.
  - Alternating directions algorithm.
- *Custom algorithms*
  - Hooke and Jeeves
  - Random search
  - Torczon's MDS

Merlin Optimization Environment

`http://merlin.cs.uoi.gr`

- General purpose optimization package
- Deterministic - sequential algorithms for single local optimum
- Command line mode and user callable subroutine mode
- *Gradient–based algorithms*
  - BFGS and DFP with line search
  - BFGS with trust region
  - Levenberg–Marquardt
  - Powell's TOLMIN algorithm (BFGS with different matrix factorization)
  - Conjugate Gradients
- *Gradient–free algorithms*
  - Nonlinear Simplex method of Nelder and Mead.
  - Alternating directions algorithm.
- *Custom algorithms*
  - Hooke and Jeeves
  - Random search
  - Torczon's MDS

Merlin Optimization Environment

`http://merlin.cs.uoi.gr`

- General purpose optimization package
- Deterministic - sequential algorithms for single local optimum
- Command line mode and user callable subroutine mode
- *Gradient–based algorithms*
  - BFGS and DFP with line search
  - BFGS with trust region
  - Levenberg–Marquardt
  - Powell's TOLMIN algorithm (BFGS with different matrix factorization)
  - Conjugate Gradients
- *Gradient–free algorithms*
  - Nonlinear Simplex method of Nelder and Mead.
  - Alternating directions algorithm.
- *Custom algorithms*
  - Hooke and Jeeves
  - Random search
  - Torczon's MDS

Merlin Optimization Environment

`http://merlin.cs.uoi.gr`

- General purpose optimization package
- Deterministic - sequential algorithms for single local optimum
- Command line mode and user callable subroutine mode
- *Gradient–based algorithms*
  - BFGS and DFP with line search
  - BFGS with trust region
  - Levenberg–Marquardt
  - Powell's TOLMIN algorithm (BFGS with different matrix factorization)
  - Conjugate Gradients
- *Gradient–free algorithms*
  - Nonlinear Simplex method of Nelder and Mead.
  - Alternating directions algorithm.
- *Custom algorithms*
  - Hooke and Jeeves
  - Random search
  - Torczon's MDS

Adaptive Selection (I)

Roulette wheel selection based on adaptive probabilities $\mathcal{P}^{(i)}$.

Notation

Local search pool: $\mathcal{L} = \{LS^{(1)}, LS^{(2)}, \ldots, LS^{(k)}\}$
Each local search $LS^{(i)}$

- Counter of the number of applications: $c^{(i)}$
- Score for the $j-$th application: $score_j^{(i)} = \frac{\frac{\left|\tilde{f}_j^{(i)} - \hat{f}_j^{(i)}\right|}{\left|\hat{f}_j^{(i)}\right|}}{\#fevals_j^{(i)}}$
- Average score: $\mathcal{S}^{(i)} = \frac{\sum_{j=1}^{c^{(i)}} score_j^{(i)}}{c^{(i)}}$
- Probability: $\mathcal{P}^{(i)} = \frac{\mathcal{S}^{(i)}}{\sum_{i=1}^{k} \mathcal{S}^{(i)}}$

Adaptive Selection (II)

Two-phase alternating scheme

- Training phase:
  - Collecting information (average score) for the local searches
  - Applying roulette selection with equal probabilities.
- Adaptive phase: Probabilities are adapted to the average score

Important

At the beginning of a training phase counters are reset.

Adaptive Selection(III)

```
// Evaluate Population or Apply Local search
for i = 1, 2, ..., N do
    if act_i = 0 then
        f_i ← f(x_i) // Perform FE
    else
        j ← RouletteSelection(P)
        [p_i^+, f_{p_i}^+, f_evals] ← LS^(j)(p_i) // Perform LS
        score^(j) ← (|f_{p_i} − f_{p_i}^+| / |f_{p_i}^+|) / f_evals
        ls ← ls + 1;
        if train = 1 and mod(ls, K) = 0 then
            // Entering adaptive phase
            train ← 0
        else if train = 0 and mod(ls, 3 * K) = 0 then
            // Entering training phase
            train ← 1
            // Average scores and probabilities
               are reset
            for κ = 1, 2, ..., k do
                c^(κ) ← 0;  S^(κ) ← 0;  P^(κ) ← 1/k
            end
        end
        UpdateProb(j, score, S, P, train)
    end
end
```

UpdateProb(j, score, $\mathcal{S}$, $\mathcal{P}$, train)

Input: Selected index: i ; Score for the selected: $score^{(i)}$; Score array: $\mathcal{S}$, Probability array: $\mathcal{P}$, Phase flag: *train*
Output: New score: $\mathcal{S}$, New probability: $\mathcal{P}$; New count: c

$\mathcal{S}^{(i)} \leftarrow \frac{c^{(i)}\mathcal{S}^{(i)} + \text{score}}{c^{(i)}+1}$ // Update mean $\mathcal{S}^{(i)}$
$c^{(i)} \leftarrow c^{(i)} + 1$ // Update count $c^{(i)}$
// Not in training phase, update the probabilities
if train $= 0$ **then**
    for $\kappa = 1, 2, \ldots, k$ **do**
        $\mathcal{P}^{(\kappa)} \leftarrow \frac{\mathcal{S}^{(\kappa)}}{\sum^k \mathcal{S}^{(\kappa)}}$
    end
end

Contents

1. Introduction
   - Memetic global optimization
   - Adaptive memetic global optimization
   - This year's contribution (BBOB 2013)
2. adapt-MEMPSODE description
   - Swarm Intelligent components
   - Memetic strategy
   - Local searches
   - Adaptive selection
3. Many local searches?
   - Local search mapping
   - Local search on a track
4. Experimental results
   - Experimental procedure
   - Results

Why using a pool of local searches?

Facts:

- Each local search maps a starting point to a minimum

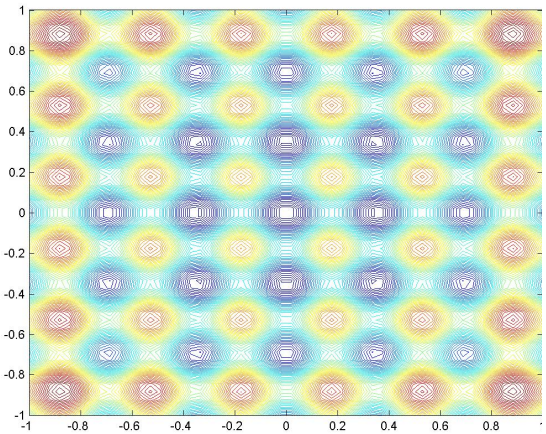
$$x^* \leftarrow LS^{(j)}(x)$$

- Each local search performs different mapping
- Each local search has similar behaviour from near points

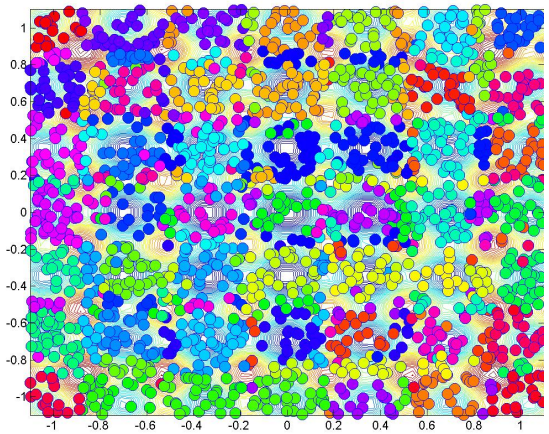
Conjecture:

- Many local searches provide *diversity* hence better exploration capabilities

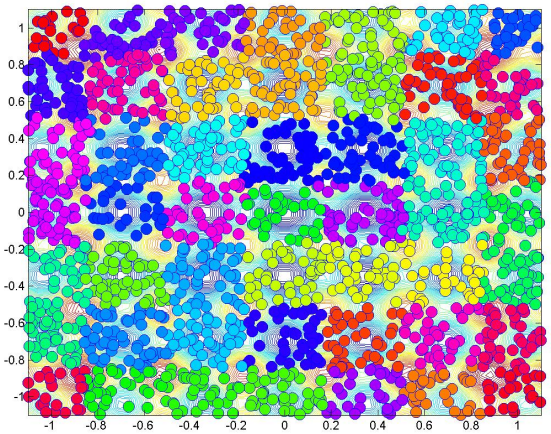
Diverse mapping



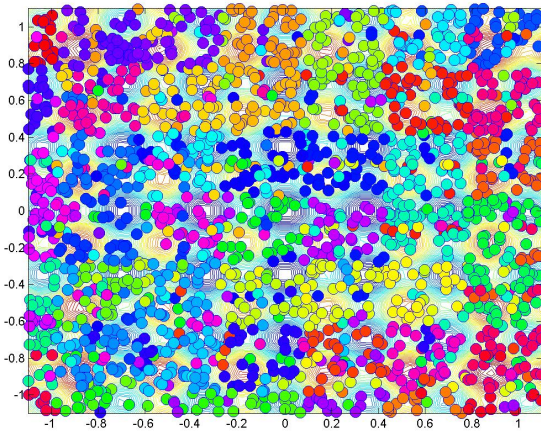
(a) BFGS line searh



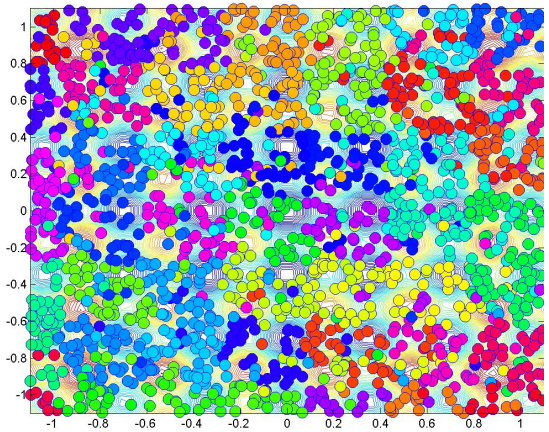
(b) BFGS trust region



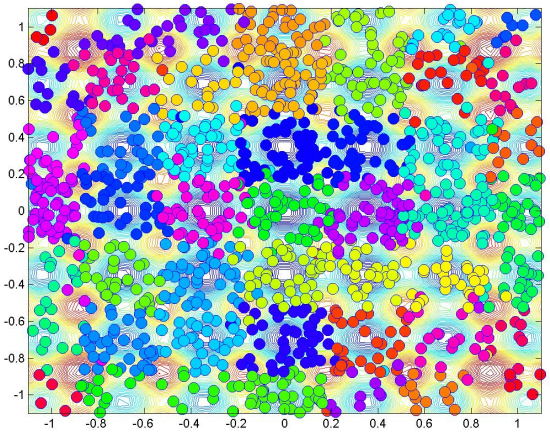
(c) Nelder-Mead



(d) Hooke-Jeeves

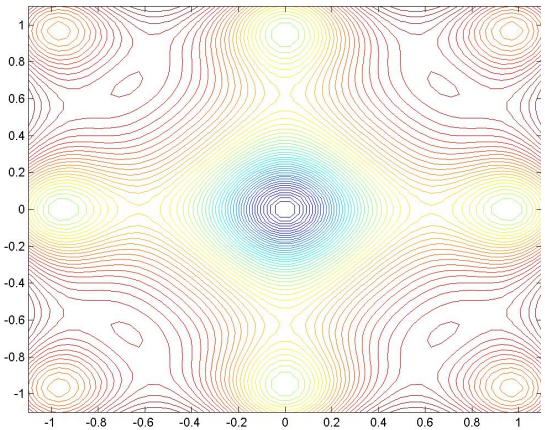


(e) Torczon's simplex

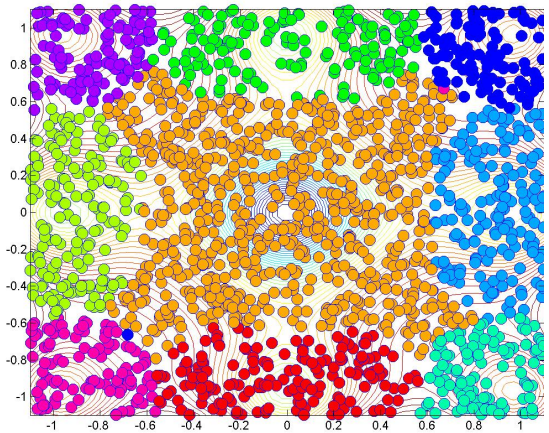


(f) Random

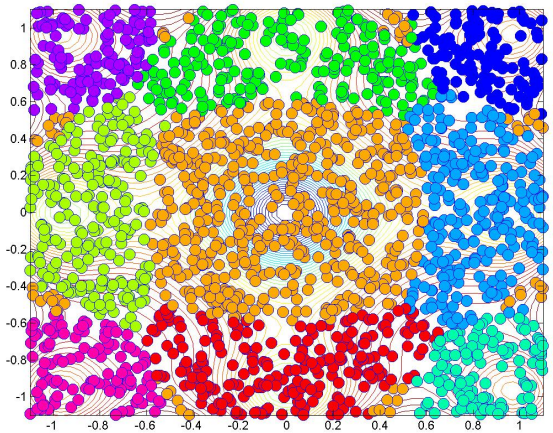
Diverse mapping



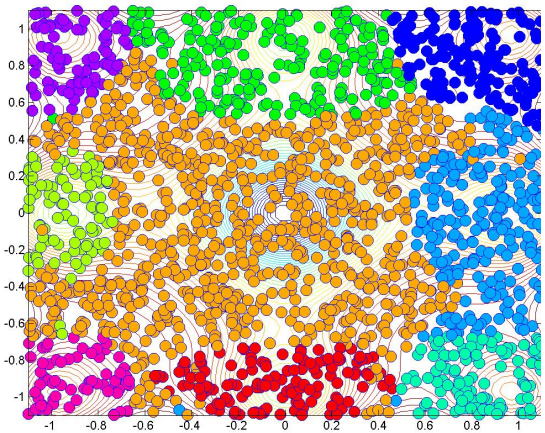
(g) BFGS line searh



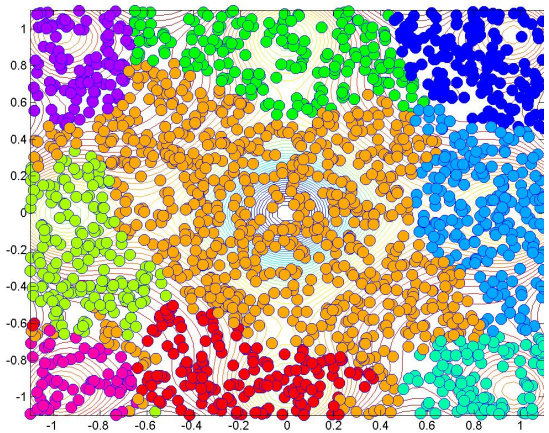
(h) BFGS trust region



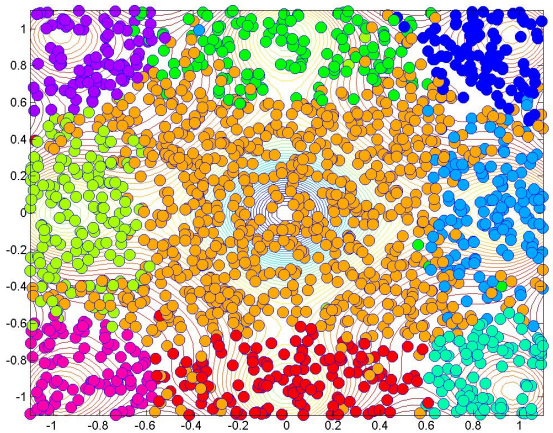
(i) Nelder-Mead



(j) Hooke-Jeeves

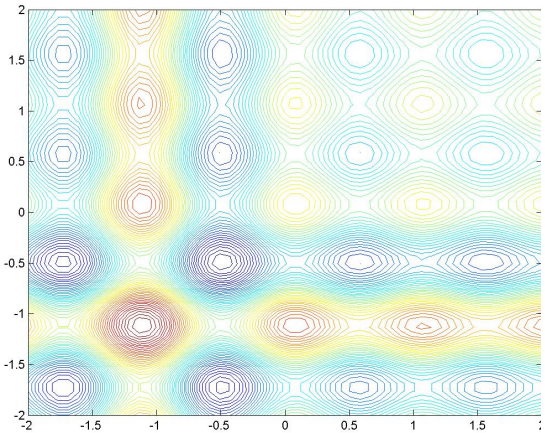


(k) Torczon's simplex

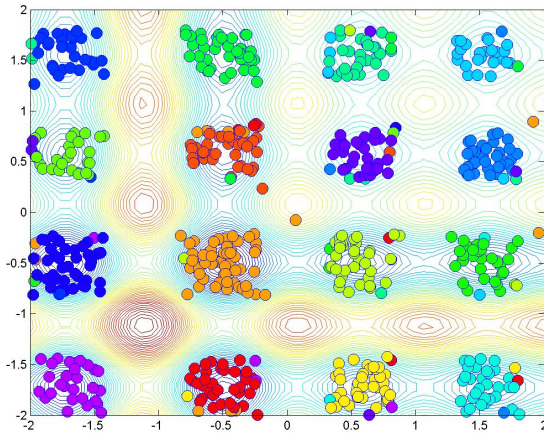


(l) Random

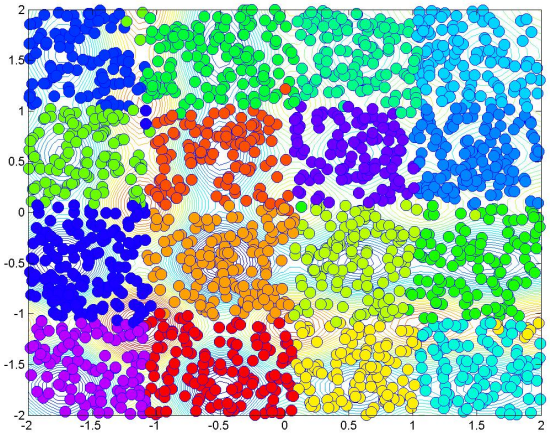
Diverse mapping



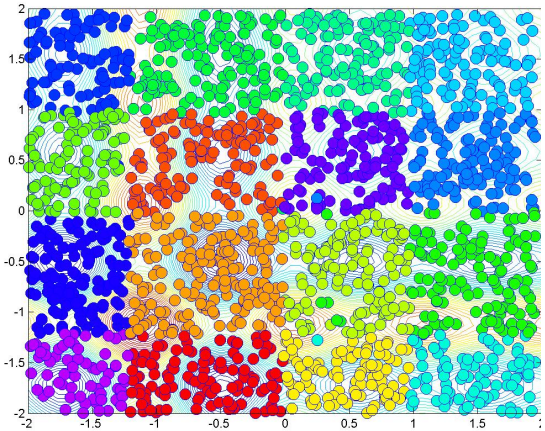
(m) BFGS line searh



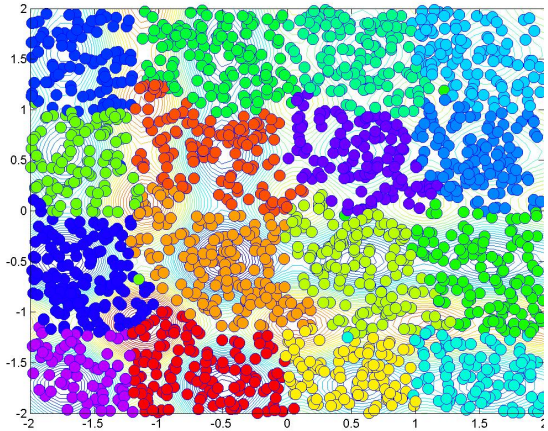
(n) BFGS trust region



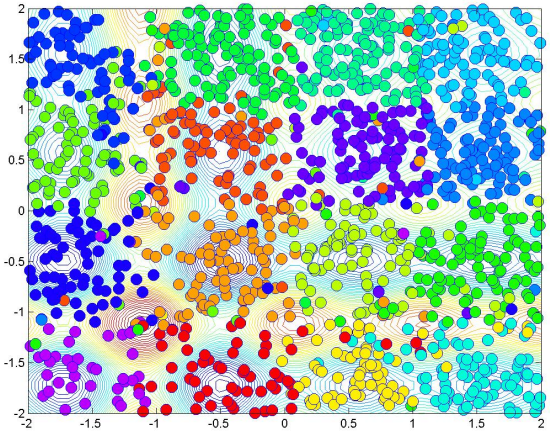
(o) Nelder-Mead



(p) Hooke-Jeeves

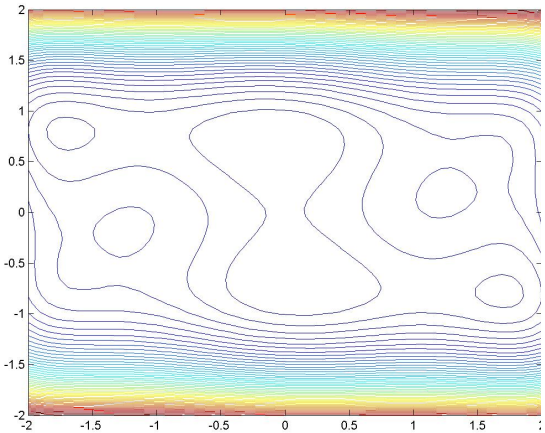


(q) Torczon's simplex

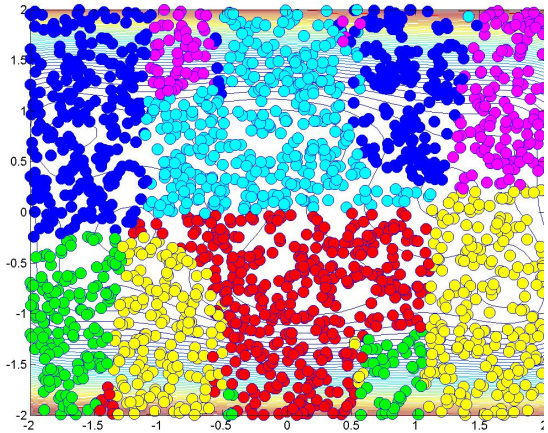


(r) Random

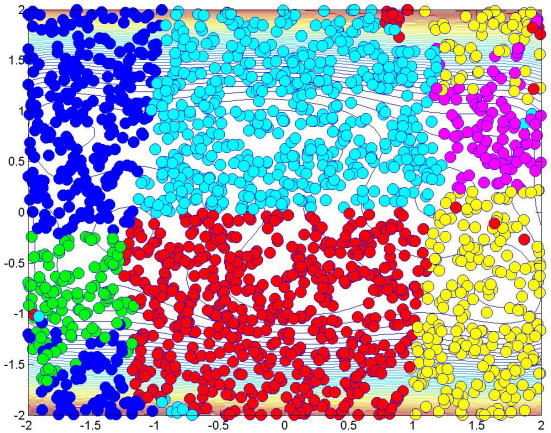
Diverse mapping



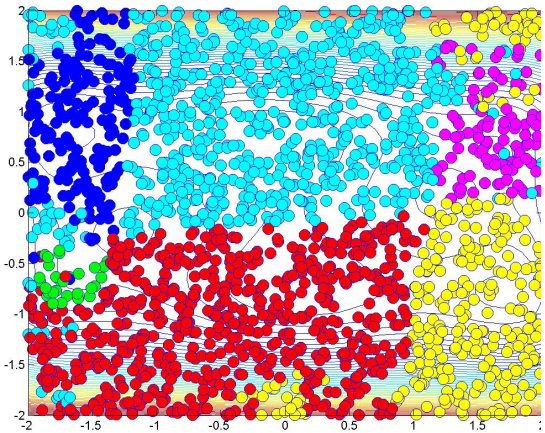
(s) BFGS line searh



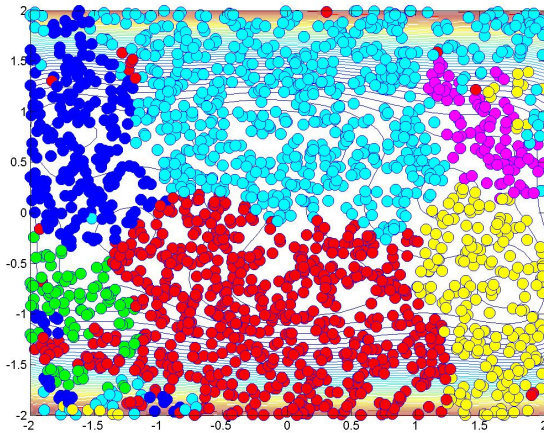
(t) BFGS trust region



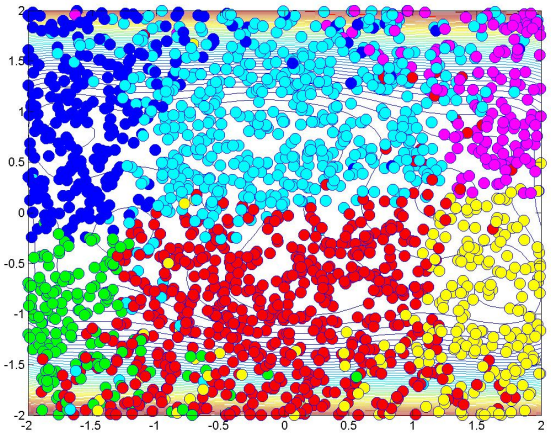
(u) Nelder-Mead



(v) Hooke-Jeeves



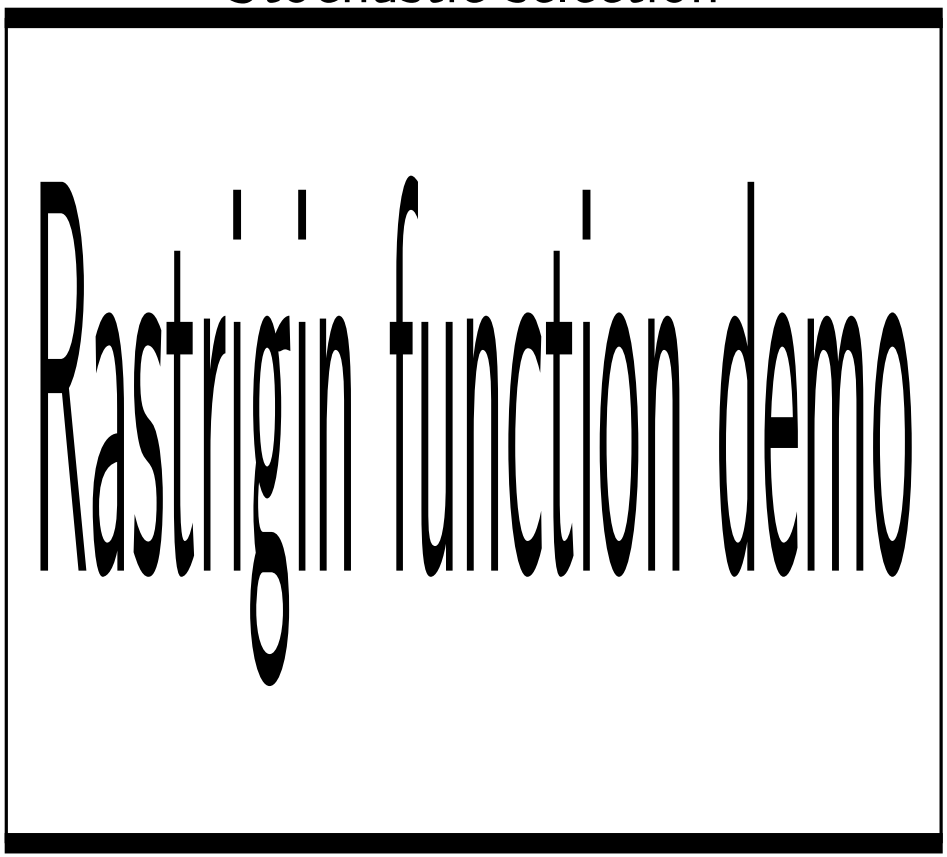
(w) Torczon's simplex



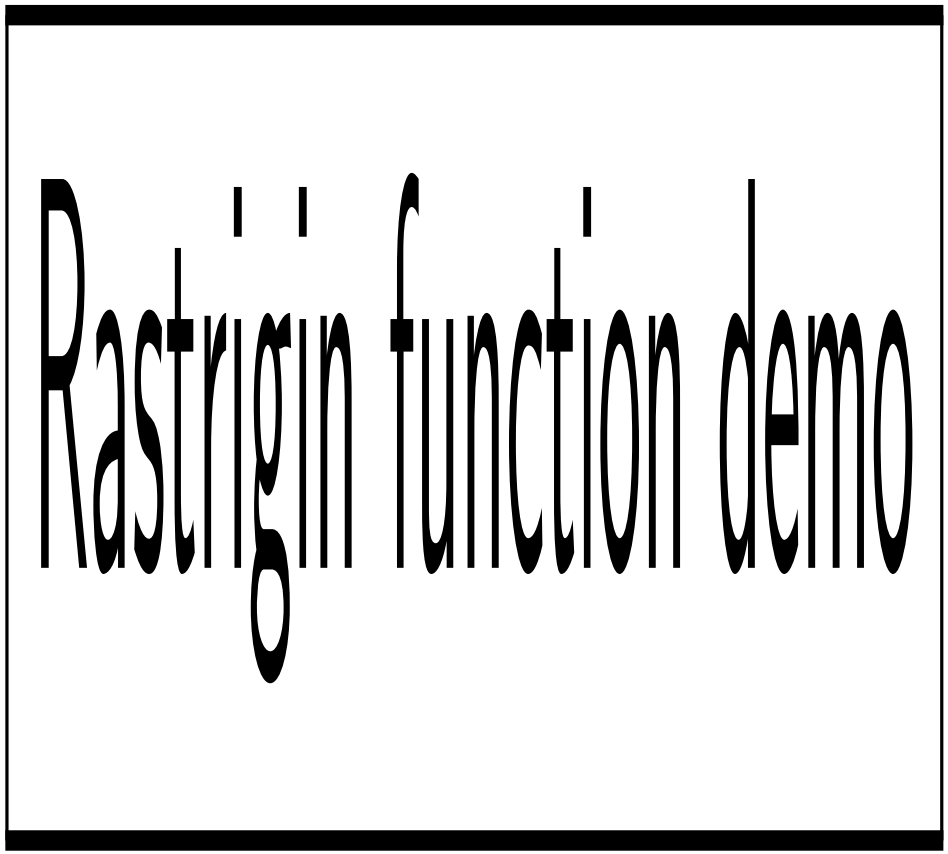
(x) Random

Illustration: local search from a track

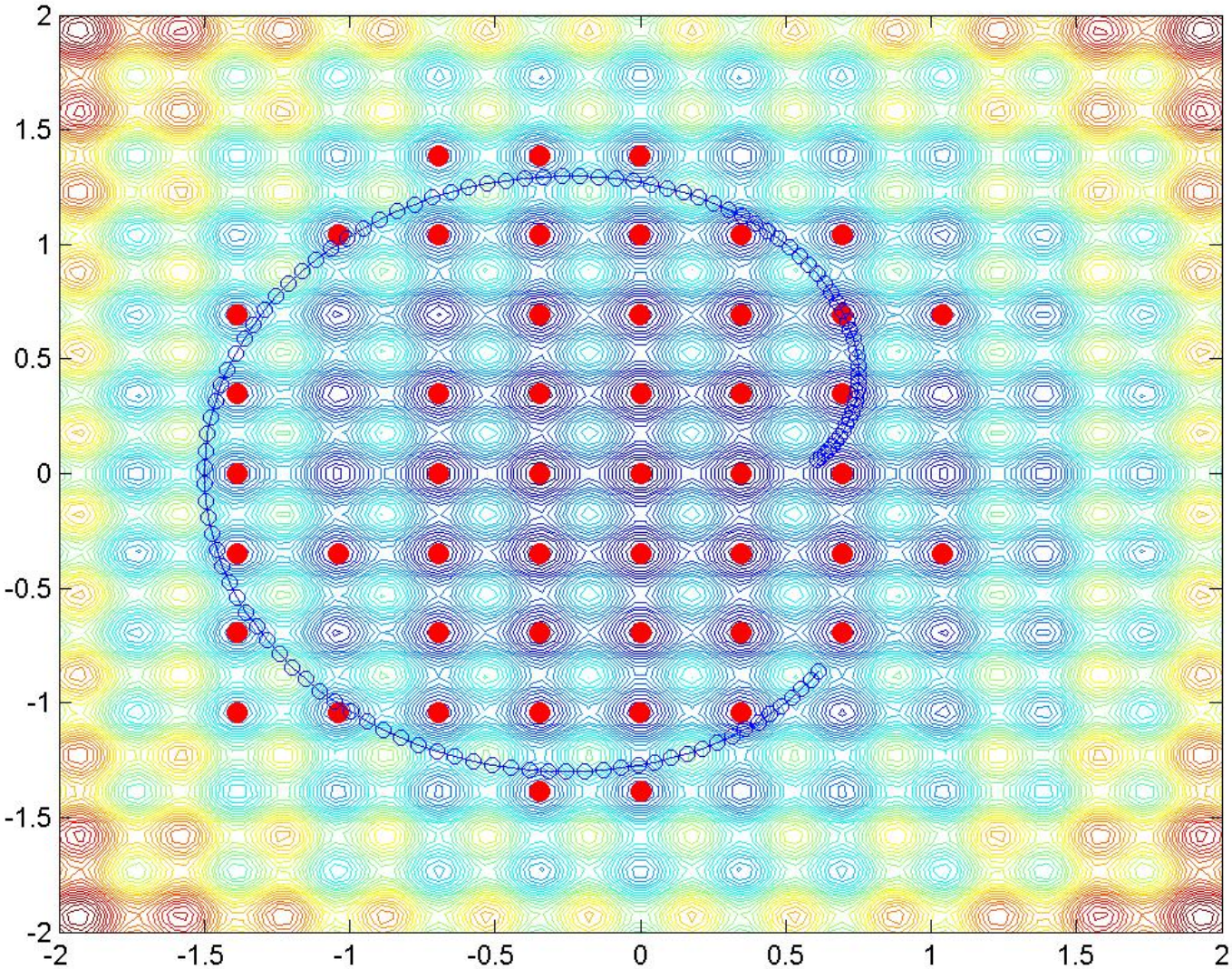
Stochastic selection



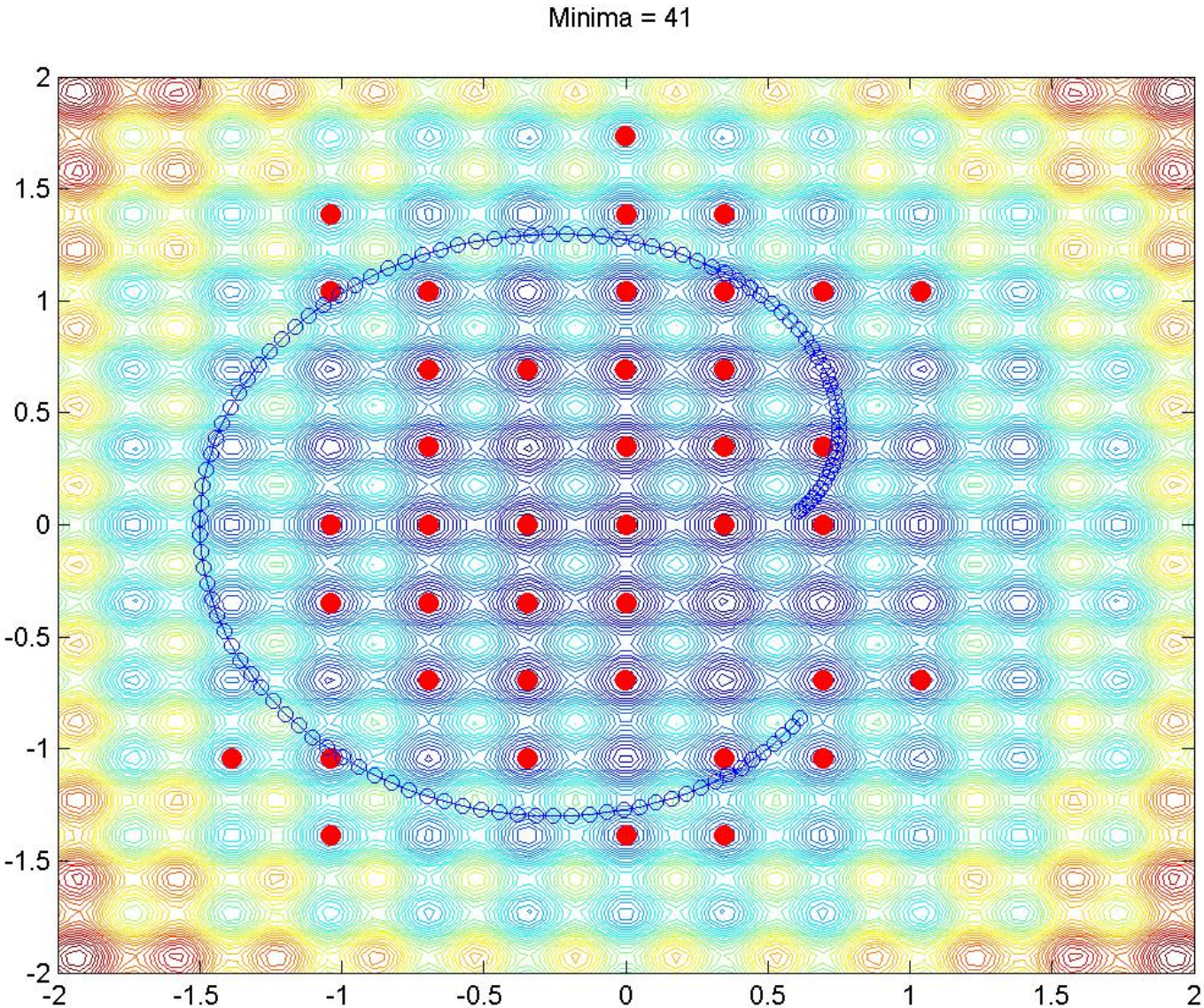
Single local search(Simplex)

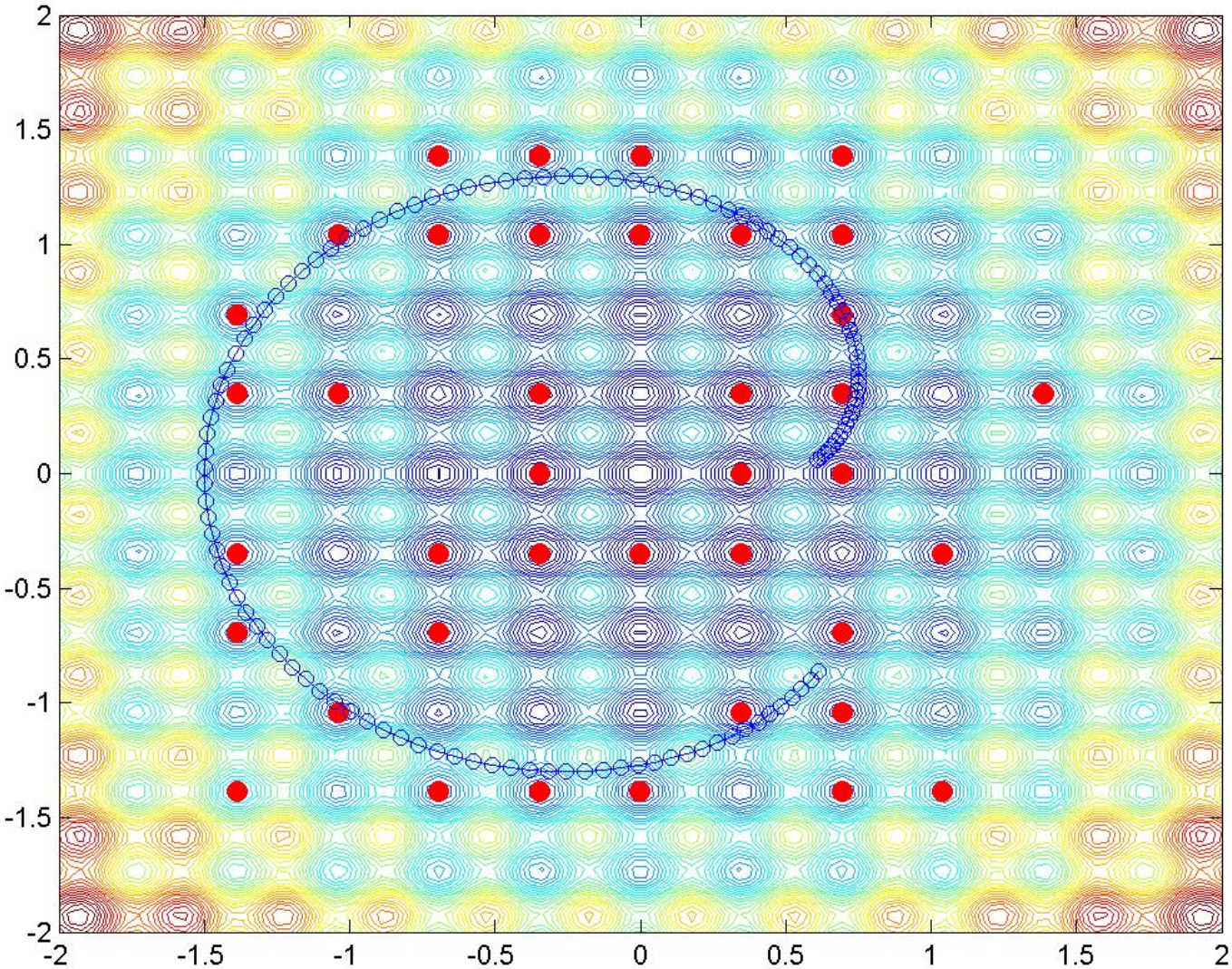


Stochastic selection



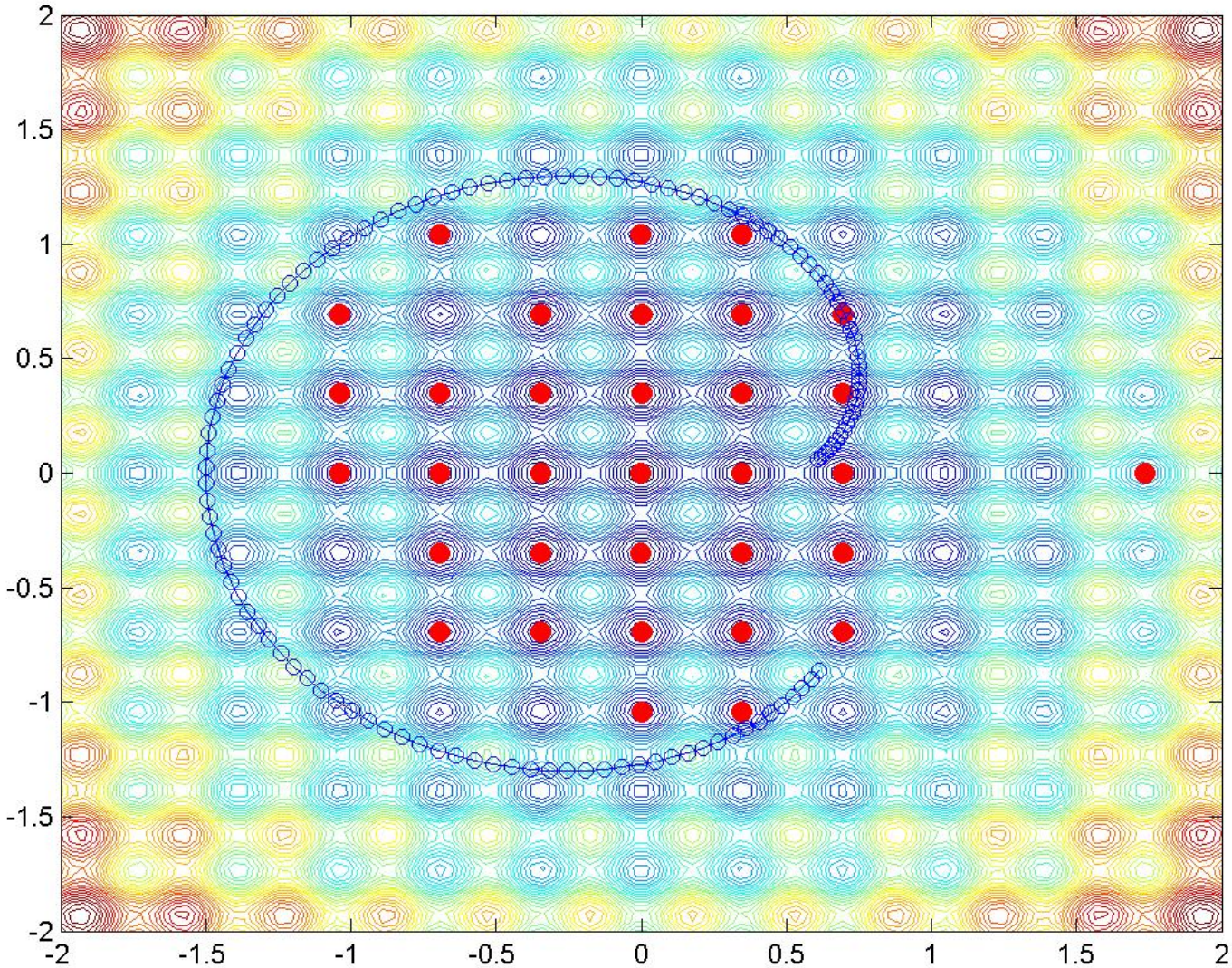
Simplex




BFGS
Minima = 39
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2

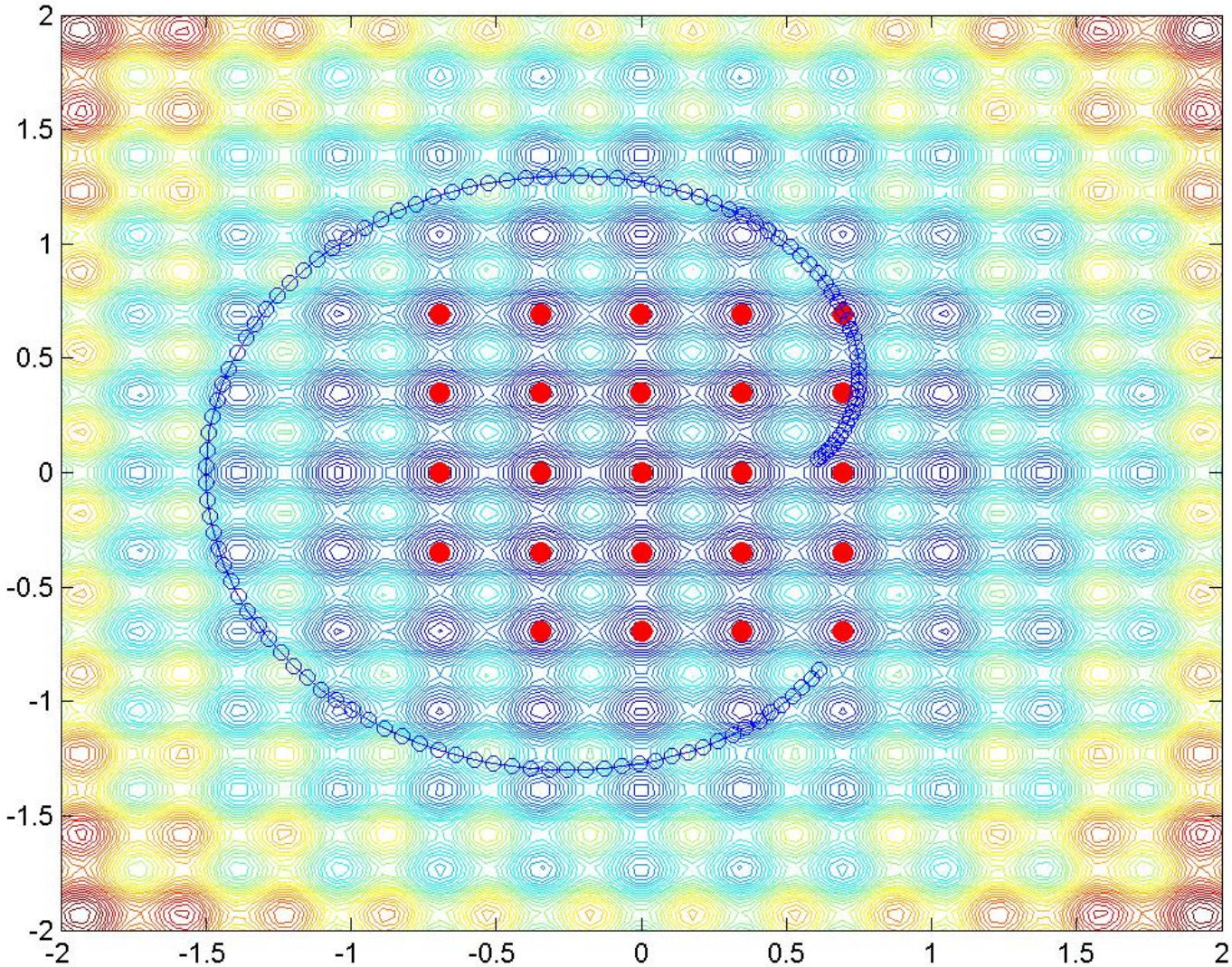
Torczon's MDS

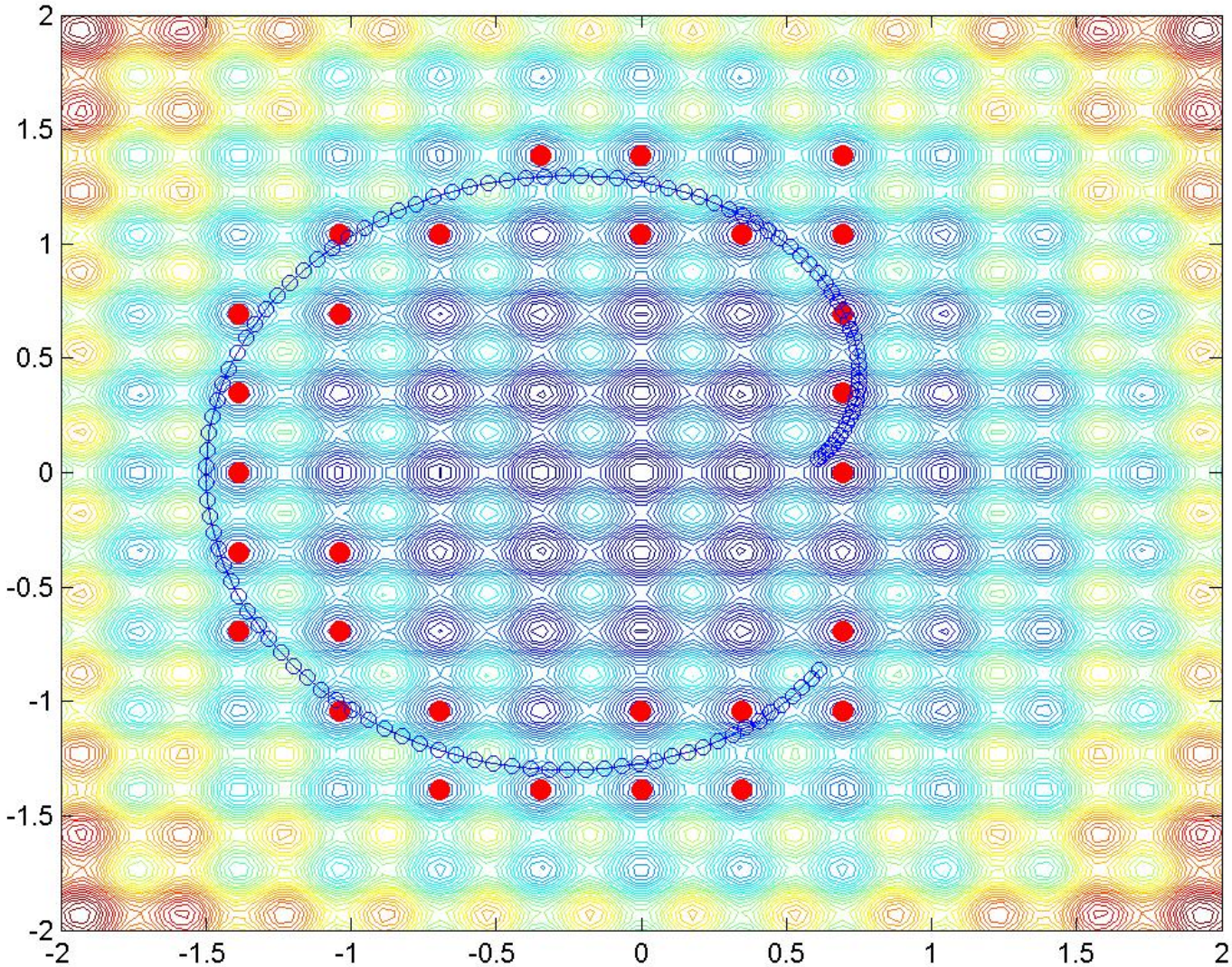
Minima = 33



Hooke-Jeeves

Minima = 24




Random
Minima = 29
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2

Contents

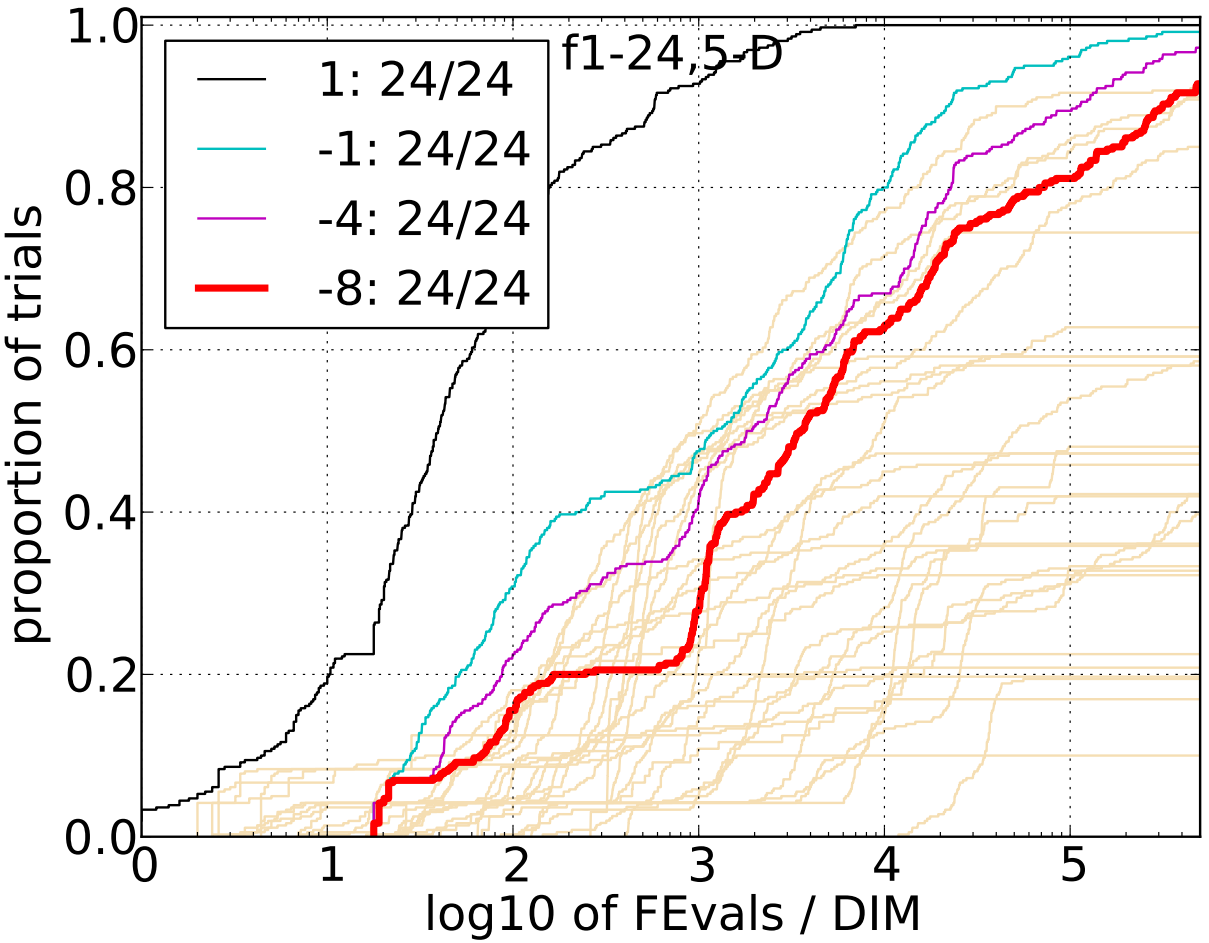
1. Introduction
   - Memetic global optimization
   - Adaptive memetic global optimization
   - This year's contribution (BBOB 2013)
2. adapt-MEMPSODE description
   - Swarm Intelligent components
   - Memetic strategy
   - Local searches
   - Adaptive selection
3. Many local searches?
   - Local search mapping
   - Local search on a track
4. Experimental results
   - Experimental procedure
   - Results

Experimental procedure

- Swarm size: 30
- Maximum function evaluations: $5 \times 10^5 \times n$
- Maximum function evaluations/local search: 2 000
- Memetic scheme: Apply local search from a best position vector with probability $\rho_i = 0.05$
- Local search pool:
  1. BFGS with line search
  2. Nelder-Mead downhill simplex
  3. Alternating directions
  4. Conjugate gradient
  5. Auto: meta–local search procedure
  6. Random search
  7. Hooke and Jeeves pattern search with discrete steps
  8. Hooke and Jeeves pattern search with line search
  9. TOLMIN: BFGS with line search

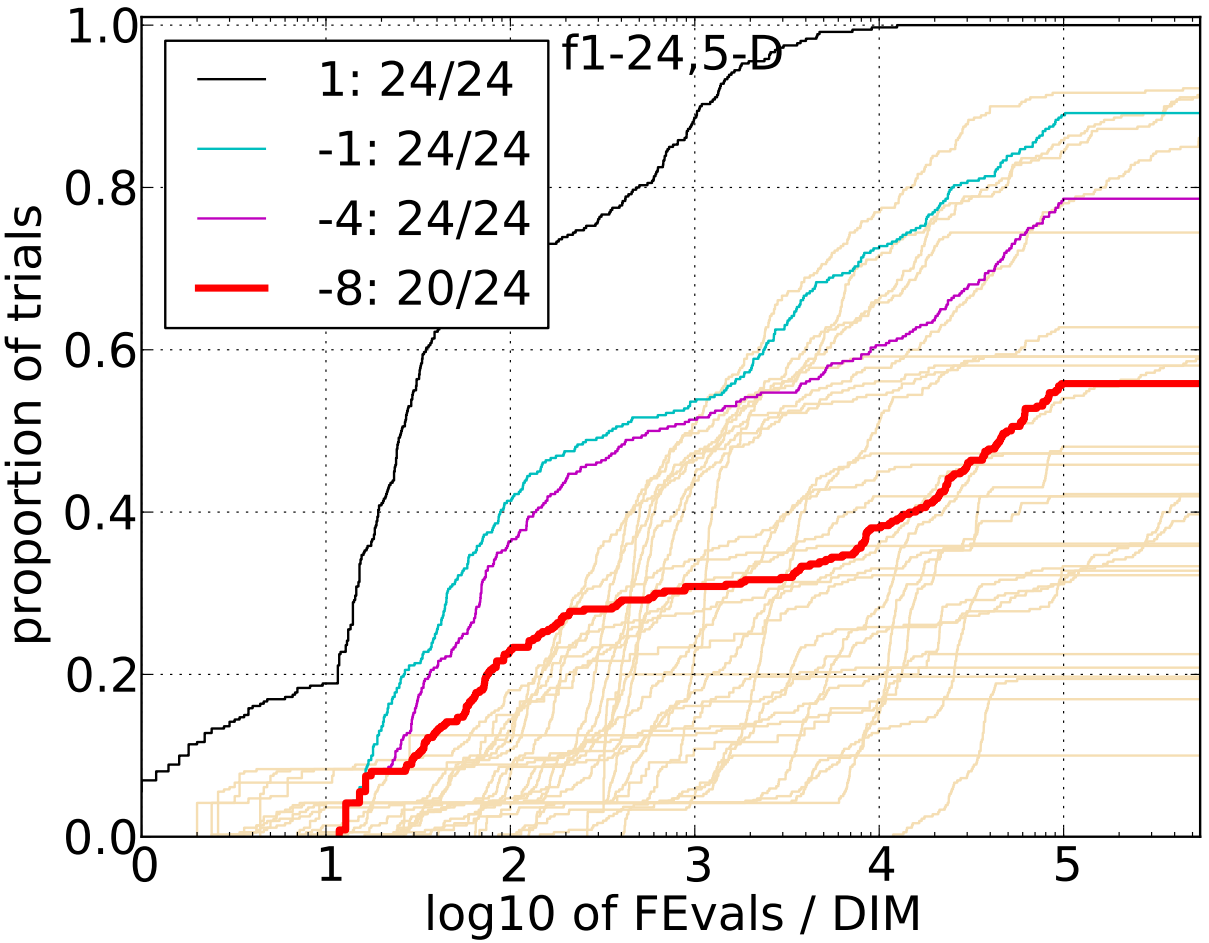
ECDF 5D all functions

This year's competition



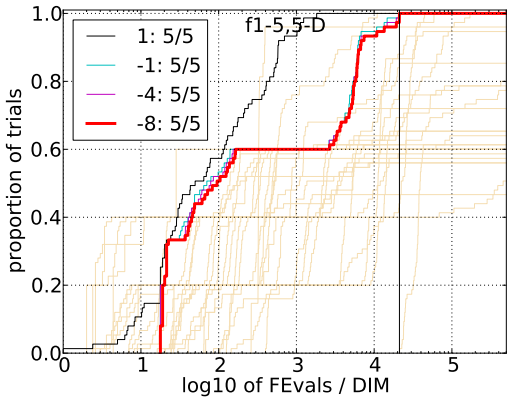
ECDF 5D all functions

BBOB 2012 competition

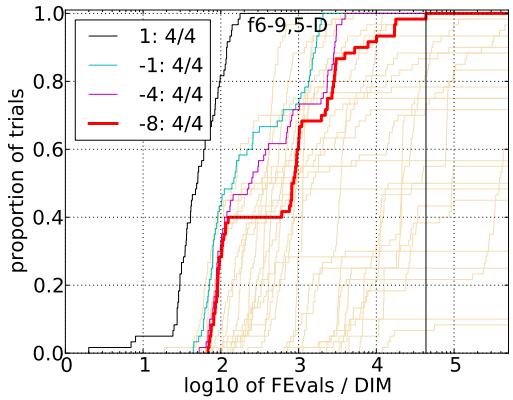


ECDF 5D function categorized

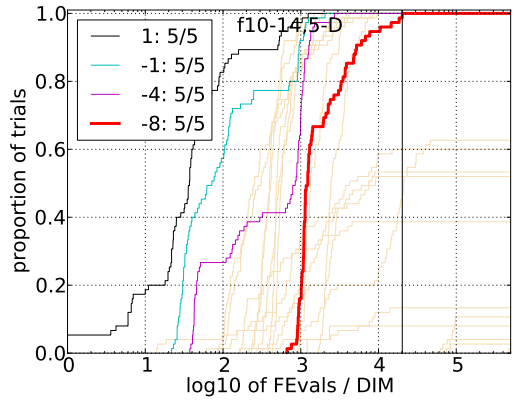
This year's competition



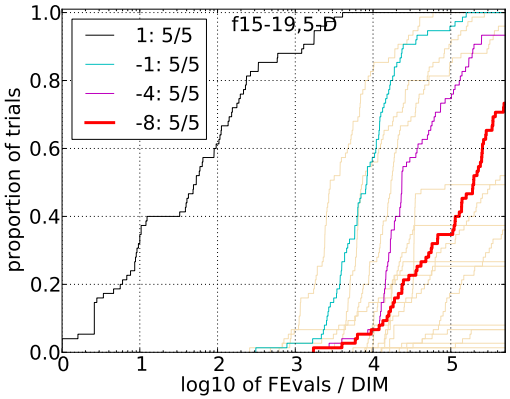
(a)



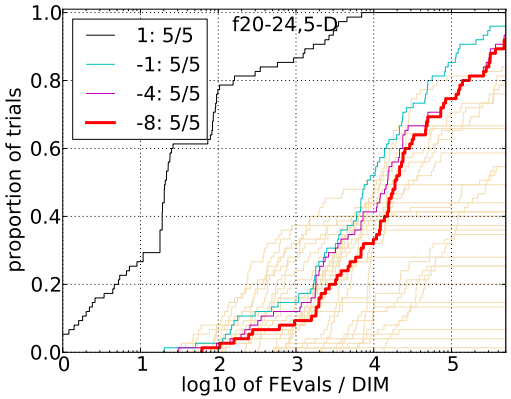
(b)



(c)



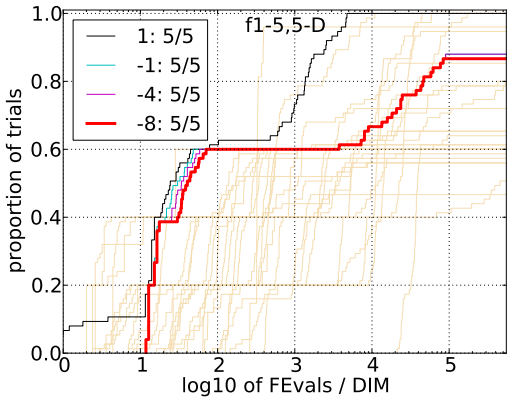
(d)



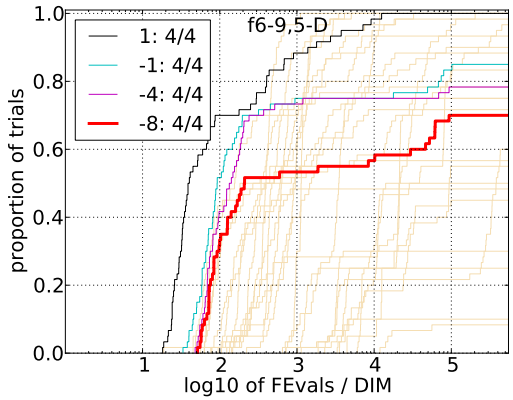
(e)

ECDF 5D function categorized

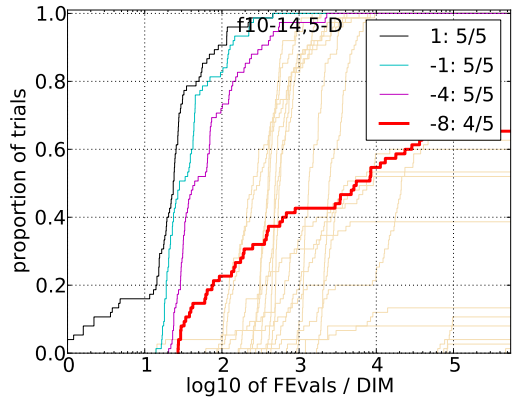
BBOB 2012 competition



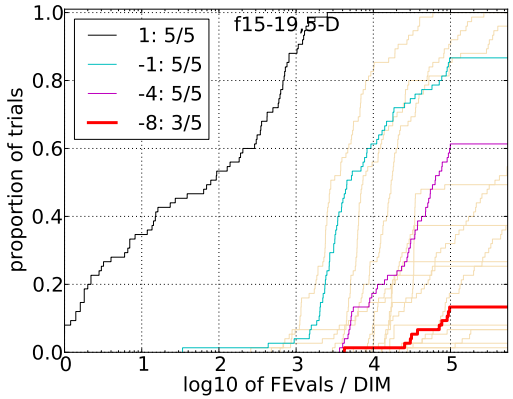
(a)



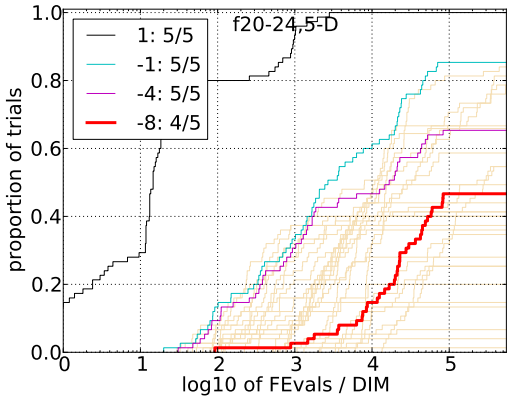
(b)



(c)



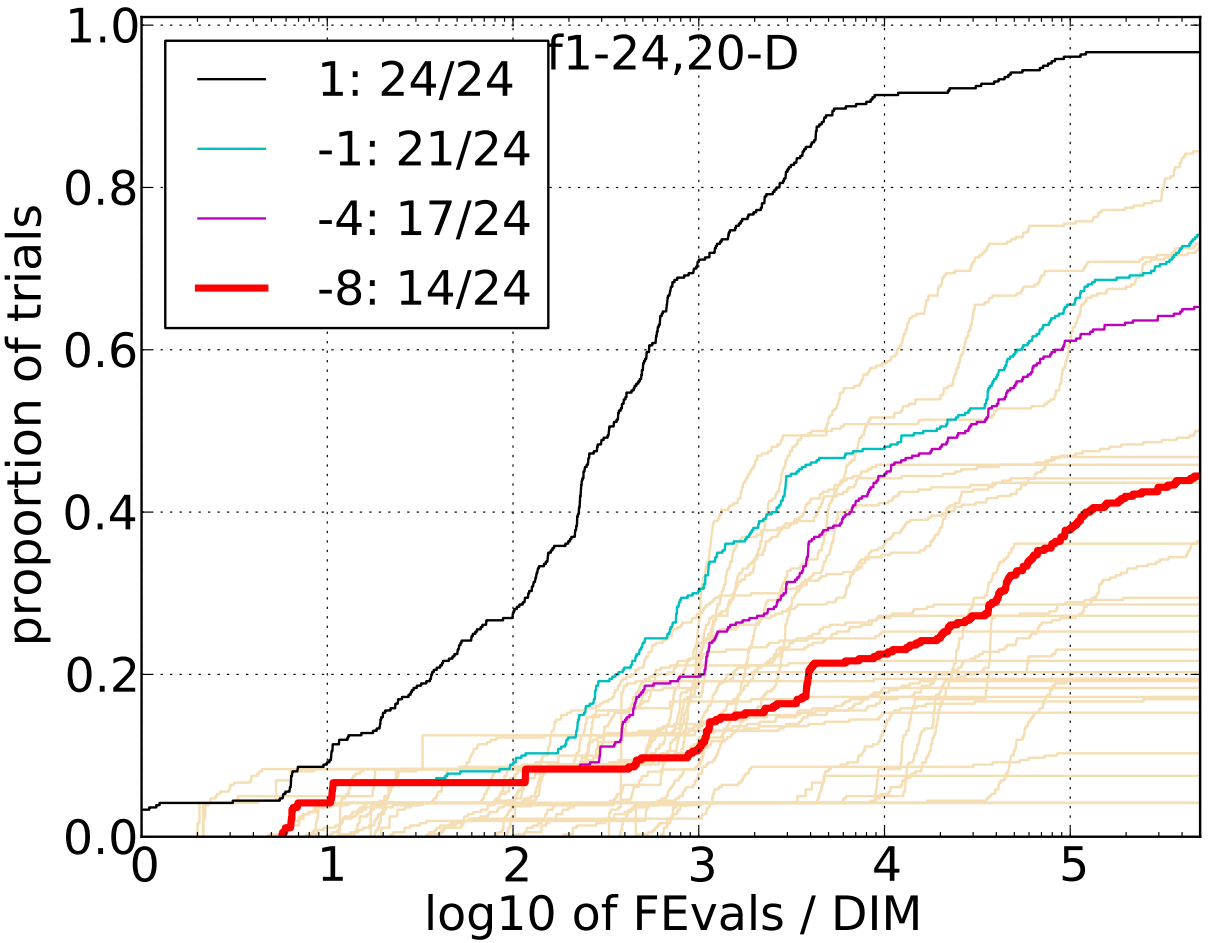
(d)



(e)

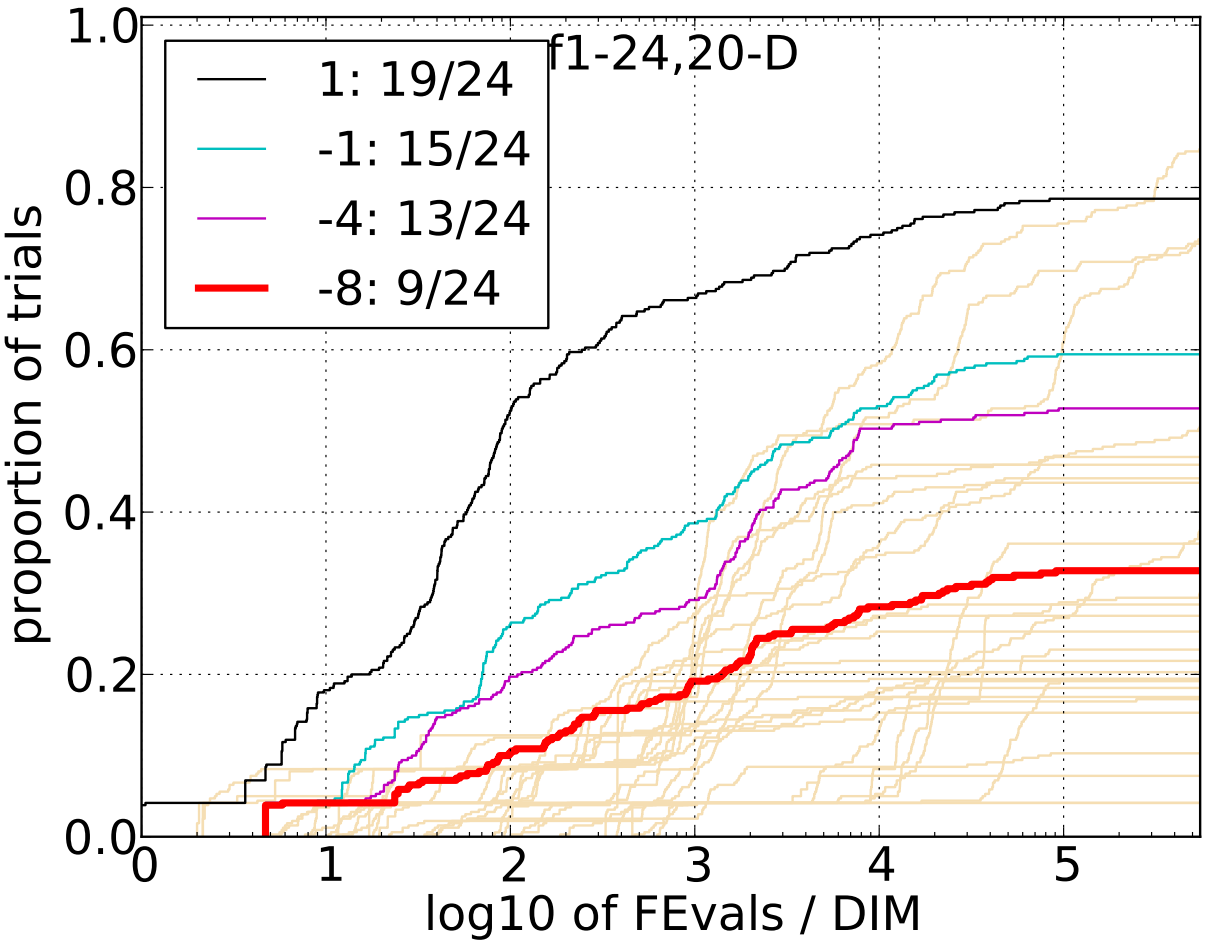
ECDF 20D all functions

This year's competition



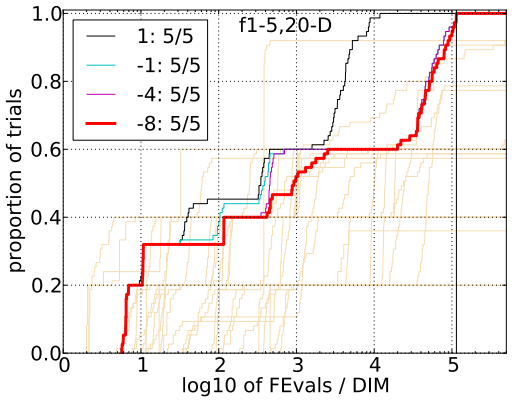
ECDF 20D all functions

BBOB 2012 competition

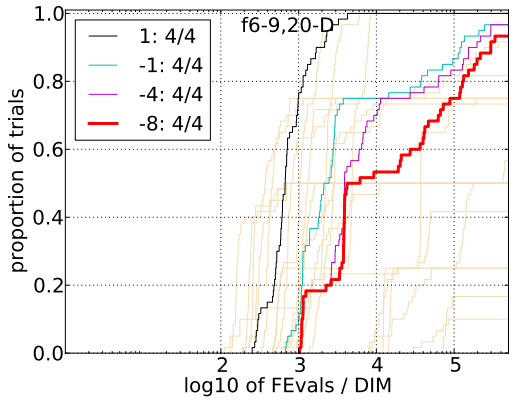


ECDF 20D function categorized

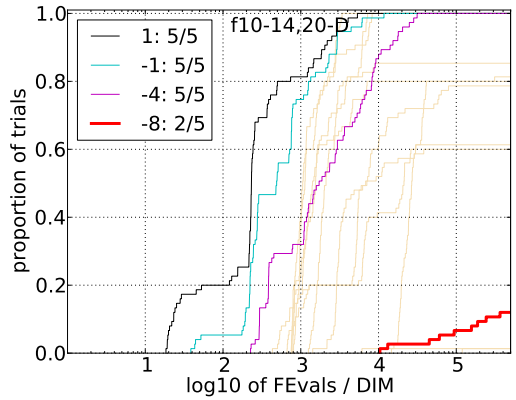
This year's competition



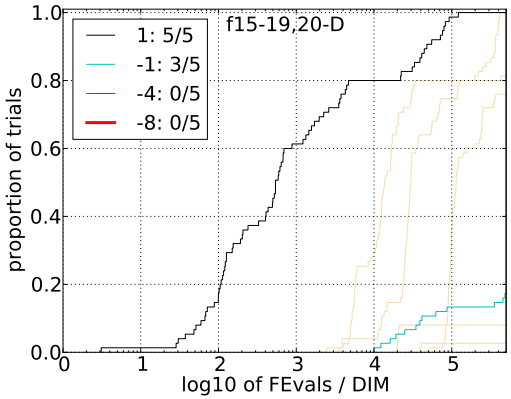
(a)



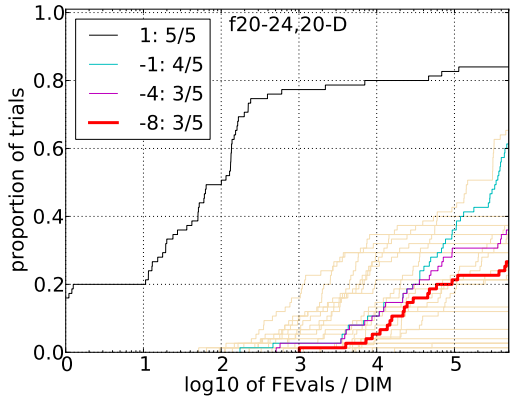
(b)



(c)



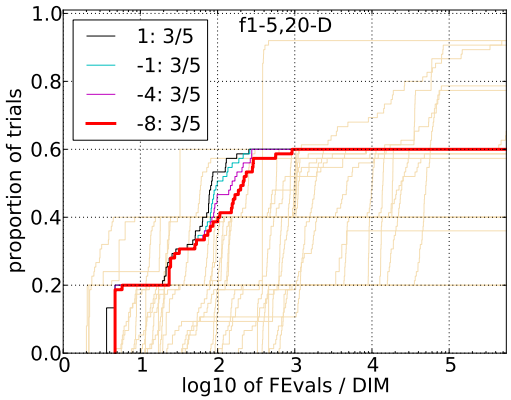
(d)



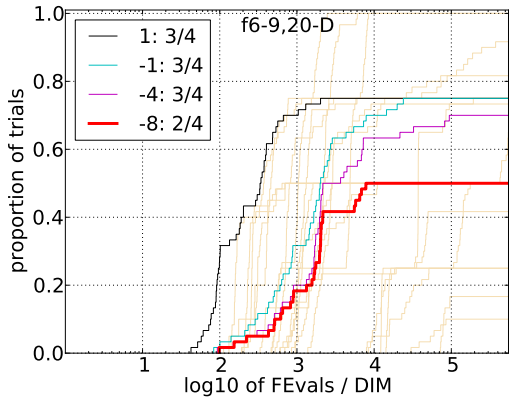
(e)

ECDF 20D function categorized

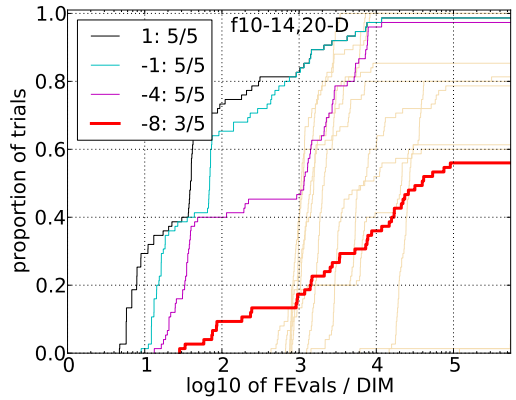
BBOB 2012 competition



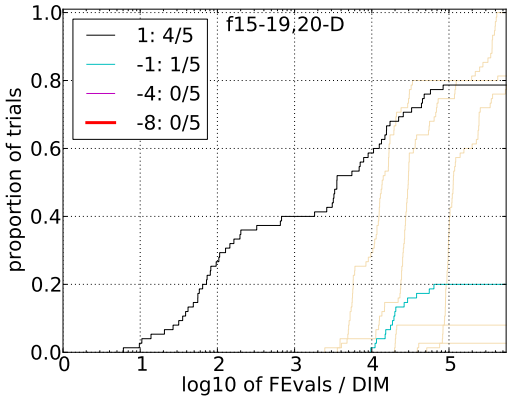
(a)



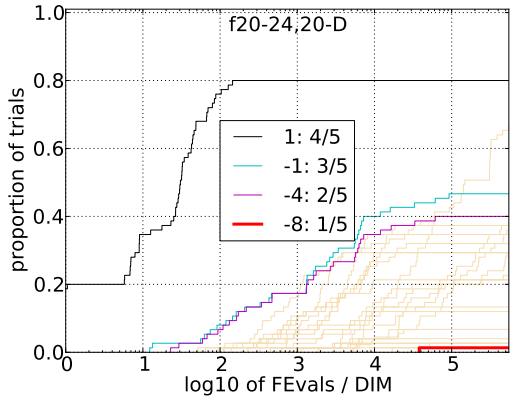
(b)



(c)



(d)



(e)

References


Ong, Y.-S., Lim, M.-H., Zhu, N., and Wong, K.-K. (2006).
Classification of adaptive memetic algorithms: A comparative study.
IEEE Transactions on systems, man, and cybernetics, 36(1):141–152.


Parsopoulos, K. E. and Vrahatis, M. N. (2010).
Particle Swarm Optimization and Intelligence: Advances and Applications.
Information Science Publishing (IGI Global).


Petalas, Y. G., Parsopoulos, K. E., and Vrahatis, M. N. (2007).
Memetic particle swarm optimization.
Annals of Operations Research, 156(1):99–127.


Voglis, C., Parsopoulos, K., Papageorgiou, D., Lagaris, I., and Vrahatis, M. (2012).
Mempsode: A global optimization software based on hybridization of population-based algorithms and local searches.
Computer Physics Communications, 183(5):1139–1154.