CEC'07

25th Sept 2007

Nanyang Technological University

Particle Swarm Optimization & Differential Evolution

Presenter: Assoc. Prof. P. N. Suganthan
School of Electrical and Electronic Engineering
Nanyang Technological University, Singapore.

Some Software Resources Available from:
http://www.ntu.edu.sg/home/epnsugan

Outline of the Presentation

I. Benchmark Test Functions

II. Real Parameter Particle swarm optimization (PSO)

- Basic PSO, its variants, Comprehensive learning PSO (CLPSO), Dynamic multi-swarm PSO (DMS-PSO)

III. Real Parameter Differential evolution (DE)

- DE, its variants, Self-adaptive differential evolution

IV. Constrained optimization

V. Multi-objective PSO / DE

VI. Multimodal optimization (niching)

VII. Binary / Discrete PSO & DE

VIII. Benchmarking results of CEC 2005, 2006, 2007.

- Dynamic, Robust optimization – excluded.

I - Benchmark Test Functions

Resources available from
http://www.ntu.edu.sg/home/epnsugan
(limited to our own work)

From Prof Xin Yao's group
http://www.cs.bham.ac.uk/research/projects/ecb/
Includes diverse problems.

Why do we require benchmark problems?

- Why we need test functions?
  - To evaluate a novel optimization algorithm's property on different types of landscapes
  - Compare different optimization algorithms
- Types of benchmarks
  - Bound constrained problems (real, binary, discrete, mixed)
  - Constrained problems
  - Single / Multi-objective problems
  - Static / Dynamic optimization problems
  - Multimodal problems
  - Various combinations of the above

Shortcomings in Bound constrained Benchmarks

- Some properties of benchmark functions may make them unrealistic or may be exploited by some algorithms:
  - Global optimum having the same parameter values for different variables / dimensions
  - Global optimum at the origin
  - Global optimum lying in the center of the search range
  - Global optimum on the bound
  - Local optima lying along the coordinate axes
  - no linkage among the variables / dimensions or the same linkages over the whole search range
  - Repetitive landscape structure over the entire space

Do real-world problems possess these properties?

Liang *et. al* 2006c (*Natural Computation*) has more details.

How to Solve?

- Shift the global optimum to a random position to make the global optimum to have different parameter values for different dimensions

- Rotate the functions as below:

$$F(\mathbf{x}) = f(\mathbf{R} * \mathbf{x})$$

where **R** is an orthogonal rotation matrix

- Use different classes of benchmark functions, different rotation matrices to compose a single test problem.

- These **Composition Functions** mix different properties of different basic test functions together to destroy repetitive structures.

Novel Composition Test Functions

- Compose the standard benchmark functions to construct a more challenging function with a randomly located global optimum and several randomly located deep local optima with different linkage properties over the search space.

- Gaussian functions are used to combine these benchmark functions and to blur individual functions' structures mainly around the transition regions.

- More details in Liang, *et al* 2005, CEC 2005 special sessions on benchmarking RP-EAs.

Novel Composition Test Functions

$F(x)$: new composition function.

$f_i(x)$: ith basic function used to construct the composition function.

n : number of basic functions. The bigger n is, the more complex $F(x)$ is.

D : dimension.

$[X\min, X\max]^D$: $F(x)$'s search range

$[x\min_i, x\max_i]^D$: $f_i(x)$'s search range

M_i : orthogonal rotation matrix for each $f_i(x)$

o_i : new shifted optimum position for each $f_i(x)$

o_{old} : old optimum position for each $f_i(x)$

$$F(x) = \sum_{i=1}^{n} \{w_i * [f_i((x - o_i + o_{old}) / \lambda_i * M_i) + bias_i]\} + f\_bias$$

Novel Composition Test Functions

w_i : weight value for each $f_i(x)$, calculated as below:

$$w_i = \exp\left(-\frac{\sum_{k=1}^{D}(x_k - o_{ik} + o_{ikold})^2}{2D\sigma_i^2}\right),$$

$$w_i = \begin{cases} w_i & if \quad w_i = \max(w_i) \\ w_i*(1-\max(w_i).^10) & if \quad w_i \neq \max(w_i) \end{cases}$$

then normalize the weight $w_i = w_i / \sum_{i=1}^{n} w_i$

σ_i : used to control each $f_i(x)$'s coverage range, a small σ_i gives a narrow range for $f_i(x)$.

λ_i : used to stretch or compress the function, $\lambda_i > 1$ means stretch, $\lambda_i < 1$ means compress.

usually set $\lambda_i = \sigma_i * \frac{X\max - X\min}{x\max_i - x\min_i}$

Novel Composition Test Functions

o_i define the global and local optima's position,

bias define which optimum is global optimum.

If $f_i(x)$ are different functions, different functions have different properties and height,

In order to get a better mixture, we estimate the biggest function value f_{max},

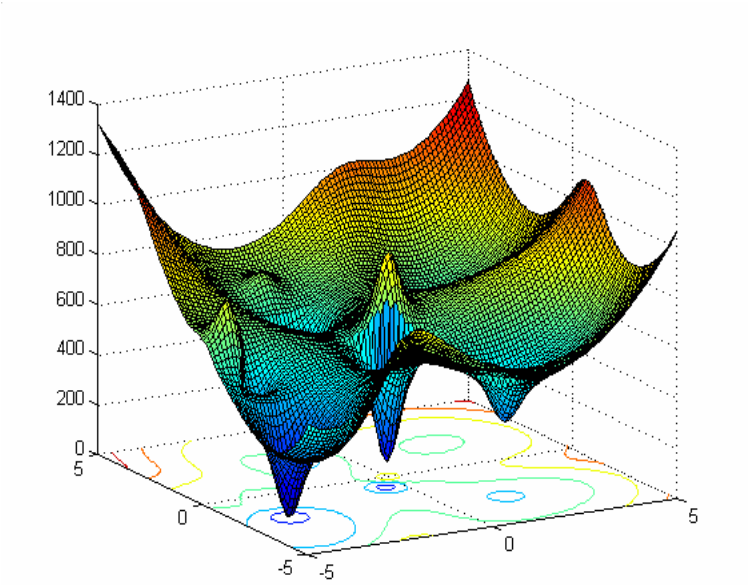
then normalize each basic function to similar height as below:

$f_i'(x) = C^* f_i(x) / |f_{max\,i}|$, C is a predefined constant.

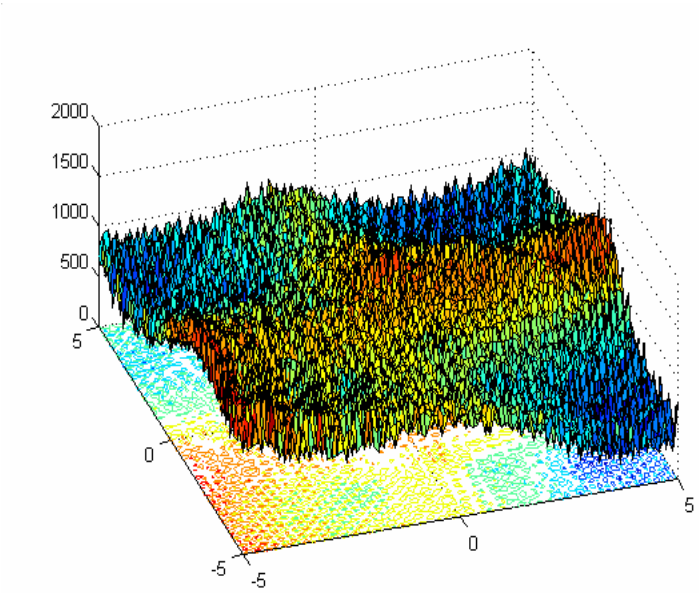
These composition functions can also be used as multimodal functions.

A couple of Examples

- Many composition functions are available from our homepage



Composition Function 1 (F1):
Made of Sphere Functions



Composition Function 2 (F2):
Made of Griewank's Functions

Similar analysis is needed for other benchmarks such as the multi-objective, constrained, etc.

NANYANG TECHNOLOGICAL UNIVERSITY

II - Particle Swarm Optimizer

- Introduced by Kennedy and Eberhart in 1995 (Eberhart & Kennedy,1995; Kennedy & Eberhart,1995)
- Emulates flocking behavior of birds, animals, insects, fish, etc. to solve optimization problems
- Each solution in the landscape is a particle
- All particles have fitness values and velocities
- The standard PSO does not have mutation, crossover, selection ,etc.

Particle Swarm Optimizer

- Two versions of PSO
  - Global version (**May not be used alone to solve multimodal problems**): Learning from the personal best (**pbest**) and the best position achieved by the whole population (**gbest**)

$$V_i^d \leftarrow c_1 * rand1_i^d * (pbest_i^d - X_i^d) + c_2 * rand2_i^d * (gbest^d - X_i^d)$$

$$X_i^d \leftarrow X_i^d + V_i^d$$

i – particle counter & *d* – dimension counter

  - Local Version: Learning from the **pbest** and the best position achieved in the particle's neighborhood population (**lbest**)

$$V_i^d \leftarrow c_1 * rand1_i^d * (pbest_i^d - X_i^d) + c_2 * rand2_i^d * (lbest_k^d - X_i^d)$$

$$X_i^d \leftarrow X_i^d + V_i^d$$

- **The random numbers (rand1 & rand2) should be generated for each dimension of each particle in every iteration.**

lbest to be defined w. r. t. a neighborhood.

General parameters in PSO

- c_1 and c_2 denote the acceleration constants usually set to ~2.
- $rand1_i^d$ and $rand2_i^d$ are two uniform random numbers within the range [0,1]
- $\mathbf{x}_i = (x_i^1, x_i^2, ..., x_i^D)$ represents the position of the *i*th particle
- $\mathbf{v}_i = (v_i^1, v_i^2, ..., v_i^D)$ represents the position changing rate (velocity) of the *i*th particle
- $\mathbf{pbest}_i = (pbest_i^1, pbest_i^2, ..., pbest_i^D)$ represents the best previous position (the position giving the best objective function value) of the *ith* particle
- $\mathbf{gbest} = (gbest^1, gbest^2, ..., gbest^D)$ represents the best previous position of the whole swarm
- $\mathbf{lbest}_i = (lbest_i^1, lbest_i^2, ..., lbest_i^D)$ represents the best previous position achieved by those particles within the neighborhood of the *ith* particle

PSO variants

- Modifying the Parameters
  - Inertia weight ω (Shi & Eberhart, 1998; Shi & Eberhart, 2001; Eberhart & Shi, 2001, …)
  - Constriction coefficient (Clerc,1999; Clerc & Kennedy, 2002)
  - Time varying acceleration coefficients (Ratnaweera *et al*. 2004)
  - Linearly decreasing V_{max} (Fan & Shi, 2001)
  - Tribes (Clerc 2006)
- Using Topologies
  - Extensive experimental studies (Kennedy, 1999; Kennedy & Mendes, 2002, …)
  - Dynamic neighborhood (Suganthan,1999; Hu and Eberhart, 2002; Peram *et al.* 2003)
  - Combine the global version and local version together (Parsopoulos and Vrahatis, 2004) named as the unified PSO or UPSO.
  - Fully informed PSO or FIPS (Mendes & Kennedy 2004) and so on …

PSO variants and Applications

- Hybrid PSO Algorithms
  - PSO + selection operator (Angeline,1998)
  - PSO + crossover operator (Lovbjerg, 2001)
  - PSO + mutation operator (Lovbjerg & Krink, 2002; Blackwell & Bentley,2002; … …)
  - PSO + dimension-wise search (Bergh & Engelbrecht, 2004)
  - …
- Various Optimization Scenarios & Applications
  - Binary Optimization (Kennedy & Eberhart, 1997; Agrafiotis *et. al* 2002;)
  - Constrained Optimization (Parsopoulos *et al.* 2002; Hu & Eberhart, 2002; …)
  - Multi-objective Optimization (Ray *et. al* 2002; Coello *et al*. 2002/04; …)
  - Dynamic Tracking (Eberhart & Shi 2001; …)
  - Yagi-Uda antenna (Baskar *et al* 2005b), Photonic FBG design (Baskar *et al* 2005a), FBG sensor network design (Liang *et al* June 2006)

PSO with Momentum / Constriction

- In PSO with momentum [SE98], a momentum term ω is introduced to the original equation:

$$v_i^d \leftarrow \omega v_i^d + c_1 rand1_i^d (pbest_i^d - x_i^d) + c_2 rand2_i^d (gbest^d - x_i^d)$$

$$x_i^d \leftarrow x_i^d + v_i^d$$

ω is usually reduced form 0.9 to 0.4

- PSO with constriction factor [CK02]:

$$v_i^d \leftarrow \chi [v_i^d + c_1 rand1_i^d (pbest_i^d - x_i^d) + c_2 rand2_i^d (gbest^d - x_i^d)]$$

$$x_i^d \leftarrow x_i^d + v_i^d$$

Constriction factor χ can be set to 0.7298.

PSO Variants by Kennedy *et. al*

- In fully informed particle swarm (FIPS) [KM06, MKN04], each particle's velocity is adjusted based on contributions from *pbest* of all its neighbors.
- Bare bones PSO [K03]: PSO without the velocity term, i.e. with the social & cognitive terms only.
- Essential Particle swarm [K06]: The velocity is expressed as direction defined by the particle's position at time *t* and time (*t*-1), i.e. the persistence and social influence.
- Essential Particle Swarm is another realization of the FIPS.

Comprehensive learning PSO (CLPSO)

- CLPSO learning strategy:

$$v_i^d \leftarrow w \times v_i^d + c \times rand_i^d \times \left(pbest_{f_i(d)}^d - x_i^d \right)$$

$$x_i^d \leftarrow x_i^d + v_i^d$$

- $f_i = [f_i(1), f_i(2), ..., f_i(D)]$ denotes a set of particle indices with respect to each dimension of the particle *i*. $f_i(d)$ represents a comprehensive exemplar with each dimension composed of the value from the corresponding dimension of the **pbest** of particle $\mathbf{pbest}_{f_i}$. These indices take the value *i* itself with the probability Pc_i, referred to as the learning probability, which takes different values with respect to different particles.

- For each dimension of particle *i*, we generate a random number. If this random number is larger than Pc_i, the corresponding dimension of particle *i* will learn from its own **pbest**, otherwise it will learn from the **pbest** of another randomly chosen particle.

CLPSO

- Tournament selection with size 2 is used to choose the index $f_i(d)$.
- To ensure that a particle learns from good exemplars and to minimize the time wasted on poor directions, we allow each particle to learn from the exemplars until such particle stop to improve for a certain number of generations, called the refreshing gap *m* (7 generations).
- After that, we re - assign $f_i = [f_i(1), f_i(2),..., f_i(D)]$ for each particle *i*.
- The detailed description and algorithmic implementation can be found in [LQSB06]. Matlab codes including CLPSO and several state-of-the-art PSO variants are available for academic use.

CLPSO

- Three major differences between CLPSO and the conventional PSO are highlighted:
  - Instead of using particle's **pbest** and **gbest** as the exemplars, all particles' **pbest**s can be used to guide a particle's flying direction.
  - Instead of learning from the same exemplar for all dimensions, different dimensions of a particle may learn from different exemplars within certain generations. In other words, at one iteration, each dimension of a particle may learn from the corresponding dimension of different particle's **pbest**.
  - Instead of learning from two exemplars (**pbest** and **gbest**) in every generation, each dimension of a particle in CLPSO learns from just one comprehensive exemplar within certain generations.
- Experimental results [LQSB06] over a suite of 16 numerical test functions have demonstrated the promising performance of the CLPSO to solve the multi-modal optimization problems in comparison with 8 state-of-the-art PSO variants.

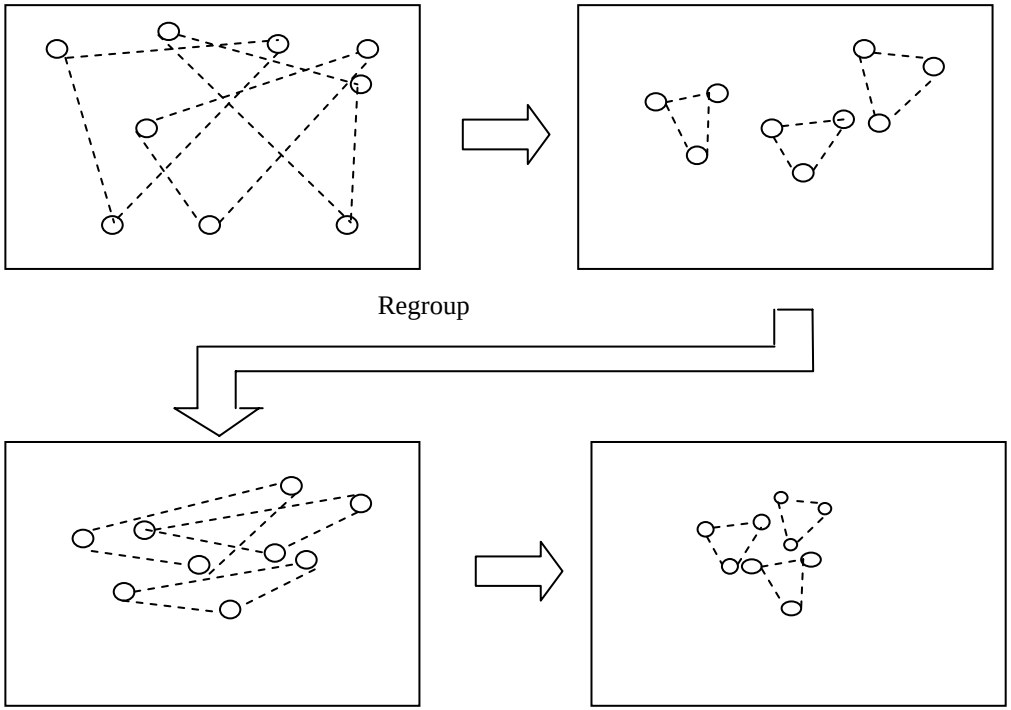
CLPSO with Probability Adaptation

Adaptive Self-Learning Strategy

- Assume **Pc** normally distributed in a range with mean(**Pc**) and a standard deviation of 0.1.
- Initially, mean(**Pc**) is set at 0.5 and different **Pc** values conforming to this normal distribution are generated for each individual in the current population.
- During every generation, the **Pc** values associated with the particles which find new **pbest** are recorded.
- The mean of normal distribution of **Pc** is recalculated according to all the recorded **Pc** values corresponding to successful movements during the last several generations.
- As a result, the proper **Pc** value range for the current problem can be learned to suit the particular problem.

Dynamic multi-swarm PSO (DMS-PSO)

- DMS-PSO is constructed based on the local version of PSO with a novel neighborhood topology
- Two major characteristics of the novel neighborhood topology:
  - Small sized swarms
  - Randomized re-grouping scheme



- The population is divided into several sub-swarms randomly.
- Each sub-swarm utilizes its own particles to search for better solutions and converge to some suboptimal solution.
- The whole population is re-grouped into new sub-swarms periodically. New sub-swarms continue the search procedure.
- This process continues until a termination criterion is satisfied.

DMS-PSO

- DMS-PSO learning strategy

  Each particle *i* has an associated vector $\mathbf{Pc}_i$. After every *R* generations, an indicator vector $\mathbf{keepid}_i$ will be updated according to $\mathbf{Pc}_i$: if $rand_i^d$ is larger than or equal to $\mathbf{Pc}_i(d)$, $\mathbf{keepid}_i(d)$ is set to 1 and the d^{th} dimension of particle *i* will be set as the value of its own $pbest_i(d)$, otherwise $\mathbf{keepid}_i(d)$ is set to 0, and the d^{th} dimension of particle *i* will learn from its $\mathbf{lbest}_i(d)$, and its own $\mathbf{pbest}_i(d)$, as the PSO with constriction coefficients:

If $keep\_id_i^d = 0$

$$v_i^d \leftarrow 0.729 \times v_i^d + 1.49445 \times rand1_i^d \times (pbest_i^d - x_i^d) + 1.49445 \times rand2_i^d \times (lbest_i^d - x_i^d)$$

$$v_i^d = \min(v_{\max}^d, \max(-v_{\max}^d, v_i^d))$$

$$x_i^d \leftarrow x_i^d + v_i^d$$

Otherwise

$$x_i^d \leftarrow pbest_i^d$$

Somewhat similar to DE & CPSO

DMS-PSO

- Parameter adaptation scheme
  - Assume $\mathbf{Pc}_i$ is normally distributed with mean *mean_Pc* and standard deviation 0.1.
  - Initially, *mean_Pc* is set to 0.5 and a set of $\mathbf{Pc}_i$ vectors with respect to each particle *i* in the current population are generated according to such normal distribution.
  - At each generation, the $\mathbf{Pc}_i$ values associated with those particles that find new **pbests** are recorded.
  - When sub-swarms are regrouped, *mean_Pc* is re-calculated according to all the recorded successful $\mathbf{Pc}_i$ values. The recorded successful $\mathbf{Pc}_i$ values will be cleared when *mean_Pc* is recalculated.
  - As a result, a proper $\mathbf{Pc}_i$ distribution with respect to the given problem can be evolved.

DMS-PSO with local search

- Although we can achieve larger diversity using DMS-PSO, the convergence rate may slow down. In order to alleviate this problem, a local search procedure is incorporated:
  - Every *L* generations, **pbest**s of five randomly chosen particles will be used as the starting points and the Quasi-Newton method is applied to conduct the local search with maximum function evaluations *L_FEs.*
  - At the end of the DMS-PSO search, particles in each sub-swarm are grouped into a whole swarm to perform the global PSO. The best solution achieved so far is refined using the Quasi-Newton method every *L* generations with the $5 \times L\_FEs$ as the maximum search step.
  - If local search results in improvements, the nearest **pbest** is replaced.

III - Outline of Presentation on DE

- Motivation for Differential Evolution (DE)
- Classical DE
- DE Variants
- Self-adaptive DE (SaDE)

Motivation for DE

- DE, proposed by Price and Storn in 1995 [PS95], was motivated by the attempts to use Genetic Annealing [P94] to solve the Chebychev polynomial fitting problem.

- Genetic annealing is a population-based, combinatorial optimization algorithm that implements a thermodynamic annealing criterion via thresholds. Although successfully applied to solve many combinatorial tasks, genetic annealing could not solve the Chebychev problem satisfactorily.

- Price modified genetic annealing by using floating-point encoding instead of bit-string one, arithmetic operations instead of logical ones, population-driven differential mutation instead of bit-inversion mutation and removed the annealing criterion. Storn suggested creating separate parent and children populations. Eventually, Chebychev problem can be solved effectively.

- DE is closely related to many other multi-point derivative free search methods [PSL05] such as evolutionary strategies, genetic algorithms, Nelder and Mead direct search and controlled random search.

DE at a glance

- **Characteristics**
  - Population-based stochastic direct search
  - Self-referential mutation
  - Simple but powerful
  - Reliable, robust and efficient
  - Easy parallelization
  - Floating-point encoding
- **Basic components**
  - Initialization
  - Trial vector generation
    - Mutation
    - Recombination
  - Replacement

Insight into classical DE (DE/rand/1/bin)

Initialization

A population $P_{\mathbf{x},0}$ of Np D-dimensional parameter vectors $\mathbf{x}_{i,0}=[x_{1,i,0},\ldots,x_{D,i,0}]$, $i=1,\ldots,Np$ is randomly generated within the prescribed lower and upper bound $\mathbf{b}_L=[b_{1,L},\ldots,b_{D,L}]$ and $\mathbf{b}_U=[b_{1,U},\ldots,b_{D,U}]$

Example: the initial value (at generation $g=0$) of the j^{th} parameter of the i^{th} vector is generated by: $x_{j,i,0} = \text{rand}_j[0,1] \cdot(b_{j,U}-b_{j,L}) + b_{j,L}$, $j=1,\ldots,D$, $i=1,\ldots,Np$

Trial vector generation

At the g^{th} generation, a trial population $P_{\mathbf{u},g}$ consisting of Np D-dimensional trial vectors $\mathbf{v}_{i,g}=[v_{1,i,g},\ldots v_{D,i,g}]$ is generated via mutation and recombination operations applied to the current population $P_{\mathbf{x},g}$

Differential mutation: with respect to each vector $\mathbf{x}_{i,g}$ in the current population, called target vector, a mutant vector $\mathbf{v}_{i,g}$ is generated by adding a scaled, randomly sampled, vector difference to a basis vector randomly selected from the current population

Insight into classical DE (DE/rand/1/bin)

Example: at the g^{th} generation, the i^{th} mutant vector $\mathbf{v}_{i,g}$ with respect to i^{th} target vector $\mathbf{x}_{i,g}$ in the current population is generated by $\mathbf{v}_{i,g} = \mathbf{x}_{r0,g} + F\cdot(\mathbf{x}_{r1,g}-\mathbf{x}_{r2,g})$, $i \neq r0 \neq r1 \neq r2$, mutation scale factor $F \in (0,1+)$

Discrete recombination: with respect to each target vector $\mathbf{x}_{i,g}$ in the current population, a trial vector $\mathbf{u}_{i,g}$ is generated by crossing the target vector $\mathbf{x}_{i,g}$ with the corresponding mutant vector $\mathbf{v}_{i,g}$ under a pre-specified crossover rate $Cr \in [0,1]$

Example: at the g^{th} generation, the i^{th} trial vector $\mathbf{u}_{i,g}$ with respect to i^{th} target vector $\mathbf{x}_{i,g}$ in the current population is generated by:

$$u_{j,i,g} = \begin{cases} v_{j,i,g} & \text{if } \text{rand}_j[0,1] \leqslant Cr \text{ or } j=j_{\text{rand}} \\ x_{j,i,g} & \text{otherwise} \end{cases}$$

Replacement

If the trial vector $\mathbf{u}_{i,g}$ has equal or better objective function value than that of its corresponding target vector $\mathbf{x}_{i,g}$, it replaces the target vector in the $(g+1)^{th}$ generation; otherwise the target vector remains in the $(g+1)^{th}$ generation

Illustration of classical DE

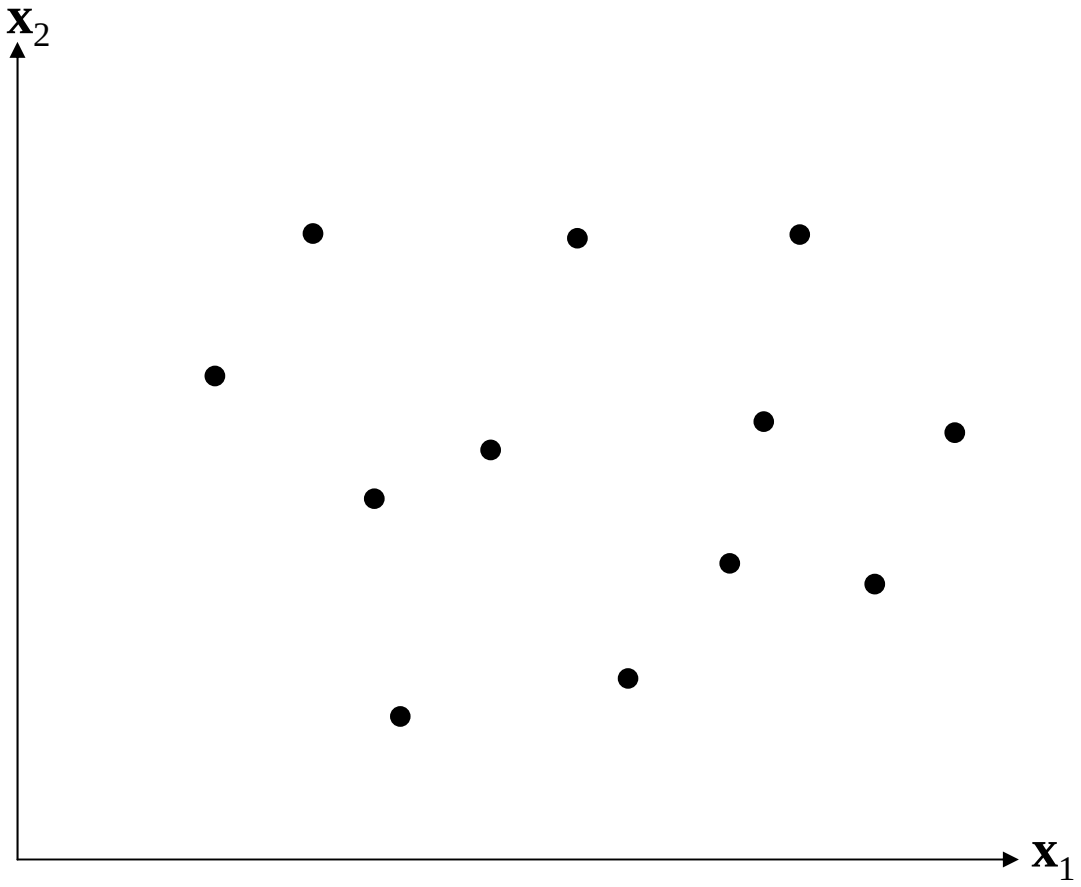


Illustration of classic DE

Illustration of classical DE

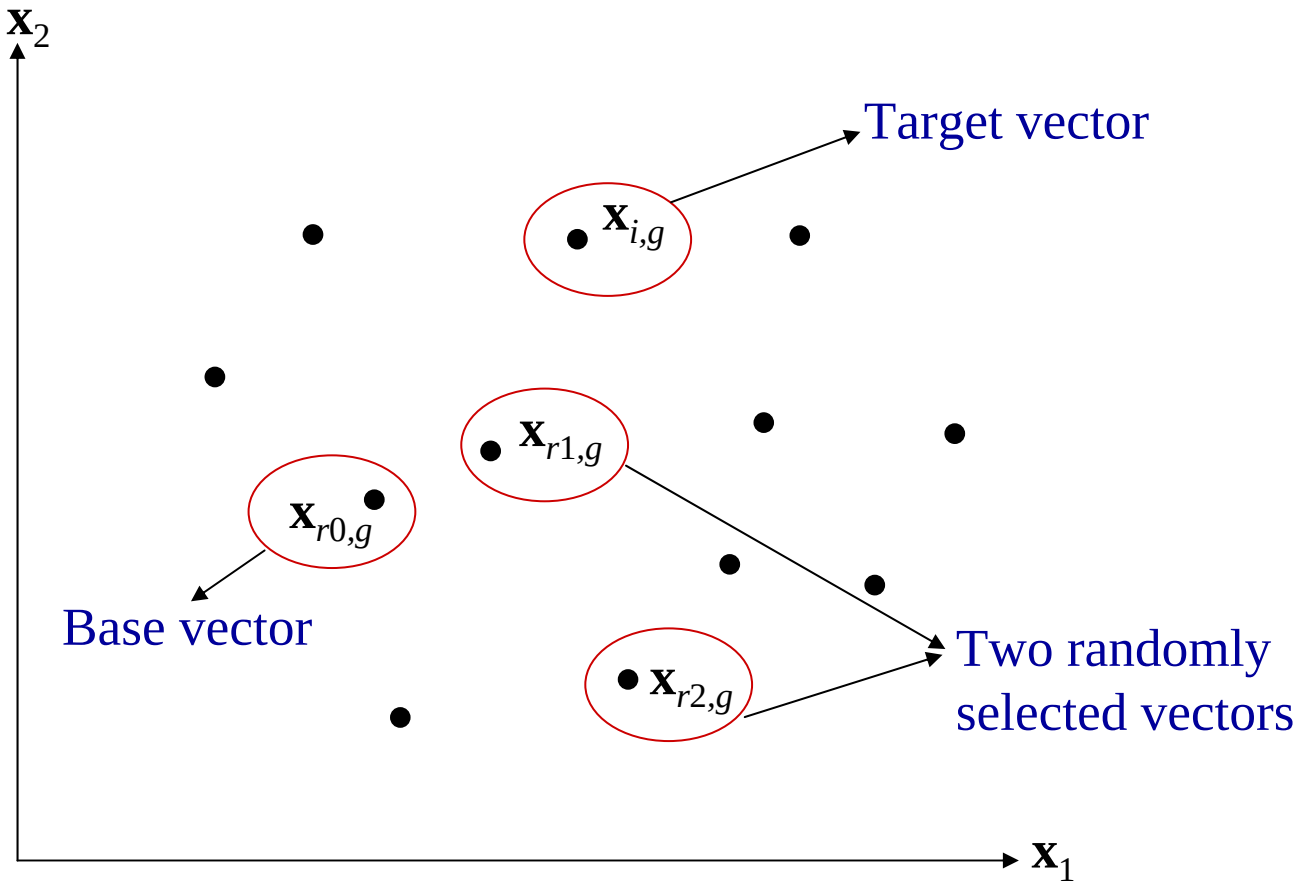
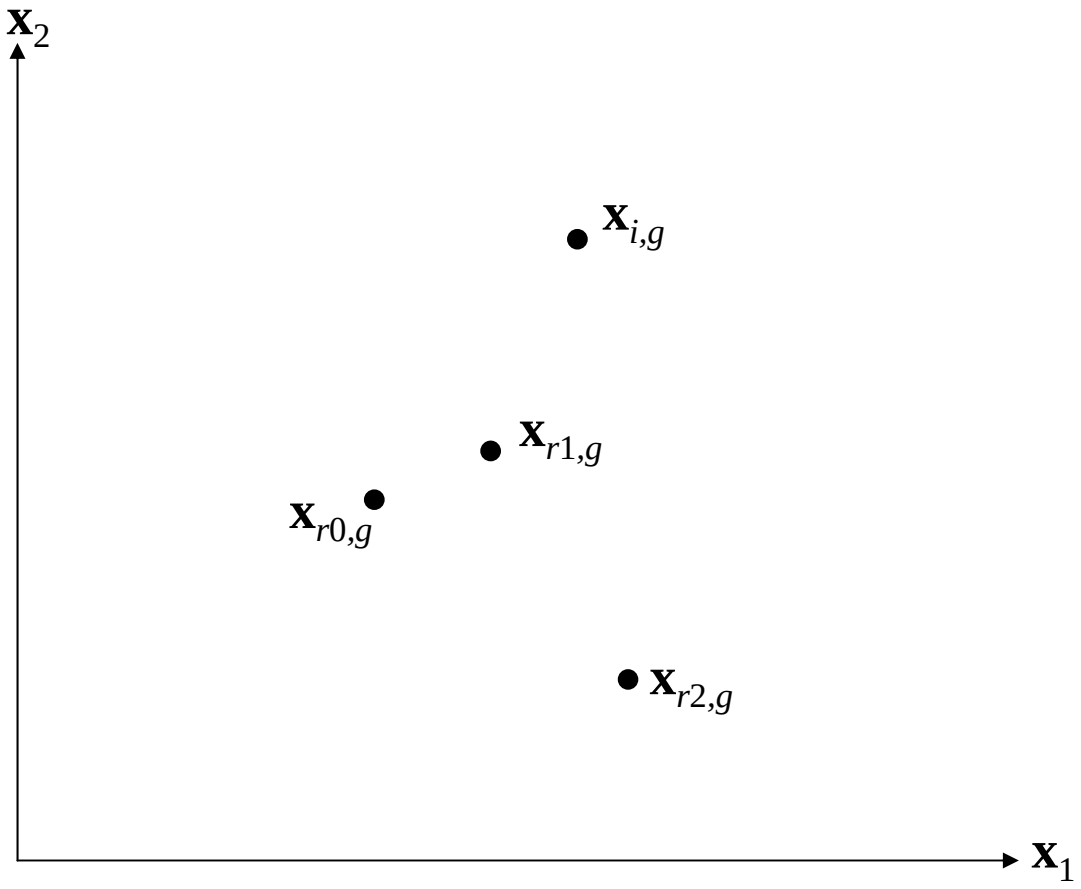


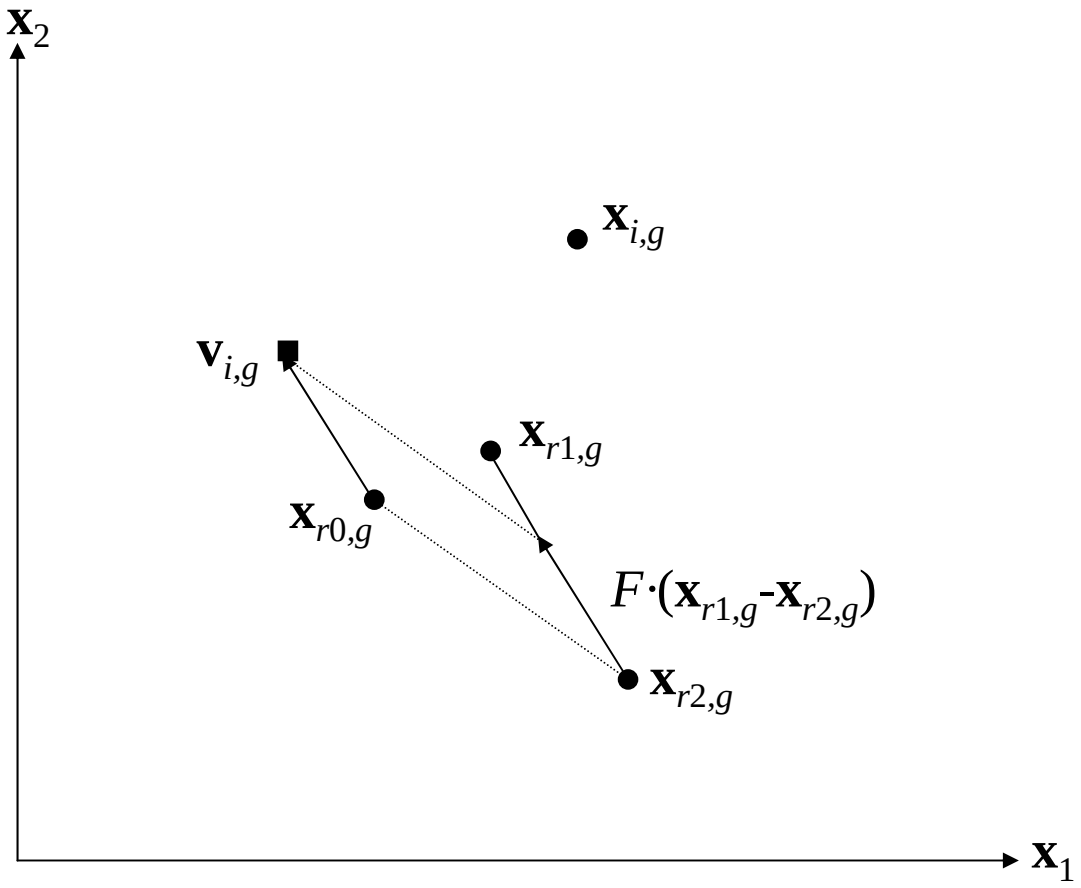
Illustration of classic DE

Illustration of classical DE



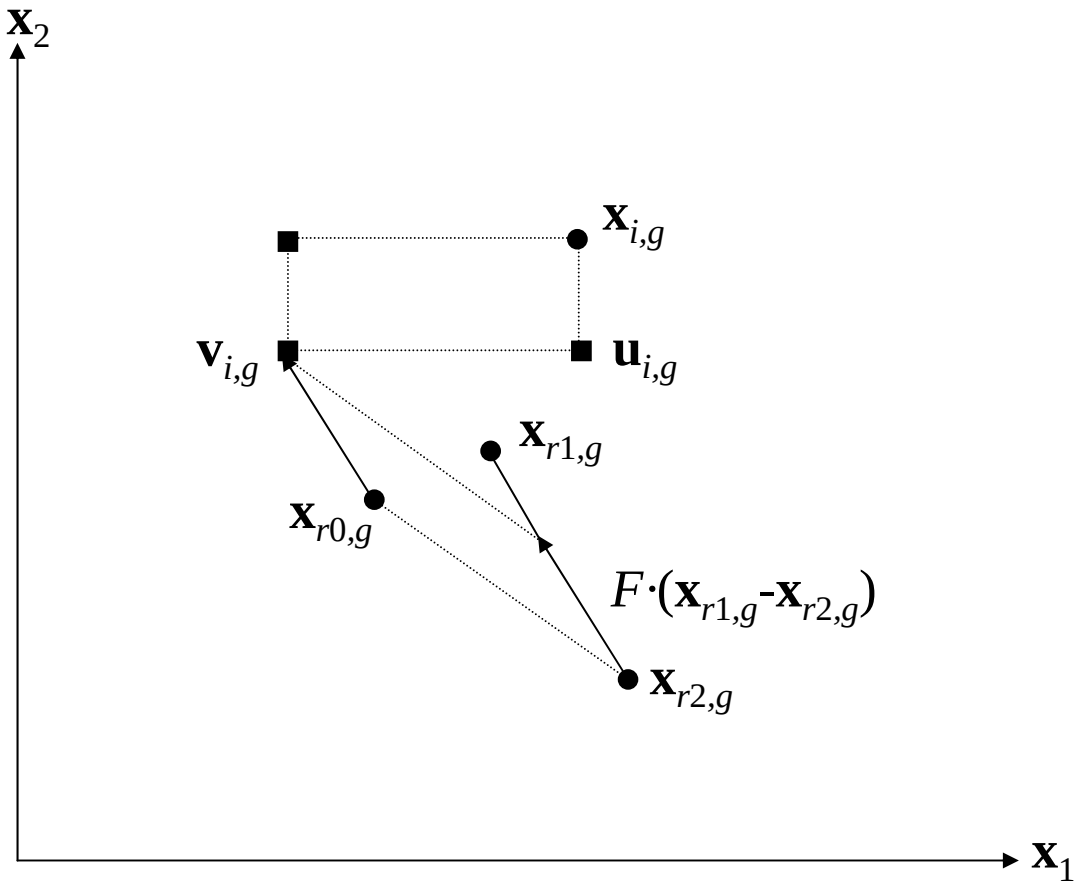
Four operating vectors in 2D continuous space

Illustration of classical DE



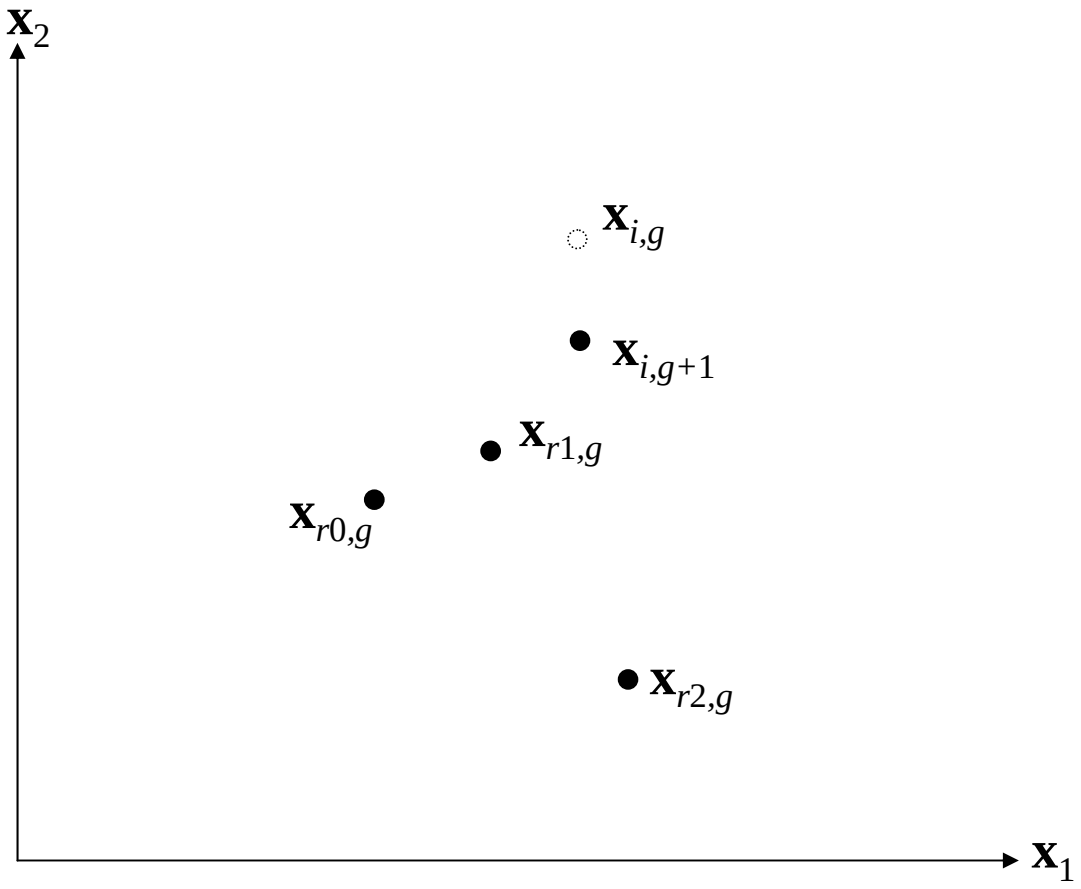
Trial vector after Mutation

Illustration of classical DE



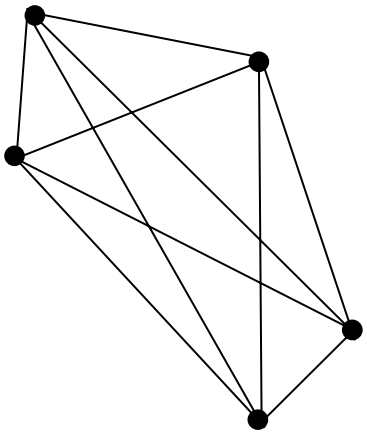
Trial vector after Crossover

Illustration of classical DE

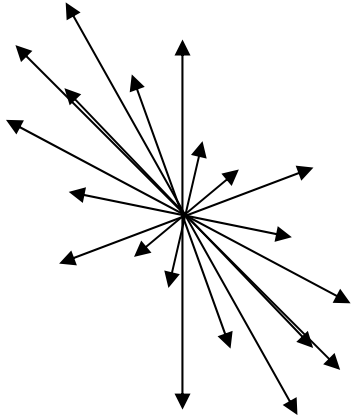


Replacement of target vector by the trial vector

Differential vector distribution



A population of 5 vectors



20 generated difference vectors

Most important characteristics of DE: self-referential mutation!

ES: fixed probability distribution function with adaptive step-size

DE: adaptive distribution of difference vectors with fixed step-size

DE variants

Modification of different components of DE can result in many DE variants:

Initialization

Uniform distribution and Gaussian distribution

Trial vector generation

- Choices in base vector selection
  - Random selection without replacement: r_0=ceil(rand$_i$[0,1]·Np)
  - Permutation selection: r_0=permute[i]
  - Random offset selection: r_0=(i+r_g)%Np (e.g. r_g=2)
  - Biased selection: global best, local best or tournament

DE variants

- Differential mutation
  - One difference vector: $F \cdot (\mathbf{x}_{r1} - \mathbf{x}_{r2})$
  - Two difference vector: $F \cdot (\mathbf{x}_{r1} - \mathbf{x}_{r2}) + F \cdot (\mathbf{x}_{r3} - \mathbf{x}_{r4})$
  - Mutation scale factor F
    - Crucial role: balance exploration and exploitation
    - Dimension dependence?: *jitter, if yes* (rotation variant) and *dither, if no* (rotation invariant).
    - Randomization: different distributions of F

DE/rand/1: $$\mathbf{V}_{i,G} = \mathbf{X}_{r_1,G} + F \cdot \left(\mathbf{X}_{r_2,G} - \mathbf{X}_{r_3,G}\right)$$

DE/best/1: $$\mathbf{V}_{i,G} = \mathbf{X}_{best,G} + F \cdot \left(\mathbf{X}_{r_1,G} - \mathbf{X}_{r_2,G}\right)$$

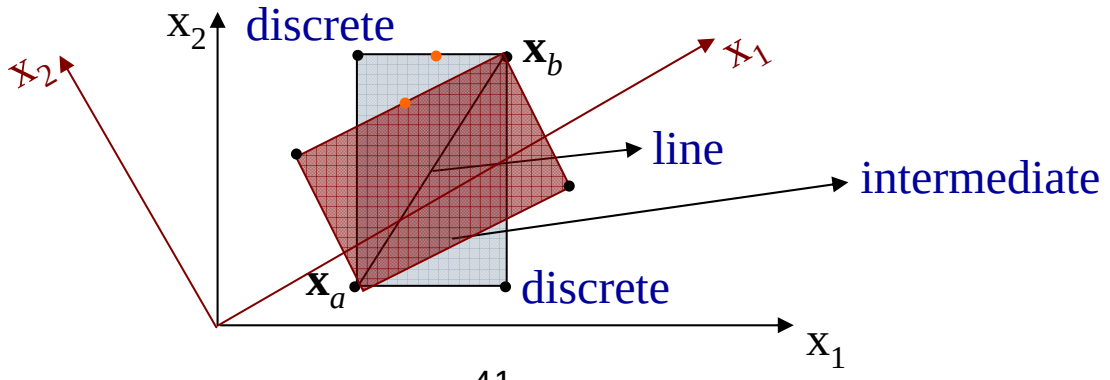
DE/current-to-best/1: $$\mathbf{V}_{i,G} = \mathbf{X}_{i,G} + F \cdot \left(\mathbf{X}_{best,G} - \mathbf{X}_{i,G}\right) + F \cdot \left(\mathbf{X}_{r_1,G} - \mathbf{X}_{r_2,G}\right)$$

DE/rand/2: $$\mathbf{V}_{i,G} = \mathbf{X}_{r_1,G} + F \cdot \left(\mathbf{X}_{r_2,G} - \mathbf{X}_{r_3,G} + \mathbf{X}_{r_4,G} - \mathbf{X}_{r_5,G}\right)$$

DE/best/2: $$\mathbf{V}_{i,G} = \mathbf{X}_{best,G} + F \cdot \left(\mathbf{X}_{r_1,G} - \mathbf{X}_{r_2,G} + \mathbf{X}_{r_3,G} - \mathbf{X}_{r_4,G}\right)$$

DE variants

- Recombination
  - Discrete recombination (crossover) (rotation variant)
    - One point and multi-point
    - Exponential (somewhat comparable to two-point)
    - Binominal (uniform)
  - Arithmetic recombination
    - Line recombination (rotation invariant, vector operation)
    - Intermediate recombination (rotation variant, dimension-wise)
    - Extended intermediate recombination (rotation variant)



Motivation for self-adaptation in DE

The performance of DE on different problems depends on:

- Population size
- Strategy and the associated parameter setting to generate trial vectors
- Replacement scheme

It is hard to choose a unique combination to successfully solve any problem at hand

- Population size usually depends on the problem scale and complexity
- During evolution, different strategies coupled with specific parameter settings may be effective for different search stages.
- Replacement schemes influence the population diversity
- Trial and error scheme may be a waste of computational time & resources

Automatically adapt the configuration in DE so as to generate effective trial vectors during evolution

Related works

Practical guideline [SP95], [SP97], [CDG99], [BO04], [PSL05],[GMK02]: for example, $Np \in [5D, 10D]$; Initial choice of F=0.5 and CR=0.1/0.9; Increase NP and/or F if premature convergence happens. Conflicting conclusions with respect to different test functions.

Fuzzy adaptive DE [LL02]: use fuzzy logical controllers whose inputs incorporate the relative function values and individuals of successive generations to adapt the mutation and crossover parameters.

Self-adaptive Pareto DE [A02]: encode crossover rate in each individual, which is simultaneously evolved with other parameters. Mutation scale factor is generated for each variable according to Gaussian distribution $N(0,1)$.

Zaharie [Z02]: theoretically study the DE behavior so as to adapt the control parameters of DE according to the evolution of population diversity.

Self-adaptive DE (1) [OSE05]: encode mutation scale factor in each individual, which is simultaneously evolved with other parameters. Crossover rate is generated for each variable according to Gaussian distribution $N(0.5,0.15)$.

DE with self-adaptive population [T06]: population size, mutation scale factor and crossover rate are all encoded into each individual.

Self-Adapting Control Parameters in DE

- **[BGBMZ06]** jDE algorithm encodes mutation scale factor *F* and crossover rate *CR* in each individual.
- New values for *F* & *CR* are assigned to each individual from a set of values and the assignment is performed randomly with respect to pre-specified 2 parameter values.
- jDE2 algorithm [BBG06] introduces re-initialization of poorly performing individuals to the jDE algorithm.

Self-adaptive DE (SaDE)

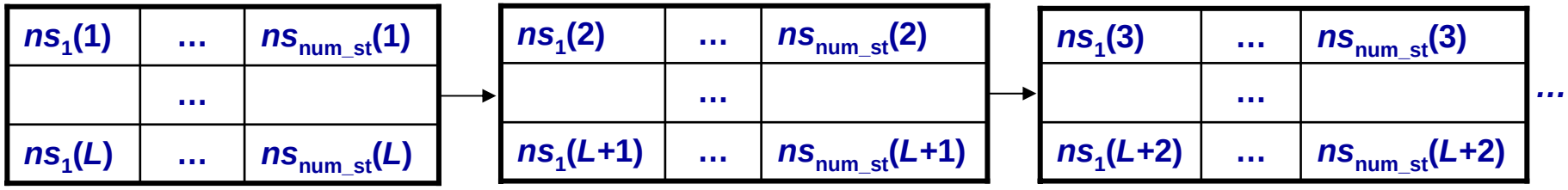
DE with strategy and parameter self-adaptation [QS05, HQS06]

Strategy adaptation: select one strategy from a pool of candidate strategies with the probability proportional to its previous successful rate to generate effective trial vectors during a certain learning period

Steps:

1. Initialize selection probability $p_i = 1/num\_st$, $i = 1,\ldots,num\_st$ for each strategy
2. According to the current probabilities, we employ ***stochastic universal selection*** to assign one strategy to each target vector in the current population
3. For each strategy, define vectors ns_i *and* nf_i, $i = 1,\ldots num\_st$ to store the number of trial vectors successfully entering the next generation or discarded by applying such strategy, respectively, within a specified number of generations, called "learning period (LP)"
4. Once the current number of generations is over LP, the first element of ns_i *and* nf_i with respect to the earliest generation will be removed and the behavior in current generation will update ns_i *and* nf_i

Self-adaptive DE (SaDE)



5. The selection probability p_i is updated by $\sum ns_i / \left(\sum ns_i + \sum nf_i\right)$. Go to 2nd step

Parameter adaptation

Mutation scale factor (F): *for each target vector in the current population*, we randomly generate F value according to a normal distribution $N(0.5,0.3)$. Therefore, 99% F values fall within the range of [–0.4,1.4]

Crossover rate (CR_j): when applying strategy j with respect to a target vector, the corresponding CR_j value is generated according to an assumed distribution, and those CR_j values that have generated trial vectors successfully entering the next generation are recorded and updated every LP generations so as to update the parameters of the CR_j distribution. We hereby assume that each CR_j, j=1,…,*num_st* is normally distributed with its mean and standard deviation initialized to 0.5 and 0.1, respectively

Instantiations

- In CEC'05, we use 2 strategies:

DE/rand/1/bin: $\mathbf{V}_{i,G} = \mathbf{X}_{r_1,G} + F \cdot \left(\mathbf{X}_{r_2,G} - \mathbf{X}_{r_3,G}\right)$

DE/current-to-best/2/bin: $\mathbf{V}_{i,G} = \mathbf{X}_{i,G} + F \cdot \left(\mathbf{X}_{best,G} - \mathbf{X}_{i,G}\right) + F \cdot \left(\mathbf{X}_{r_1,G} - \mathbf{X}_{r_2,G} + \mathbf{X}_{r_3,G} - \mathbf{X}_{r_4,G}\right)$

LP = 50

- In CEC'06, we employ 4 strategies:

DE/rand/1/bin: $\mathbf{V}_{i,G} = \mathbf{X}_{r_1,G} + F \cdot \left(\mathbf{X}_{r_2,G} - \mathbf{X}_{r_3,G}\right)$

DE/rand/2/bin: $\mathbf{V}_{i,G} = \mathbf{X}_{r_1,G} + F \cdot \left(\mathbf{X}_{r_2,G} - \mathbf{X}_{r_3,G} + \mathbf{X}_{r_4,G} - \mathbf{X}_{r_5,G}\right)$

DE/current-to-best/2/bin: $\mathbf{V}_{i,G} = \mathbf{X}_{i,G} + F \cdot \left(\mathbf{X}_{best,G} - \mathbf{X}_{i,G}\right) + F \cdot \left(\mathbf{X}_{r_1,G} - \mathbf{X}_{r_2,G} + \mathbf{X}_{r_3,G} - \mathbf{X}_{r_4,G}\right)$

DE/current-to-rand/1: $\mathbf{V}_{i,G} = \mathbf{X}_{i,G} + F \cdot \left(\mathbf{X}_{r_1,G} - \mathbf{X}_{i,G}\right) + F \cdot \left(\mathbf{X}_{r_2,G} - \mathbf{X}_{r_3,G}\right)$

LP = 50

Local search enhancement

To improve convergence speed, we apply a local search procedure every 500 generations:

- To apply local search, we choose $n = 0.05 \cdot Np$ individuals, which include the individual having the best objective function value and the n-1 individuals randomly selected from the top 50% individuals in the current population

- We perform the local search by applying the Quasi-Newton method to the selected n individuals

Overview of DE research trends

DE Applications

- **Digital Filter Design**
- **Multiprocessor synthesis**
- **Neural network learning**
- **Diffraction grating design**
- **Crystallographic characterization**
- **Beam weight optimization in radiotherapy**
- **Heat transfer parameter estimation in a trickle bed reactor**
- **Electricity market simulation**
- **Scenario-Integrated Optimization of Dynamic Systems**
- **Optimal Design of Shell-and-Tube Heat Exchangers**
- **Optimization of an Alkylation's Reaction**
- **Optimization of Thermal Cracker Operation**
- **Optimization of Non-Linear Chemical Processes**
- **Optimum planning of cropping patterns**
- **Optimization of Water Pumping System**
- **Optimal Design of Gas Transmission Network**
- **Differential Evolution for Multi-Objective Optimization**
- **Bioinformatics**

IV - Constrained Optimization

- Optimization of constrained problems is an important area in the optimization field.
- In general, the constrained problems can be transformed into the following form:

- Minimize $f(\mathbf{x}), \mathbf{x} = [x_1, x_2, ..., x_D]$

  subjected to:

$$g_i(\mathbf{x}) \le 0, i = 1, ..., q$$

$$h_j(\mathbf{x}) = 0,\ j = q+1, ..., m$$

q is the number of inequality constraints and *m*-*q* is the number of equality constraints.

Constrained Optimization

- For convenience, the equality constraints can be transformed into inequality form:

$$|h_j(\mathbf{x})| - \varepsilon \le 0$$

where ε is the allowed tolerance.

- Then, the constrained problems can be expressed as

Minimize $f(\mathbf{x}), \mathbf{x} = [x_1, x_2, ..., x_D]$

subjected to $G_j(\mathbf{x}) \le 0,\ j = 1, ..., m,$

$$G_{1,...,q}(\mathbf{x}) = g_{1,...q}(\mathbf{x}), G_{q+1,...,m}(\mathbf{x}) = \left|h_{q+1,...m}(\mathbf{x})\right| - \varepsilon$$

- If we denote with F the feasible region and S the whole search space, $\mathbf{x} \in F$ if $\mathbf{x} \in S$ and all constraints are satisfied. In this case, $\mathbf{x}$ is a feasible solution.

Constraint-Handling (CH) Techniques

- **Penalty Functions:**
  - Static Penalties (Homaifar et al.,1994;…)
  - Dynamic Penalty (Joines & Houck,1994; Michalewicz& Attia,1994;…)
  - Adaptive Penalty (Eiben *et al.* 1998; Coello, 1999; Tessema & Gary Yen 2006, Smith & Tate 1993…)
  - …

- **Superiority of feasible solutions**
  - Start with a population of feasible individuals (Michalewicz, 1992; Hu & Eberhart, 2002; …)
  - Feasible favored comparing criterion (Ray, 2002; Takahama & Sakai, 2005; …)
  - Specially designed operators (Michalewicz, 1992; …)
  - …

Constraint-Handling (CH) Techniques

- **Separation of objective and constraints**
  - Stochastic Ranking (Runarsson & Yao, TEC, Sept 2000)
  - Co-evolution methods (Coello, 2000a)
  - Multi-objective optimization techniques (Coello, 2000b; Mezura-Montes & Coello, 2002;…)
  - Feasible solution search followed by optimization of objective (Venkatraman & Gary Yen, 2005)
  - …

- While most CH techniques are modular (i.e. we can pick one CH technique and one search method independently), there are also CH techniques embedded as an integral part of the EA.

DMS-PSO for Constrained Optimization

Novel Constraint-Handling Mechanism

- Suppose that there are *m* constraints, the population is divided into *n* sub-swarms with *sn* members in each sub-swarm and the population size is *ps* (*ps=n*sn)*. *n* is a positive integer and '*n=m'* is not required.

- The objective and constraints are assigned to the sub-swarms adaptively according to the difficulties of the constraints.

- By this way, it is expected to have population of feasible individuals with high fitness values.

DMS-PSO's Constraint-Handling Mechanism

How to assign the objective and constraints to each sub-swarm?

Define

$$a > b = \begin{cases} 1 & if \quad a > b \\ 0 & if \quad a \le b \end{cases}$$

$$p_i = \frac{\sum_{j=1}^{ps} (g_i(\mathbf{x}_j) > 0)}{ps}, \quad i = 1, 2, ..., m$$

Thus $fp = 1 - \bar{\mathrm{p}} \quad \mathbf{p} = [p_1, p_2, ..., p_m]$

$$fp + \sum_{i=1}^{m} (p_i / m) = 1$$

DMS-PSO's Constraint-Handling Mechanism

For each sub-swarm,

- Using roulette selection according to *fp* and p_i / m to assign the objective function or a single constraint as its target.

  If sub-swarm *i* is assigned to improve constraint *j*, set *obj(i)=j* and if sub-swarm *i* is assigned to improve the objective function, set *obj(i)*=0.

- Assigning swarm member for this sub-swarm: Sort the unassigned particles according to *obj(i),* and assign the best and *sn*-1 worst particles to sub-swarm *i*.

DMS-PSO's Comparison Criteria

- 1. If *obj(i) = obj(j) = k (*particle *i* and *j* handling the same constraint *k)*, particle *i* wins if

$$G_k(\mathbf{x}_i) < G_k(\mathbf{x}_j) \text{ with } G_k(\mathbf{x}_j) > 0$$

$$or \quad V(\mathbf{x}_i) < V(\mathbf{x}_j) \,\&\, G_k(\mathbf{x}_i), G_k(\mathbf{x}_j) \le 0$$

$$or \quad f(\mathbf{x}_i) < f(\mathbf{x}_j) \,\&\, V(\mathbf{x}_i) == V(\mathbf{x}_j)$$

- 2. If *obj(i) = obj(j) = 0* (particle *i* and *j* handling *f(x)*) or *obj(i)* $\neq$ *obj(j) (i* and *j* handling different objectives*)*, particle *i* wins if

$$V(\mathbf{x}_i) < V(\mathbf{x}_j)$$

$$or \quad f(\mathbf{x}_i) < f(\mathbf{x}_j) \,\&\, V(\mathbf{x}_i) == V(\mathbf{x}_j)$$

$$V(\mathbf{x}) = \sum_{i=1}^{m} (weight_i \cdot G_i(\mathbf{x}) \cdot (G_i(\mathbf{x}) \ge 0))$$

$$weight_i = \frac{1/G_i \max}{\sum_{i=1}^{m} (1/G_i \max)}, \; i = 1, 2, \ldots m$$

DMS-PSO for Constrained Optimization

- **Step 1:** Initialization -

  Initialize *ps* particles (position **X** and velocity **V**), calculate $f(\mathbf{X})$, $G_j(\mathbf{X})$ (j=1,2...,m) for each particle.

- **Step 2:** Divide the population into sub-swarms and assign *obj* for each sub-swarm using the novel constraint-handling mechanism, calculate the mean value of **Pc** (except in the first generation, mean(**Pc**)=0.5), calculate *Pc* for each particle. Then empty **Pc**.

- **Step 3:** Update the particles according to their objectives; update **pbest** and **gbest** of each particle according to the same comparison criteria, record the *Pc* value if **pbest** is updated.

DMS-PSO for Constrained Optimization

- **Step 5:** Local Search-

  Every *L* generations, randomly choose 5 particles' **pbest** and start local search with Sequential Quadratic Programming (SQP) method using these solutions as start points (fmincon(…,…,…) function in Matlab is employed). The maximum fitness evaluations for each local search is *L_FEs.*

- **Step 6:** If $FEs \leqslant 0.7*Max\_FEs$, go to Step 3. Otherwise go to Step 7.

- **Step 7:** Merge the sub-swarms into one swarm and continue PSO (Global Single Swarm). Every *L* generations, start local search using **gbest** as start points using $5*L\_FEs$ as the Max FEs. Stop search if $FEs \geqslant Max\_FEs$

SaDE for Constrained Optimization

- Strategy Adaptation

  Probabilistically select one out of several available learning strategies to apply for each individual in the current population

DE/Rand/1: $$\mathbf{V}_{i,G} = \mathbf{X}_{r_1,G} + F \cdot \left(\mathbf{X}_{r_2,G} - \mathbf{X}_{r_3,G}\right)$$

DE/Current to best/2: $$\mathbf{V}_{i,G} = \mathbf{X}_{i,G} + F \cdot \left(\mathbf{X}_{best,G} - \mathbf{X}_{i,G}\right) + F \cdot \left(\mathbf{X}_{r_1,G} - \mathbf{X}_{r_2,G} + \mathbf{X}_{r_3,G} - \mathbf{X}_{r_4,G}\right)$$

DE/Rand/2: $$\mathbf{V}_{i,G} = \mathbf{X}_{r_1,G} + F \cdot \left(\mathbf{X}_{r_2,G} - \mathbf{X}_{r_3,G}\right) + F \cdot \left(\mathbf{X}_{r_4,G} - \mathbf{X}_{r_5,G}\right)$$

DE/Current-to-rand/1: $$U_{i,G} = \mathbf{X}_{r_1,G} + K \cdot \left(\mathbf{X}_{r_3,G} - \mathbf{X}_{i,G}\right) + F \cdot \left(\mathbf{X}_{r_1,G} - \mathbf{X}_{r_2,G}\right)$$

Self-adaptive Differential Evolution

- Initial probabilities $p1=p2=p3=p4=0.25$
- According to the probability, we apply Stochastic Universal Selection to select the strategy for each individual in the current population.
- ns_i (nf_i), $i=1,2,3,4$: the accumulated number of trial vectors, *successfully entering (discarded)* the next generation while generated by each strategy
- ns_i and nf_i are accumulated within a specified number of generations, called the "*learning period (LP)".* The probability p_i is updated as:

$$p_i = \frac{ns_i}{ns_i + nf}$$

Self-adaptive Differential Evolution

Parameters adaptation

F: different random values *normrnd(0.5,0.3)* in the range *(0,2]* for different individuals

CR: accumulating the previous learning experience within a certain generational interval so as to dynamically adapt the value of *CR* to a suitable range

```
IF REM (G, LP)=0
    CRm=mean(CRpool)
END IF
```

```
IF REM (G, 5)=0
    FOR i =1 to NP
        CRi=normrand(CRm,0.1)
    END FOR
END IF
```

Extend SaDE to Handle Constraints

Selection procedure

The trial vector $U_{i,G}$ is compared to its corresponding target vector $X_{i,G}$ in the current population considering both the fitness value and constraints.

$U_{i,G}$ will replace $X_{i,G}$ if any of the following conditions is true

1. *$U_{i,G}$ is feasible, $X_{i,G}$ is not.*
2. *$U_{i,G}$ and $X_{i,G}$ are both feasible, and $U_{i,G}$ has smaller or equal fitness value (for minimization problem) than $X_{i,G}$.*
3. *$U_{i,G}$ and $X_{i,G}$ are both infeasible, but $U_{i,G}$ has a smaller overall constrain violation.*

Local Search

- To speed up the convergence, we apply a local search procedure once every 500 generations

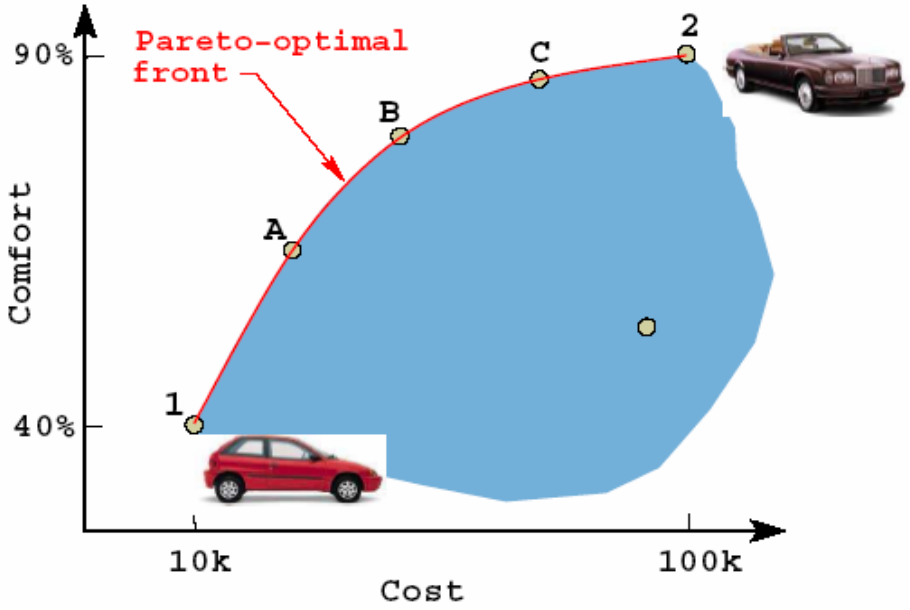
- n=5% of NP individuals

  DE_gbest + randomly selected *n-1* individuals from the best 50% individuals in the current population

- We employ the Sequential Quadratic Programming (SQP) method as the local search method.

V - Multi-objective Problems [D01]

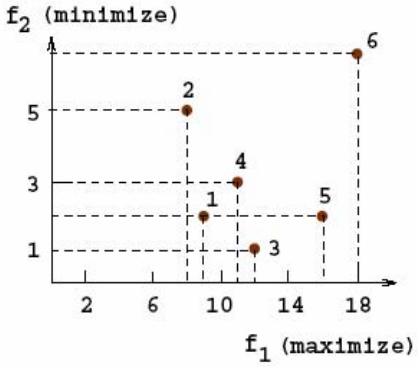
- Many real-world problems involve multiple, conflicting objectives



Relates to the concept of domination

$\mathbf{x}^{(1)}$ dominates $\mathbf{x}^{(2)}$ if

1. $\mathbf{x}^{(1)}$ is no worse than $\mathbf{x}^{(2)}$ in all objectives
2. $\mathbf{x}^{(1)}$ is strictly better than $\mathbf{x}^{(2)}$ in at least one objective



Applications: [CVL02], [SP05], [MB06], [TLL05]

- Robotics and control engineering
- Transport engineering
- Scheduling
- Finance
- Bioinformatics
- Pattern recognition
- PID design

- Non-dominated solutions: In a set of solutions P, the non-dominated set of solutions P′ are those that are not dominated by any member of the set P .
- Pareto-optimality: When the set P is the entire search space, the resulting P′ is called the Pareto-optimal set.

NANYANG TECHNOLOGICAL UNIVERSITY

Multi-Objective Optimization

- Mathematically, we can use the following formula to express the multi-objective optimization problems (MOP):

$$\text{Minimize} \quad \mathbf{y} = f(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \mathrm{K}, f_m(\mathbf{x})) \quad \mathbf{x} \in [\mathbf{Xmin}, \mathbf{Xmax}]$$

$$\text{subject to} \quad g_j(\mathbf{x}) \le 0,\ j = 1, \ldots, z$$

$$h_k(\mathbf{x}) = 0, k = q+1, \ldots, m$$

- The objective of multi-objective optimization is to find a set of solutions which can represent the Pareto-optimal set well, thus there are two goals for the optimization:
  - 1) Convergence to the Pareto-optimal set
  - 2) Diversity of solutions in the Pareto-optimal set

Representative MOEAs [CJKO01][D01][ZLT01][CVL02][AJG05][TKL05]

- **Non-elitist MOEAs**
  - Weight based GA (WBGA)
  - Multiple objective GA (MOGA)
  - Niched Pareto GA
  - Non-dominated sorting GA(NSGA)
- **Elitist MOEAs**
  - Distance-based Pareto GA (DPGA)
  - Strength Pareto GA (SPEA), SPEA-II
  - Non-dominated GA-II (NSGA-II)
  - Pareto-archived ES (PAES),
  - Pareto envelope-based selection algorithm (PESA), PESA-II
  - Multi-objective messy GA (MOMGA)

From PSO to MOPSO

- Mainly used techniques:
  - External archive for the non-dominated solution set.
  - How to update **pbest** and **gbest** (or **lbest**)
    - Execute non-domination comparison with **pbest** or **gbest**
    - Execute non-domination comparison among all particles' **pbest**s and their offspring in the entire population
  - How to choose **gbest** (or **lbest**)
    - Choose **gbest** (or **lbest**) from the recorded non-dominated solutions
    - Choose good local guides
  - How to keep diversity
    - Crowding distance sorting
    - Subpopulation

From CLPSO and DMS-PSO to MOPSOs

- Combine an external archive which is used to record the non-dominated solutions found so far.
- Use Non-dominated Sorting and Crowding distance Sorting which have been used in NSGAII (Deb *et al*., 2002) to sort the members in the external archive.
- Choose exemplars (CLPSO) or **lbest** (DMS-PSO) from the non-dominated solutions recorded in the external archive.
- Experiments show that MO-CLPSO and DMS-MO-PSO are both capable of converging to the true Pareto optimal front and maintaining a good diversity along the Pareto front.

Selection of *pbest*, *gbest* in MO-CLPSO [HSL06]

Selection of *pbest*

---

if X_i dominates $pbest_i$, $pbest_i = X_i$

if $pbest_i$ and X_i are non-dominated with each other,

if $rand < 0.5$, $pbest_i = X_i$

Selection of *gbest in MOCLPSO*

- all the non-dominated solutions are good individuals
- Randomly choose a particle from the non-dominated solutions.
- Other alternatives would be to form grids in the objective space and to select representatives from each cells, or to select more from less crowded cells, etc.

MO-CLPSO Algorithm

1) Initialize

Randomly initialize particle positions, Initialize particle velocities

Evaluate the fitness values of particles, initialize the external archive.

2) Optimize

WHILE stopping criterion is not satisfied

DO

For i=1 to NP

Select *gbest* from external archive

Assign each dimension to learn from *gbest*, *pbest* of this particle and *pbests* of other particles,

MO-CLPSO Algorithm (cont.)

Update particle velocity

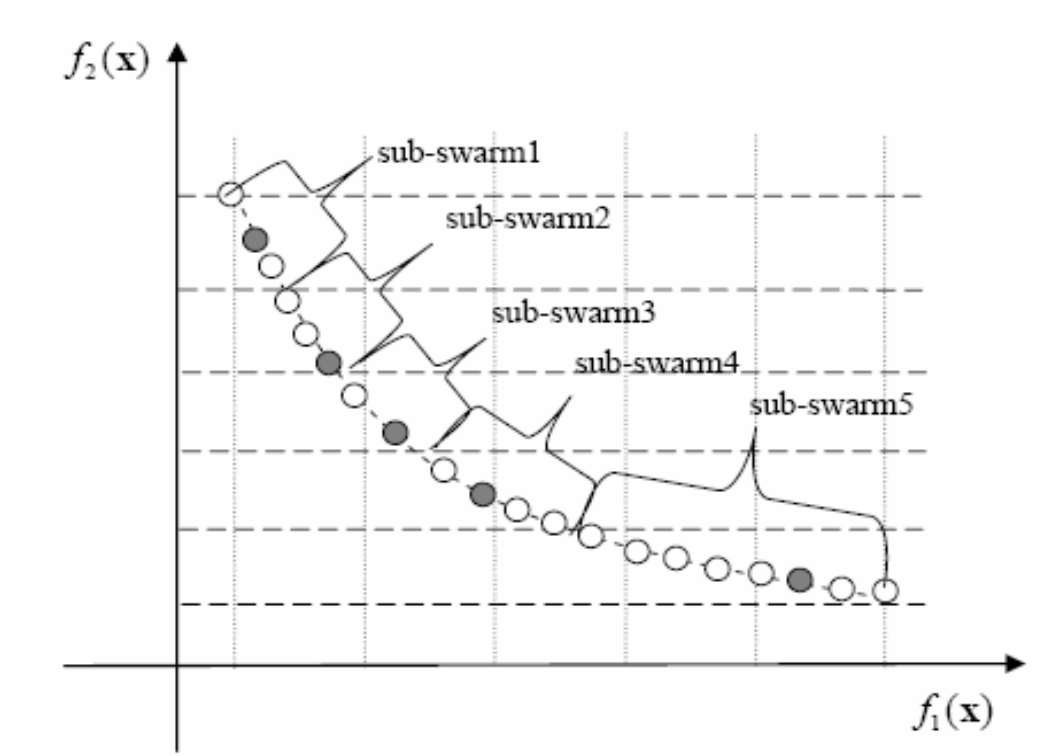
$$\begin{cases} if \quad a_i(d) == 1 \\ \quad V_i(d) = \omega_k * V_i(d) + rand() * (gbest(d) - X_i(d)) \\ if \quad b_i(d) == 1 \\ \quad V_i(d) = \omega_k * V_i(d) + rand() * (pbest_{fi(d)}(d) - X_i(d)) \\ else \\ \quad V_i(d) = \omega_k * V_i(d) + rand() * (pbest_i(d) - X_i(d)) \end{cases}$$

Update particle position $X_i(d) = X_i(d) + V_i(d)$

Evaluate the fitness values of particle
Update *pbest* if current position is better than *pbest*
End For
Update the external archive
Increment the generation count
END WHILE

MO-DMS-PSO Algorithm

- The solutions in the external archive are sorted based on one randomly chosen objective and then partitioned into *n* groups where *n* is the number of sub-swarms.
- Each sub-swarm randomly selects one representative as the gbest from each partition of the external archive.



MOSaDE

- MOSaDE is an extension of SaDE to optimize problems with multiple objectives.
- Similar to SaDE, the MOSaDE algorithm automatically adapts the trial vector generation strategies and their associated parameters according to their previous experience of generating promising or inferior individuals.
- However, when extending the single-objective algorithm to multi-objective domain, the evaluation criteria of promising or inferior individuals must be changed.

MOSaDE

We use the following:

Individual A is better than individual B, if

(1) individual A dominates B, or

(2) individual A and individual B are non-dominated with each other, but A is less crowded than individual B.

Therefore, in case that the trial vector is better than the target vector according to this criterion, we will record the associated parameter and strategy.

MOSaDE

The strategies incorporated into our proposed MOSaDE algorithm are ‘rand/1/bin’ and ‘best/2/bin’.

The $\mathbf{X}_{best}$ in ‘best/2/bin’ is randomly selected from external archive.

MOSaDE Algorithm

Step 1. Randomly initialize a population of *NP* individuals. Initialize strategy probability (*pk, k=1*,…,*K*, *K* is the no. of available strategies), the median value of *CR*(*CRmk*) for each strategy, learning period (*LP=50*) .

Step 2. Evaluate the individuals in the population, and fill the external archive with these individuals.

MOSaDE Algorithm

Step 3.Repeat

(1) Calculate strategy probability p_k: the percentage of the success rate of trial vectors generated by each strategy during the learning period.

(2) Assign trial vector generation strategy and parameters to each target vector X_i

(a) Use stochastic universal sampling to select one strategy *k* for each target vector X_i

(b) Assign control parameters *F* and *CR*

F: Generate the *F* values under *Normrnd*(0.3,0.1)

CR: After the first *LP* generation, calculate CRm_k according to the recorded *CR* values. Generate the *CR* values under *Normrnd*(CRm_k,0.1)

(3) Generate a new population where each trial vector is generated according to associated trial vector generation strategy *k* and parameter *F* and *CR* in (2).

(4) Selection:

FOR *i=1:NP*

(a) Evaluate the trial vector $\mathbf{U}_i^k$, and compare with the target vector $\mathbf{X}_{n(i)}$ nearest to the trial vector in the solution space.

IF $\mathbf{X}_{n(i)}$ dominates $\mathbf{U}_i^k$, discard $\mathbf{U}_i^k$.

ELSE

IF $\mathbf{U}_i^k$ dominates $\mathbf{X}_{n(i)}$,replace $\mathbf{X}_{n(i)}$ with $\mathbf{U}_i^k$;

IF non-dominated with each other, choose less crowded one to be the new target vector;

$\mathbf{U}_i^k$ will enter the external archive if (i) $\mathbf{U}_i^k$ dominates some individual(s) of the archive (the dominated individuals in the archive are deleted); or (ii) $\mathbf{U}_i^k$ is nondominated with archived individuals

END IF

(b) If trial vector is better than $\mathbf{X}_{n(i)}$, record the associated parameter *CR* and flag strategy *k* as successful strategy. Otherwise, flag strategy *k* as failed strategy.

(c) When the external archive exceeds the maximum specified size, we select the less crowded individuals based on harmonic average distance to keep the archive size.

END FOR

Local search

- Use local search to further improve solutions found by the MOSaDE algorithm.
- Employ the Quasi-Newton method as the local search method, considering only one objective randomly selected each time.
- The local search procedure is applied once every 200 generations, on 10 individuals randomly selected among the non-dominated solutions that were not applied local search previously.

VI - *Niching Methods* [SD-06]: *Fitness sharing*

- Fitness sharing [GR-87] modifies the search landscape by reducing the fitness of individuals in densely-populated regions. A sharing radius σ_s is used to determine whether two individuals share the same niche.
- Reducing an individual's fitness is controlled by two operations, a similarity function and a sharing function. The shared fitness f_i' is given by the formula as below:

$$f_i' = f_i \Big/ \sum_{j=1}^{N} sh(d_{ij})$$

$$\text{with} \quad sh(d_{ij}) = \begin{cases} 1-\left(d_{ij}/\sigma_s\right)^{\alpha} & \text{if } d_{ij} < \sigma_s \\ 0 & \text{otherwise} \end{cases}$$

where f_i denotes the original fitness of the individual ***i***, ***N*** the population size, and d_{ij} the distance between the individual ***i*** and the individual ***j***. α is a constant parameter which regulates the shape of the sharing function ***sh*** (typically α=1).

- The effect of this scheme is to encourage search in unexplored regions. The weakness of this method lies in the fact that it requires a *priori* knowledge about the distance between the peaks in the search space.

k-means clustering based

- *K*-means clustering algorithm is used to divide the population into niches [YG-93]. The fitness is calculated based on the distance d_{ic} between the individual and its niche centroid.
- The final fitness of an individual is calculated by the relation:

$$F_i = \frac{f_i}{n_c \left(1 - \left(d_{ic}/2d_{\max}\right)^{\alpha}\right)}$$

n_c is the number of individuals in the niche containing individual i, d_{max} is the maximum distance allowed between an individual and its niche centroid, and α is a constant.

- The formation of the niches is based on the adaptive *K*-mean algorithm. The algorithm begins with a fixed number (*k*) of seed points taken as the best *k* individuals.
- Using a minimum allowable distance d_{min} between niche centroids, a few clusters are formed from the seed points.
- The remaining population members are then added to these existing clusters or are used to form new clusters based on d_{min} and d_{max}. These computations are performed in each generation.

Deterministic crowding (DC)

- Deterministic crowding [M-95] is an extension of a technique first used by De Jong to help promote diverse populations [D-75]. After crossover and mutation, the offspring then replace their closest parent if it has a better fitness.
- Calculate the distances between $\boldsymbol{p_1}$ and $\boldsymbol{c_1}$, $\boldsymbol{p_2}$ and $\boldsymbol{c_2}$, $\boldsymbol{p_1}$ and $\boldsymbol{c_2}$, $\boldsymbol{p_2}$ and $\boldsymbol{c_1}$, and name them $\boldsymbol{d_1}$, $\boldsymbol{d_2}$, $\boldsymbol{d_3}$, $\boldsymbol{d_4}$ respectively.
  - If $\boldsymbol{d_1}+\boldsymbol{d_2} <= \boldsymbol{d_3}+\boldsymbol{d_4}$, then
  - If the fitness of $\boldsymbol{c_1}$ is higher than the fitness of $\boldsymbol{p_1}$, replace $\boldsymbol{p_1}$ with $\boldsymbol{c_1}$;
  - If the fitness of $\boldsymbol{c_2}$ is higher than the fitness of $\boldsymbol{p_2}$, replace $\boldsymbol{p_2}$ with $\boldsymbol{c_2}$.
  - Else
  - If the fitness of $\boldsymbol{c_2}$ is higher than the fitness of $\boldsymbol{p_1}$, replace $\boldsymbol{p_1}$ with $\boldsymbol{c_2}$;
  - If the fitness of $\boldsymbol{c_1}$ is higher than the fitness of $\boldsymbol{p_2}$, replace $\boldsymbol{p_2}$ with $\boldsymbol{c_1}$.
- Deterministic crowding uses a distance measure to determine similarity between individuals. As, DC does not require the use of a similarity radius, this relaxes the requirement of a *priori* domain knowledge and makes DC more suitable for difficult problems than fitness sharing. DC is an elitist niching method. This means that once a peak is discovered, it is never lost from the population.

Restricted tournament selection (RTS)

- RTS [H-94] adapts tournament selection for multimodal optimization. It initially selects two elements from the population to undergo crossover and mutation. Then a random sample of ***w*** individuals is taken from the population to be compared with each offspring created, and the most similar (or the closest) individual is chosen to compete with the offspring. If the offspring wins, it is allowed to enter the population.

Clearing

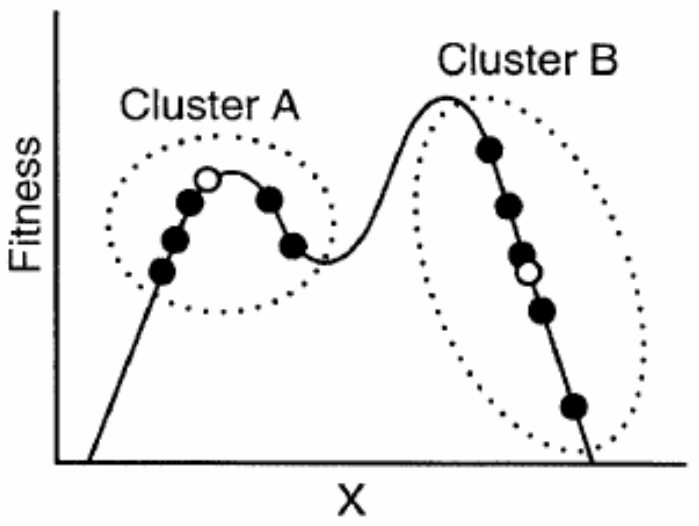
- Clearing [P-96] is best described as a variant of the sharing technique.
- Instead of sharing resources between all individuals of a same niche as in the fitness sharing scheme, clearing attributes them only to the best few members of the niche and removes the inferior individuals. The remaining individuals form the mating pool and generate offspring.

```
void Clearing(double Sigma, int Kappa) {
int i , j , nbWinners;
SortFitness (P);
for (i=0 ; i<n ; i++) {
if ( Fitness (P[i]) > 0) {
nbWinners = 1;
for (j = i+1 ; j< n-1 ; j++) {
if (Fitness(P[j])>0 && Distance (P[I], P[j])<Sigma){
if (nbWinners < Kappa) {
nbWinners++; }
else {
Fitness (P[j]) = 0.0;}
}}}}
```

the algorithm of Clearing

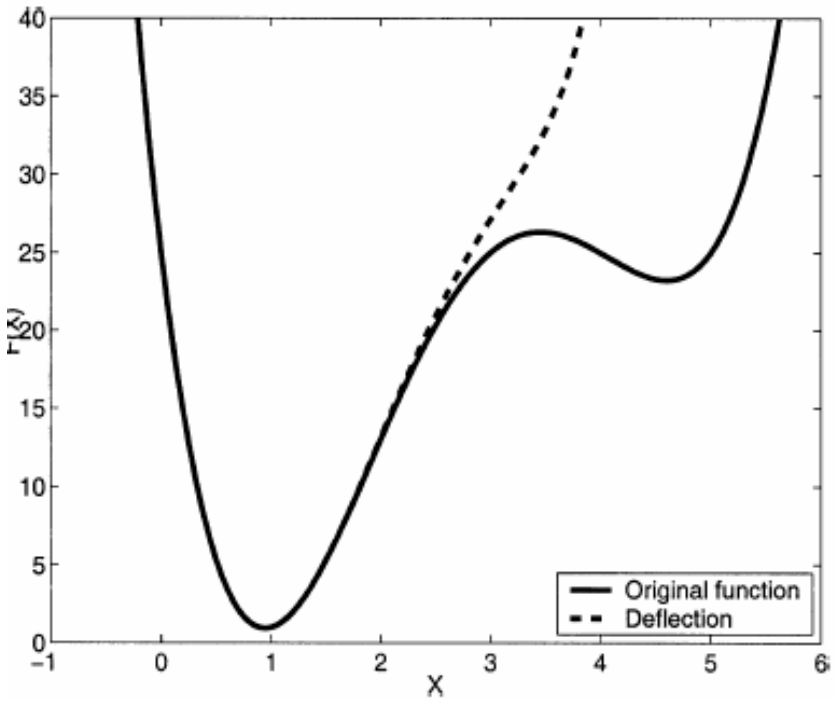
K-means Clustering-based Niched PSO

- Kennedy proposed the niched PSO using *K*-means clustering [K-00].
- The gbest / pbest / lbest were replaced by cluster centers or the best particle of each cluster to obtain several variants.
- Clustering-based variants performed better than the original PSO.



Deflection, Stretching, Repulsion based Niched PSO

- Parsopoulos [PV04b] et. al made use of deflection, stretching, repulsion, etc. to locate as many optima as possible.
- These techniques transform the objective function to make previously obtained local optima to have high function values (or low fitness).



NichePSO [BEB07]

(1) Initialize the main particle swarm.
(2) Train the main swarm particles using one iteration of the *cognition only* model.
(3) Update the fitness of each main swarm particle.
(4) For each subswarm:
  (a) Train subswarm particles using one iteration of the GCPSO algorithm.
  (b) Update each particle's fitness.
  (c) Update swarm radius
(5) If possible, merge subswarms.
(6) Allow subswarms to absorb any particles from the main swarm that moved into it.
(7) Search the main swarm for any particle that meets the partitioning criteria. If any is found, create a new subswarm with this particle and its closest neighbor.
(8) Repeat from 2 until stopping criteria are met.

Fig. 1. NichePSO algorithm.

VII - BINARY PSO ALGORITHM

- Binary PSO (K&E97)
- Sigmoid function
  - Force the real values between 0 and 1
- Velocity is updated with traditional equation
- Sigmoid function is used to squash them to be within [0,1]
  - $s(v_{ij})=1/(1+exp(-v_{ij}))$
  - $X_{ij}=1$ **if** $r \leqslant s(v_{ij})$
  - $X_{ij}=0$ **if** $r > s(v_{ij})$
  - r=uniform random number

Angle Modulated PSO / DE [PFE05, PEF06]

$$g(x) = sin(2\pi(x-a) \times b \times cos(A)) + d$$

$$\text{where} \quad A = 2\pi \times c(x-a)$$

- *a, b, c* and *d* are real valued variables to be optimized by the PSO or DE.

- If there are 10 binary variables, *x* takes 10 different values, for example, from 1 to 10.

- For every solution of "*a, b, c* and *d*" binary bits are generated by *sign*(*g*(*x*)) operation (as *x* runs from 1 to 10 in the case 10 bit problem.

VIII - Benchmarking Evolutionary Algorithms

- CEC05 comparison results (Single obj. + bound const.)
- CEC06 comparison results (Single obj + general const.)
- Experimental Results on MOPSOs
- CEC07 comparison results on MOEAs

- CEC benchmarking resources available from

http://www.ntu.edu.sg/home/epnsugan/

CEC'05 Comparison Results

- Algorithms involved in the comparison:
  - **BLX-GL50 (Garcia-Martinez & Lozano, 2005):** Hybrid Real-Coded Genetic Algorithms with Female and Male Differentiation
  - **BLX-MA (Molina *et al.*, 2005):** Adaptive Local Search Parameters for Real-Coded Memetic Algorithms
  - **CoEVO (Posik, 2005):** Mutation Step Co-evolution
  - **DE (Ronkkonen *et al.*,2005):**Differential Evolution
  - **DMS-L-PSO:** Dynamic Multi-Swarm Particle Swarm Optimizer with Local Search
  - **EDA (Yuan & Gallagher, 2005):** Estimation of Distribution Algorithm
  - **G-CMA-ES (Auger & Hansen, 2005):** A restart Covariance Matrix Adaptation Evolution Strategy with increasing population size
  - **K-PCX (Sinha *et al.*, 2005):** A Population-based, Steady-State real-parameter optimization algorithm with parent-centric recombination operator, a polynomial mutation operator and a niched -selection operation.
  - **L-CMA-ES (Auger & Hansen, 2005):** A restart local search Covariance Matrix Adaptation Evolution Strategy
  - **L-SaDE (Qin & Suganthan, 2005):** Self-adaptive Differential Evolution algorithm with Local Search
  - **SPC-PNX (Ballester *et al.*,2005):** A steady-state real-parameter GA with PNX crossover operator

CEC'05 Comparison Results

- **Problems:** 25 minimization problems (Suganthan *et al*. 2005)
- **Dimensions:** *D*=10, 30
- **Runs / problem:** 25
- **Max_FES:** 10000**D* (Max_FES_10D= 100000; for 30D=300000; for 50D=500000)
- **Initialization:** Uniform random initialization within the search space, except for problems 7 and 25, for which initialization ranges are specified. The same initializations are used for the comparison pairs (problems 1, 2, 3 & 4, problems 9 & 10, problems 15, 16 & 17, problems 18, 19 & 20, problems 21, 22 & 23, problems 24 & 25).
- **Global Optimum**: All problems, except 7 and 25, have the global optimum within the given bounds and there is no need to perform search outside of the given bounds for these problems. 7 & 25 are exceptions without a search range and with the global optimum outside of the specified initialization ranges.

CEC'05 Comparison Results

- **Termination:** Terminate before reaching Max_FES if the error in the function value is 10^{-8} or less.
- **Ter_Err**: 10^{-8} (termination error value)
- **Successful Run:** A run during which the algorithm achieves the fixed accuracy level within the Max_FES for the particular dimension.
- **Success Rate**= **(**# of successful runs) / total runs
- **Success Performance** = mean (FEs for successful runs)*(# of total runs) / (# of successful runs)

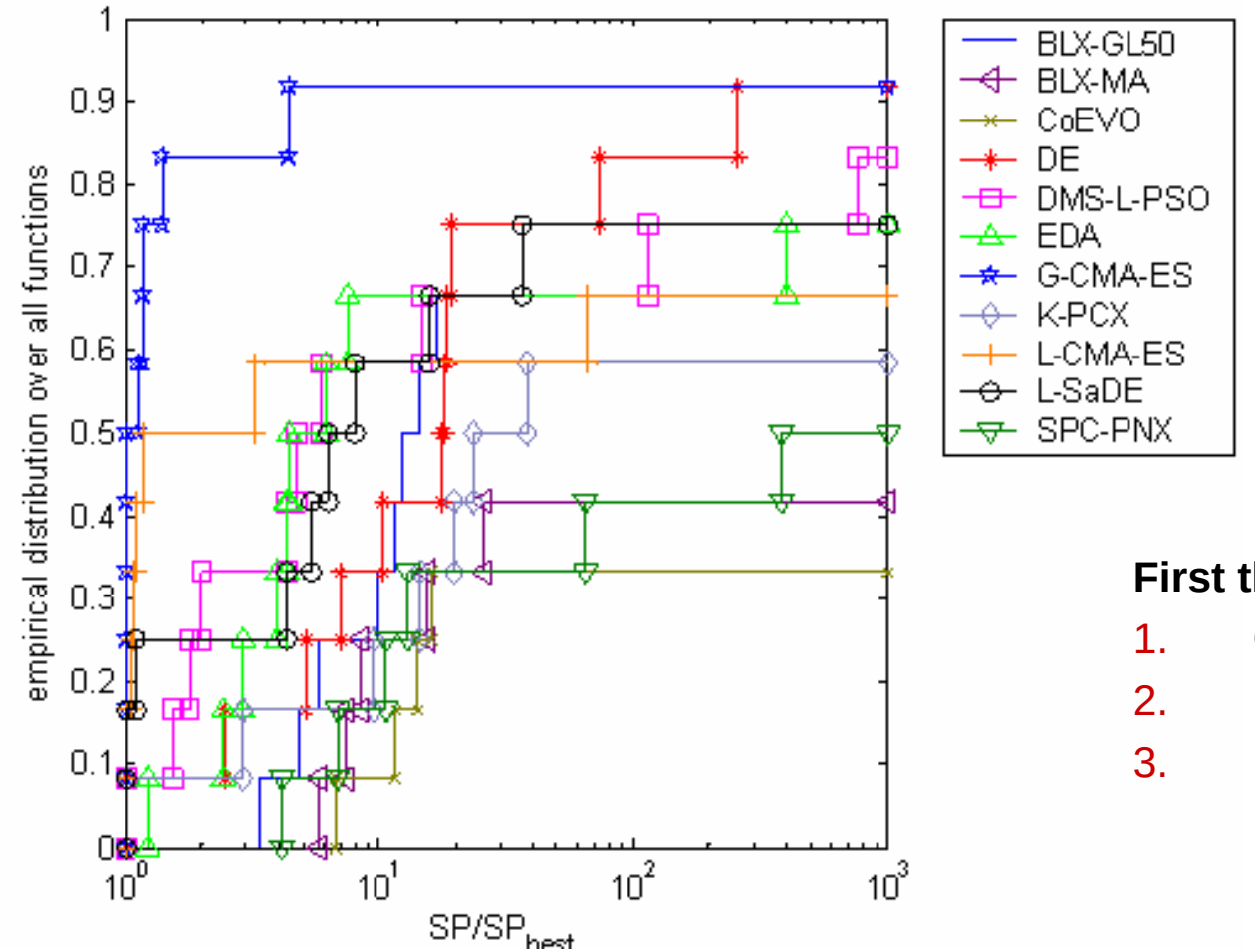
CEC'05 Comparison Results

Success Rates of the 11 algorithms for 10-D

Func / Algorithms	1	2	3	4	5	6	7	9	10	11	12	15
BLX-GL50	100%	100%	0%	100%	100%	100%	36%	12%	0%	0%	52%	0%
BLX-MA	100%	100%	0%	0.96	0%	0%	0%	72%	0%	0%	0%	20%
CoEVO	100%	100%	100%	100%	0%	0%	0%	0%	0%	0%	0%	0%
DE	100%	100%	80%	100%	100%	96%	6%	44%	0%	48%	76%	4%
DMS-L-PSO	100%	100%	100%	4%	100%	100%	16%	100%	0%	0%	80%	84%
EDA	100%	100%	92%	100%	100%	88%	4%	0%	0%	12%	40%	0%
G-CMA-ES	100%	100%	100%	100%	100%	100%	100%	76%	92%	24%	88%	0%
K-PCX	100%	100%	0%	84%	0%	40%	20%	96%	88%	0%	0%	0%
L-CMA-ES	100%	100%	100%	28%	100%	100%	100%	0%	0%	0%	48%	0%
L-SaDE	100%	100%	64%	96%	0%	100%	24%	100%	0%	0%	100%	92%
SPC-PNX	100%	100%	0%	100%	100%	0%	4%	4%	0%	0%	0%	0%

*In the comparison, only the problems in which at least one algorithm succeeded once are considered.

CEC'05 Comparison Results



First three algorithms:

1. G-CMA-ES
2. DE
3. DMS-L-PSO

Empirical distribution over all successful functions for 10-D (*SP* here means the Success Performance for each problem. *SP*=mean (FEs for successful runs)*(# of total runs) / (# of successful runs). SP_{best} is the minimal FES of all algorithms for each problem.)

NANYANG TECHNOLOGICAL UNIVERSITY

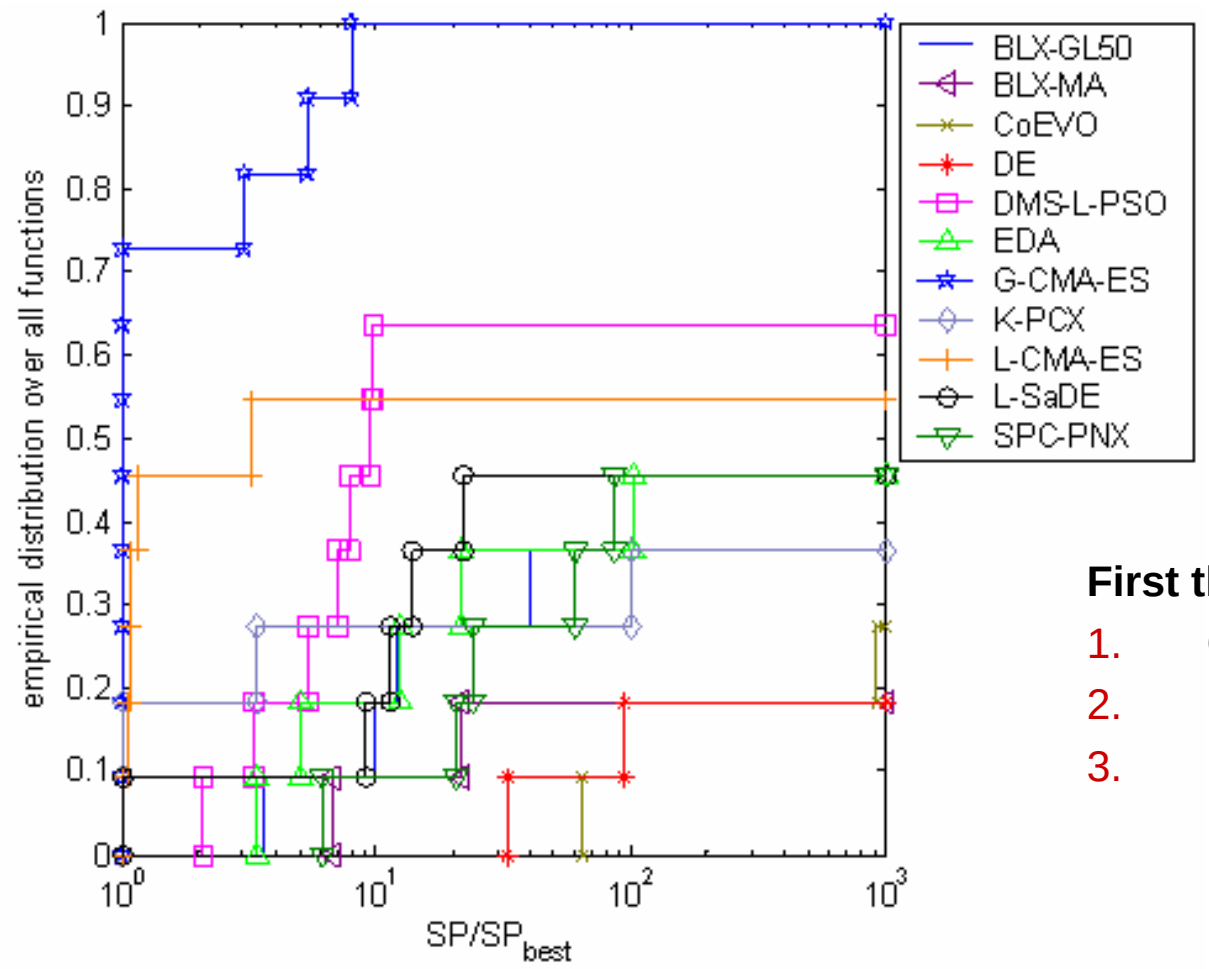
CEC'05 Comparison Results

Success Rates of the 11 algorithms for 30-D

Algorithms \ Func	1	2	3	4	5	6	7	9	10	11	12
BLX-GL50	100%	100%	0%	0%	0%	100%	100%	0%	0%	0%	0%
BLX-MA	100%	0%	0%	0%	0%	0%	0%	36%	0%	0%	0%
CoEVO	12%	32%	0%	0%	0%	0%	44%	0%	0%	0%	0%
DE	100%	0%	0%	0%	0%	0%	88%	0%	0%	0%	0%
DMS-L-PSO	100%	100%	88%	0%	0%	96%	96%	100%	0%	0%	20%
EDA	100%	100%	100%	100%	0%	0%	100%	0%	0%	0%	0%
G-CMA-ES	100%	100%	100%	40%	100%	100%	100%	36%	12%	4%	32%
K-PCX	100%	0%	0%	0%	0%	0%	44%	72%	56%	0%	0%
L-CMA-ES	100%	100%	100%	0%	100%	100%	100%	0%	0%	0%	0%
L-SaDE	100%	96%	0%	52%	0%	0%	80%	100%	0%	0%	0%
SPC-PNX	100%	88%	0%	76%	0%	4%	64%	0%	0%	0%	0%

*In the comparison, only the problems which at least one algorithm succeeded once are considered.

CEC'05 Comparison Results



First three algorithms:

1. G-CMA-ES
2. DMS-L-PSO
3. L-CMA-ES

Empirical distribution over all successful functions for 30-D (*SP* here means the Success Performance for each problem. *SP*=mean (FEs for successful runs)*(# of total runs) / (# of successful runs). *SPbest* is the minimal FES of all algorithms for each problem.)

NANYANG TECHNOLOGICAL UNIVERSITY

CEC'06 Comparison Results

Algorithms

- **DE (Zielinski & Laur, 2006):** Differential Evolution
- **DMS-C-PSO (Liang & Suganthan, 2006):** Dynamic Multi-Swarm Particle Swarm Optimizer with the New Constraint-Handling Mechanism
- **ε - DE [TS06]** Constrained Differential Evolution with Gradient-Based Mutation and Feasible Elites
- **GDE (Kukkonen & Lampinen, 2006)** : Generalized Differential Evolution
- **jDE-2 (Brest & Zumer, 2006):** Self-adaptive Differential Evolution
- **MDE (Mezura-Montes, *et al.* 2006):** Modified Differential Evolution
- **MPDE (Tasgetiren & Suganthan,2006):** Multi-Populated DE Algorithm
- **PCX (Ankur Sinha, *et al,* 2006):** A Population-Based, Parent Centric Procedure
- **PESO+ (Munoz-Žavala *et al, 2006*):** Particle Evolutionary Swarm Optimization Plus
- **SaDE (Huang *et al, 2006*):** Self-adaptive Differential Evolution Algorithm

CEC'06 Comparison Results

- **Problems:** 24 minimization problems with constraints (Liang, 2006b)
- **Runs / problem:** 25 (total runs)
- **Max_FES:** 500,000
- **Feasible Rate** = (# of feasible runs) / total runs
- **Success Rate** = (# of successful runs) / total runs
- **Success Performance** = mean (FEs for successful runs)*(# of total runs) / (# of successful runs)
  - The above three quantities are computed for each problem separately.
  - **Feasible Run:** A run during which at least one feasible solution is found in Max_FES.
  - **Successful Run:** A run during which the algorithm finds a feasible solution $\mathbf{x}$ satisfying $f(\mathbf{x}) - f(\mathbf{x}^*) \leq 0.0001$

CEC'06 Comparison Results

Algorithms' Parameters

DE	*NP*, *F*, *CR*
DMS-PSO	ω, c_1, c_2 , Vmax, *n*, *ns*, *R*, *L*, *L_FES*
ε_DE	*N*, *F*, *CR*, *Tc*, *Tmax*, *cp*, *Pg*, *Rg*, *Ne*
GDE	*NP*, *F*, *CR*
jDE-2	*NP*, *F*, *CR*, *k*, *l*
MDE	μ, *CR*, *Max_Gen*, λ, F_α, F_β,
MPDE	*F*, *CR*, *np1*, *np2*
PCX	*N* , λ, *r* (a different *N* is used for g02),
PESO+	ω , c_1, c_2 , *n*, *not sensitive to* ω , *c1*, *c2*
SaDE	*NP*, *LP*, *LS_gap*

CEC'06 Comparison Results

Success Rate for Problems 1-11

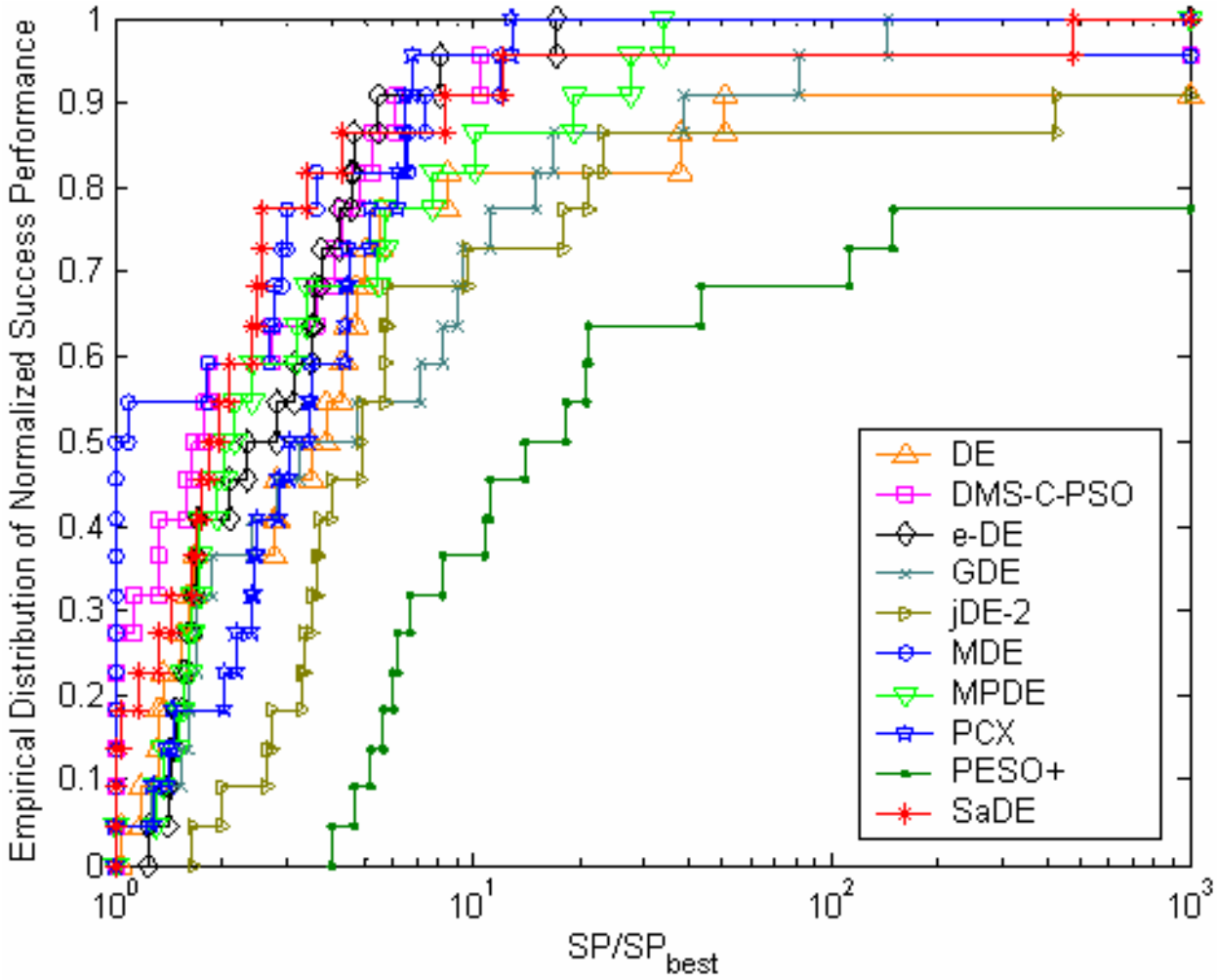
Algorithms \ Func		1	2	3	4	5	6	7	8	9	10	11
DE	81.64%	100%	84%	0%	100%	100%	100%	100%	100%	100%	100%	100%
DMS-C-PSO	95.27%	100%	96%	100%	100%	100%	100%	100%	100%	100%	100%	100%
e-DE	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
GDE	80.91%	100%	72%	4%	100%	92%	100%	100%	100%	100%	100%	100%
jDE-2	83.64%	100%	92%	0%	100%	68%	100%	100%	100%	100%	100%	96%
MDE	91.64%	100%	16%	100%	100%	100%	100%	100%	100%	100%	100%	100%
MPDE	91.64%	100%	92%	84%	100%	100%	100%	100%	100%	100%	100%	96%
PCX	98.36%	100%	64%	100%	100%	100%	100%	100%	100%	100%	100%	100%
PESO+	70.91%	100%	56%	100%	100%	100%	100%	96%	100%	100%	16%	100%
SaDE	91.09%	100%	84%	96%	100%	100%	100%	100%	100%	100%	100%	100%

CEC'06 Comparison Results

Success Rate for Problems 12-19,21,23,24

Func / Algorithms	12	13	14	15	16	17	18	19	21	23	24
DE	100%	32%	100%	100%	100%	20%	100%	100%	60%	0%	100%
DMS-C-PSO	100%	100%	100%	100%	100%	0%	100%	100%	100%	100%	100%
ε_-DE	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
GDE	100%	40%	96%	96%	100%	16%	76%	88%	60%	40%	100%
jDE-2	100%	0%	100%	96%	100%	4%	100%	100%	92%	92%	100%
MDE	100%	100%	100%	100%	100%	100%	100%	0%	100%	100%	100%
MPDE	100%	48%	100%	100%	100%	28%	100%	100%	68%	100%	100%
PCX	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
PESO+	100%	100%	0%	100%	100%	0%	92%	0%	0%	0%	100%
SaDE	100%	100%	80%	100%	100%	4%	92%	100%	60%	8%	100%

CEC'06 Comparison Results

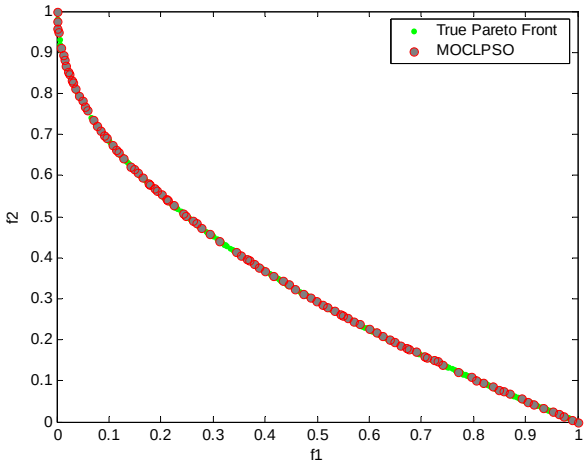


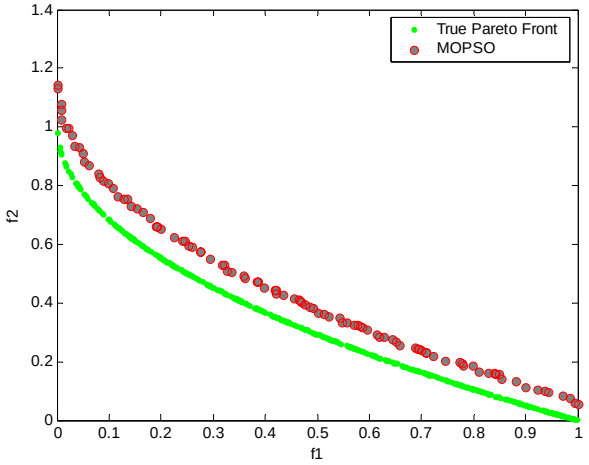
1st	ε_DE
2nd	DMS-PSO
3rd	MDE, PCX
5th	SaDE
6th	MPDE
7th	DE
8th	jDE-2
9th	GDE, PESO+

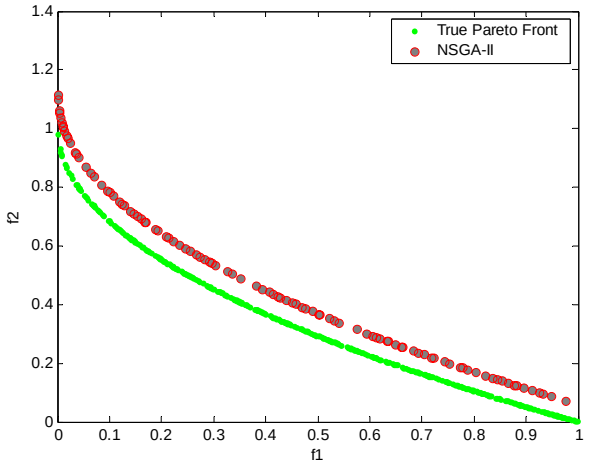
Empirical distribution over all functions(*SP* here means the Success Performance for each problem. *SP*=mean (FEs for successful runs)*(# of total runs) / (# of successful runs). SP_{best} is the minimal FES of all algorithms for each problem.)

NANYANG TECHNOLOGICAL UNIVERSITY

Results of MOCLPSO on ZDT1



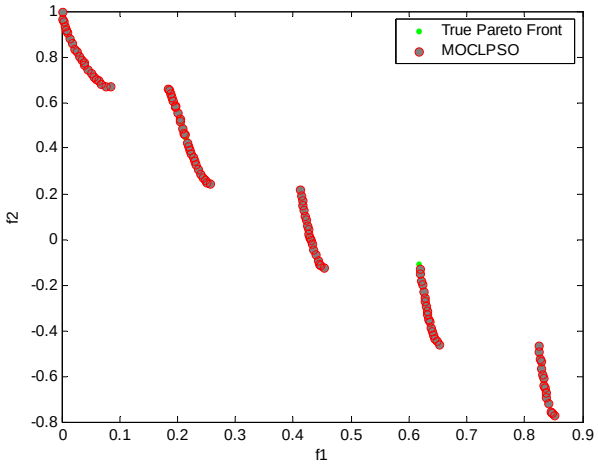


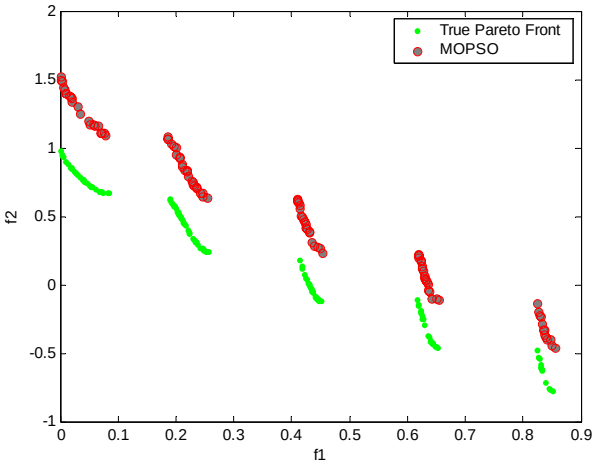


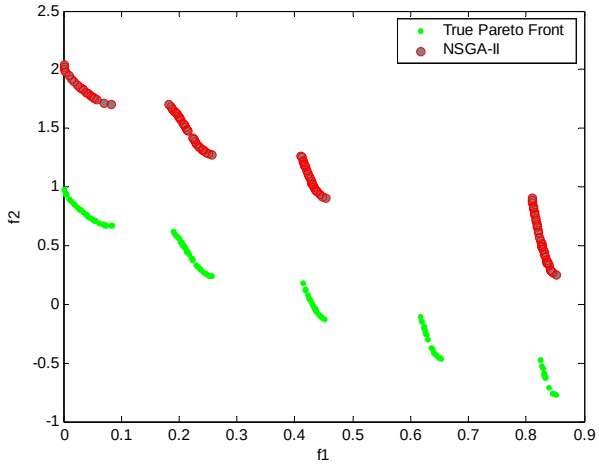
Test Problem3 (ZDT1)	Converge Metric γ			Diversity Metric Δ		
	MOCLPSO	MOPSO	NSGA-II	MOCLPSO	MOPSO	NSGA-II
Best	0.001945	0.063836	0.046512	0.251628	0.553405	0.434004
Worst	0.002455	0.124011	0.122316	0.351678	0.652837	0.790071
Average	**0.002235**	0.081128	0.075551	**0.304202**	0.591477	0.537488
Median	0.002232	0.072365	0.067414	0.307244	0.576528	0.493612
Variance	2.69E-08	4.30E-04	7.51E-04	1.02E-03	1.33E-03	1.35E-02

NSGA-II [D01]; MOPSO [CL04]; MOCLPSO [HSL06]

Results of MOCLPSO on ZDT3

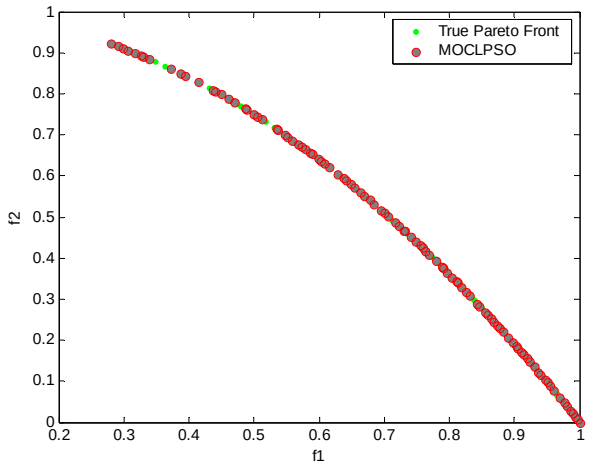


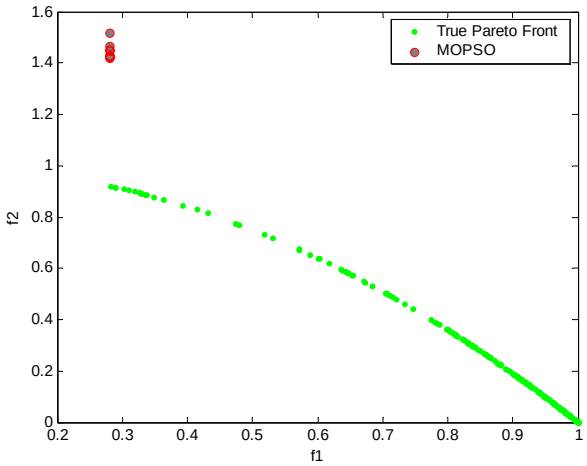


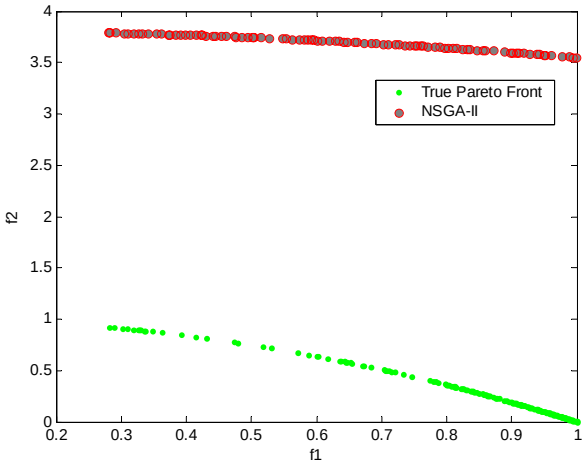


Test Problem5 (ZDT3)	Converge Metric γ			Diversity Metric Δ		
	MOCLPSO	MOPSO	NSGA-II	MOCLPSO	MOPSO	NSGA-II
Best	0.005689	0.113631	0.600920	0.503779	0.534863	0.736710
Worst	0.008323	0.227841	0.712746	0.646290	0.662133	0.844782
Average	**0.007334**	0.157366	0.645840	**0.554944**	0.607451	0.795274
Median	0.007517	0.145892	0.632486	0.549698	0.607451	0.796255
Variance	6.70E-07	1.33E-03	1.75E-03	2.23E-03	1.61E-03	1.19E-03

Results of MOCLPSO on ZDT6







Test Problem7 (ZDT6)	Converge Metric γ			Diversity Metric Δ		
	MOCLPSO	MOPSO	NSGA-II	MOCLPSO	MOPSO	NSGA-II
Best	0.002950	0.527778	2.498893	0.237585	0.788246	0.934297
Worst	0.015873	1.667019	3.707603	0.964462	1.016125	0.983176
Average	**0.006283**	1.029723	3.073925	**0.486495**	0.927657	0.954907
Median	0.004871	0.944908	3.023464	0.346380	0.929350	0.954590
Variance	1.77E-05	1.90E-01	1.40E-01	6.75E-02	5.30E-03	3.00E-04

NSGA-II [D01]; MOPSO [CL04]; MOCLPSO [HSL06]

NANYANG TECHNOLOGICAL UNIVERSITY

Results of MO-DMS-PSO

Convergence Metric (γ) comparison of the four algorithms

Algorithms		SCH	FON	KUR	ZDT1	ZDT2	ZDT3	ZDT4	ZDT6
NSGA-II	mean	0.0043	0.0021	0.0324	0.0674	0.1897	0.6211	5.1219	3.1209
	std	0.0004	0.0002	0.1074	0.0246	0.0615	0.0329	2.2526	0.3413
PAES	mean	0.0045	0.0360	1.0955	**0.0006**	**0.0005**	0.0745	3.5097	7.5964
	std	0.0004	0.1315	2.1724	0.0003	0.0003	0.0034	1.0967	0.8102
MOPSO	mean	0.0044	0.0013	0.0252	0.0189	0.0162	0.0267	5.6413	0.7501
	std	0.0004	0.0001	0.0041	0.0032	0.0099	0.0069	2.7814	0.4208
DMS	mean	**0.0044**	**0.0012**	**0.0162**	0.0018	0.0016	**0.0018**	**0.0018**	**0.0039**
	std	0.0003	0.0001	0.0015	0.0003	0.0004	0.0004	0.0008	0.0012
h		0	1	1	1	1	1	1	1

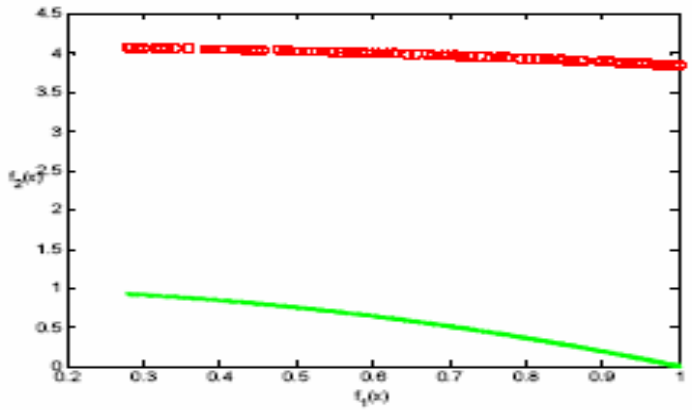
NSGA-II [D01]; MOPSO [CL04]; PAES [KC00]

Results of MO-DMS-PSO

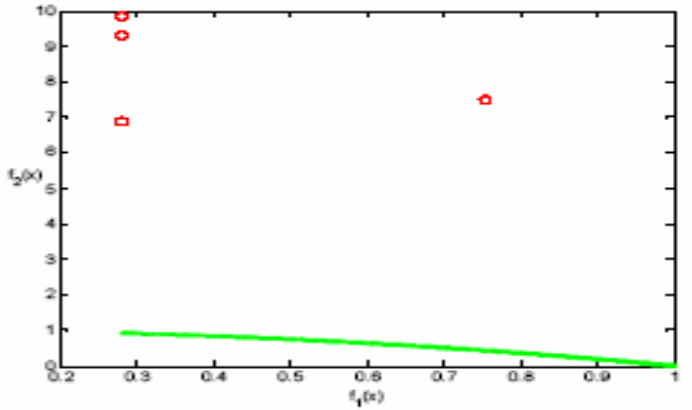
Diversity Metric (Δ) comparison of the four algorithms

Algorithms		SCH	FON	KUR	ZDT1	ZDT2	ZDT3	ZDT4	ZDT6
NSGA-II	mean	0.2823	0.4470	0.7680	0.5401	0.9482	0.7958	0.9445	0.9639
	std	0.0269	0.0335	0.0533	0.0454	0.1344	0.0148	0.0590	0.0212
PAES	mean	0.7416	0.5533	0.6896	1.1197	1.1539	0.9707	1.0108	0.8900
	std	0.0427	0.1276	0.0840	0.1561	0.1849	0.0331	0.1671	0.0815
MOPSO	mean	0.7618	0.5861	0.7849	0.5980	0.6810	0.7203	0.9720	1.0054
	std	0.0517	0.0413	0.0961	0.0470	0.2006	0.0322	0.0310	0.0741
DMS	mean	**0.1723**	**0.1371**	**0.2665**	**0.1615**	**0.1835**	**0.5015**	**0.1647**	**0.1573**
	std	0.0110	0.0142	0.0125	0.0109	0.1547	0.0155	0.0124	0.0129
h		1	1	1	1	1	1	1	1

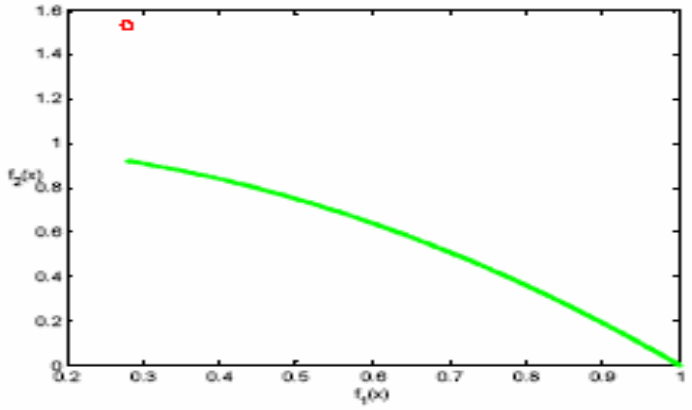
Results of MO-DMS-PSO



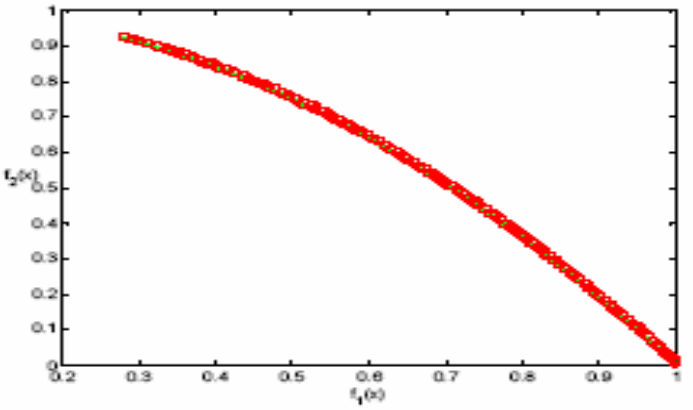
(a)NSGAII



(b)PAES



(c)MOPSO



(d)DMS-MOPSO

Fig. 6-14 *Pareto fronts generated by the four algorithms on ZDT6*

CEC07 comparison results on MOEAs: Evaluation Criteria

- Quantitative performance measurements, **R indicator** and **Hypervolume difference to a reference set** is used as a measure for the expected number of function evaluations to reach a target Pareto front.
- Invariance is a non-empirical statement on the ability to generalize performance results. Invariance guarantees identical performance on a class of functions. Possible invariances are invariance against translation, scaling, or even order preserving transformations of the objective function value invariance against angle preserving (rigid) transformations of the search space (translation, rotation)
- Parameters Settings
  - how many parameters of the algorithm need to be adjusted to the object function?
  - how many different settings were tested?
  - how many different settings were finally used?

References to Algorithms in CEC07 papers

- NSGAII_SBX: Sharma, Kumar *et al.*
- NSGAII_PCX: Kumar *et al.*
- GDE3: Kukkonen and Lampinen
- DEMOwSA: Zamuda *et al.*
- MOSaDE: Huang *et al.*
- MO_DE: Zielinski and Laur
- MO_PSO: Zielinski and Laur
- MTS: Tseng and Chen

CEC07 Function Sets

- Three subsets
  - 2-objective functions
  - 3-objective functions
  - 5-objective functions

- Comparison: Rank of the mean of the metric values from 25 runs

M=2, Rank(R indicator)

FES=5000

	Total	1.OKA2	2.SYMPART	3.S_ZDT1	4.S_ZDT2	5.S_ZDT4	6.R_ZDT4	7.S_ZDT6
NSGA2_SBX	**1.71**	2	3	1	1	2	2	1
NSGA2_PCX	5.43	4	1	8	8	8	1	8
GDE3	4.14	1	7	4	4	5	4	4
DEMOwSA	6.43	3	8	6	7	7	7	7
MOSaDE	4.14	8	4	3	3	3	5	3
MO_DE	5.71	5	6	5	5	6	8	5
MO_PSO	5.00	7	2	7	6	4	3	6
MTS	3.43	6	5	2	2	1	6	2

FES=50000

	Total	1.OKA2	2.SYMPART	3.S_ZDT1	4.S_ZDT2	5.S_ZDT4	6.R_ZDT4	7.S_ZDT6
NSGA2_SBX	3.14	2	5	2	2	2	6	5
NSGA2_PCX	4.86	1	7	6	3	3	4	7
GDE3	**2.43**	3	2	4	4	4	3	2
DEMOwSA	4.71	4	4	3	6	7	2	3
MOSaDE	3.57	8	6	1	1	1	1	1
MO_DE	4.14	5	3	5	7	6	5	4
MO_PSO	7.29	7	1	8	8	8	7	8
MTS	5.86	6	8	7	5	5	8	6

FES=500000

	Total	1.OKA2	2.SYMPART	3.S_ZDT1	4.S_ZDT2	5.S_ZDT4	6.R_ZDT4	7.S_ZDT6
NSGA2_SBX	3.43	2	5	2	2	2	6	5
NSGA2_PCX	4.43	1	7	6	3	3	4	7
GDE3	3.14	3	2	4	4	4	3	2
DEMOwSA	4.14	4	4	3	6	7	2	3
MOSaDE	**2.71**	8	6	1	1	1	1	1
MO_DE	5.00	5	3	5	7	6	5	4
MO_PSO	6.71	7	1	8	8	8	7	8
MTS	6.43	6	8	7	5	5	8	6

M=2, Rank(Hypervolumn)

FES=5000

	Total	1.OKA2	2.SYMPART	3.S_ZDT1	4.S_ZDT2	5.S_ZDT4	6.R_ZDT4	7.S_ZDT6
NSGA2_SBX	**1.86**	2	3	1	1	2	2	2
NSGA2_PCX	5.29	3	1	8	8	8	1	8
GDE3	4.43	1	7	4	4	6	4	5
DEMOwSA	6.71	4	8	7	7	7	7	7
MOSaDE	4.29	8	4	3	3	3	6	3
MO_DE	5.43	5	6	5	5	5	8	4
MO_PSO	4.71	6	2	6	6	4	3	6
MTS	3.29	7	5	2	2	1	5	1

FES=50000

	Total	1.OKA2	2.SYMPART	3.S_ZDT1	4.S_ZDT2	5.S_ZDT4	6.R_ZDT4	7.S_ZDT6
NSGA2_SBX	2.86	2	7	2	1	1	4	3
NSGA2_PCX	4.43	3	3	6	4	5	3	7
GDE3	**1.71**	1	1	1	2	4	1	2
DEMOwSA	4.43	5	2	3	5	7	5	4
MOSaDE	5.29	8	6	7	6	3	2	5
MO_DE	4.57	4	4	4	7	6	6	1
MO_PSO	7.29	7	5	8	8	8	7	8
MTS	5.43	6	8	5	3	2	8	6

FES=500000

	Total	1.OKA2	2.SYMPART	3.S_ZDT1	4.S_ZDT2	5.S_ZDT4	6.R_ZDT4	7.S_ZDT6
NSGA2_SBX	3.71	2	6	4	1	2	6	5
NSGA2_PCX	4.57	3	7	5	2	4	4	7
GDE3	**2.43**	1	2	2	4	3	3	2
DEMOwSA	4.00	6	4	1	5	7	2	3
MOSaDE	4.43	8	5	8	7	1	1	1
MO_DE	4.43	4	3	3	6	6	5	4
MO_PSO	6.57	7	1	7	8	8	7	8
MTS	5.86	5	8	6	3	5	8	6

M=3, Rank(R indicator)

FES=5000

	Total	8. S_DTLZ2	9. R_DTLZ2	10. S_DTLZ3	11. WFG1	12. WFG8	13. WFG9
NSGA2_SBX	1.46	1	4	1	4	7	2
NSGA2_PCX	1.85	8	5	3	2	3	3
GDE3	1.92	5	3	5	6	2	4
DEMOwSA	**1.31**	4	8	2	1	1	1
MOSaDE	3.08	2	7	7	8	8	8
MO_DE	2.31	3	6	6	5	4	6
MO_PSO	2.46	6	1	8	7	5	5
MTS	2.23	7	2	4	3	6	7

FES=50000

	Total	8. S_DTLZ2	9. R_DTLZ2	10. S_DTLZ3	11. WFG1	12. WFG8	13. WFG9
NSGA2_SBX	1.38	5	2	1	3	6	1
NSGA2_PCX	2.15	7	3	5	5	3	5
GDE3	**1.00**	3	1	2	4	1	2
DEMOwSA	1.62	2	8	4	2	2	3
MOSaDE	2.85	1	7	6	7	8	8
MO_DE	2.38	4	6	7	6	4	4
MO_PSO	2.92	6	5	8	8	5	6
MTS	2.31	8	4	3	1	7	7

FES=500000

	Total	8. S_DTLZ2	9. R_DTLZ2	10. S_DTLZ3	11. WFG1	12. WFG8	13. WFG9
NSGA2_SBX	1.38	5	1	1	2	6	3
NSGA2_PCX	2.23	7	3	6	6	3	4
GDE3	**1.00**	2	2	2	1	1	5
DEMOwSA	2.08	3	8	4	4	2	6
MOSaDE	2.15	1	5	3	3	8	8
MO_DE	2.23	4	6	7	7	4	1
MO_PSO	2.77	6	7	8	8	5	2
MTS	2.77	8	4	5	5	7	7

M=3, Rank(Hypervolumn)

FES=5000

	Total	8. S_DTLZ2	9. R_DTLZ2	10. S_DTLZ3	11. WFG1	12. WFG8	13. WFG9
NSGA2_SBX	**1.31**	1	3	2	8	1	2
NSGA2_PCX	2.15	8	5	5	2	4	4
GDE3	1.69	4	4	3	5	3	3
DEMOwSA	1.54	6	6	4	1	2	1
MOSaDE	3.23	5	8	6	7	8	8
MO_DE	2.38	2	7	7	4	5	6
MO_PSO	2.31	3	2	8	6	6	5
MTS	2.00	7	1	1	3	7	7

FES=50000

	Total	8. S_DTLZ2	9. R_DTLZ2	10. S_DTLZ3	11. WFG1	12. WFG8	13. WFG9
NSGA2_SBX	1.62	6	2	1	8	1	3
NSGA2_PCX	2.23	7	3	5	4	4	6
GDE3	**0.92**	3	1	2	3	2	1
DEMOwSA	1.69	2	8	6	1	3	2
MOSaDE	2.69	1	7	4	7	8	8
MO_DE	2.46	4	6	7	5	5	5
MO_PSO	2.62	5	5	8	6	6	4
MTS	2.38	8	4	3	2	7	7

FES=500000

	Total	8. S_DTLZ2	9. R_DTLZ2	10. S_DTLZ3	11. WFG1	12. WFG8	13. WFG9
NSGA2_SBX	1.77	6	1	2	8	1	5
NSGA2_PCX	2.31	7	3	5	5	4	6
GDE3	**0.92**	2	2	3	1	2	2
DEMOwSA	1.92	3	8	4	3	3	4
MOSaDE	1.92	1	5	1	2	8	8
MO_DE	2.23	4	6	7	6	5	1
MO_PSO	2.77	5	7	8	7	6	3
MTS	2.77	8	4	6	4	7	7

M=5, Rank(R indicator)

FES=5000

	Total	8. S_DTLZ2	9. R_DTLZ2	10. S_DTLZ3	11. WFG1	12. WFG8	13. WFG9
NSGA2_SBX	0.79	1	1	1	8	2	2
NSGA2_PCX	0.84	4	3	2	3	3	1
GDE3	1.42	5	5	4	5	4	4
DEMOwSA	1.89	8	8	5	2	6	7
MOSaDE	1.95	3	7	6	7	8	6
MO_DE	1.74	6	6	7	4	5	5
MO_PSO	2.11	7	4	8	6	7	8
MTS	**0.63**	2	2	3	1	1	3

FES=50000

	Total	8. S_DTLZ2	9. R_DTLZ2	10. S_DTLZ3	11. WFG1	12. WFG8	13. WFG9
NSGA2_SBX	**0.58**	1	1	1	6	1	1
NSGA2_PCX	1.63	8	4	8	5	4	2
GDE3	0.89	3	3	3	2	2	4
DEMOwSA	1.53	2	8	4	4	6	5
MOSaDE	2.05	4	6	5	8	8	8
MO_DE	1.47	6	7	6	3	3	3
MO_PSO	2.05	7	5	7	7	7	6
MTS	1.16	5	2	2	1	5	7

FES=500000

	Total	8. S_DTLZ2	9. R_DTLZ2	10. S_DTLZ3	11. WFG1	12. WFG8	13. WFG9
NSGA2_SBX	**0.47**	1	1	1	4	1	1
NSGA2_PCX	1.74	7	5	8	5	6	2
GDE3	0.84	4	3	2	1	2	4
DEMOwSA	1.58	2	8	4	7	4	5
MOSaDE	1.63	3	4	5	3	8	8
MO_DE	1.58	6	6	6	6	3	3
MO_PSO	2.16	8	7	7	8	5	6
MTS	1.37	5	2	3	2	7	7

M=5, Rank(Hypervolumn)

FES=5000

	Total	8. S_DTLZ2	9. R_DTLZ2	10. S_DTLZ3	11. WFG1	12. WFG8	13. WFG9
NSGA2_SBX	**0.74**	1	1	1	8	1	2
NSGA2_PCX	0.95	7	3	2	3	2	1
GDE3	1.16	2	5	4	5	3	3
DEMOwSA	1.68	8	8	6	1	4	5
MOSaDE	1.95	3	7	5	7	8	7
MO_DE	1.58	4	6	7	4	5	4
MO_PSO	1.74	5	2	8	6	6	6
MTS	1.58	6	4	3	2	7	8

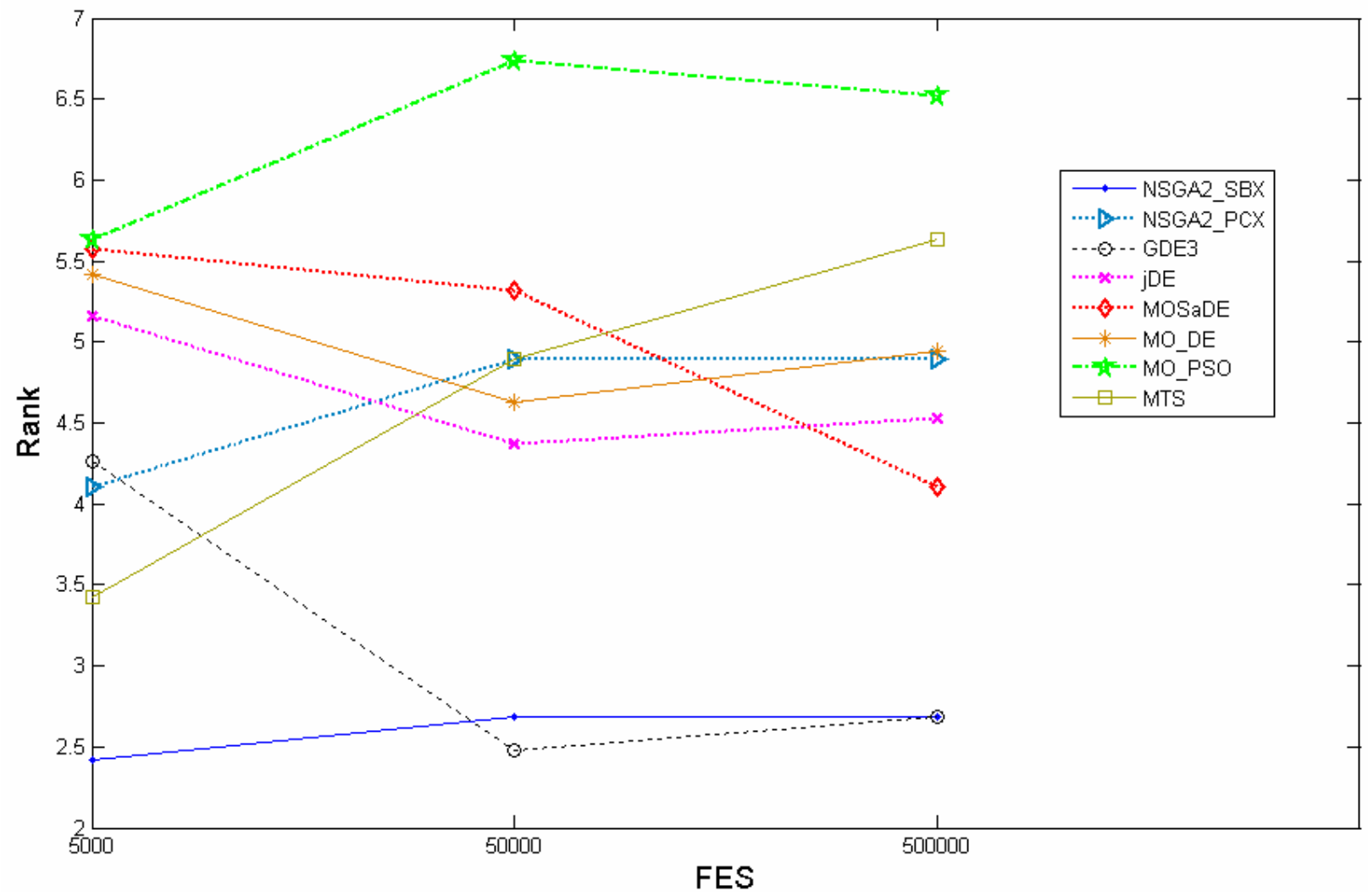
FES=50000

	Total	8. S_DTLZ2	9. R_DTLZ2	10. S_DTLZ3	11. WFG1	12. WFG8	13. WFG9
NSGA2_SBX	**0.63**	2	1	1	6	1	1
NSGA2_PCX	1.63	8	5	8	5	3	2
GDE3	0.79	3	2	2	3	2	3
DEMOwSA	1.47	4	8	4	4	4	4
MOSaDE	1.84	1	6	5	8	8	7
MO_DE	1.63	6	7	6	2	5	5
MO_PSO	1.84	5	4	7	7	6	6
MTS	1.53	7	3	3	1	7	8

FES=500000

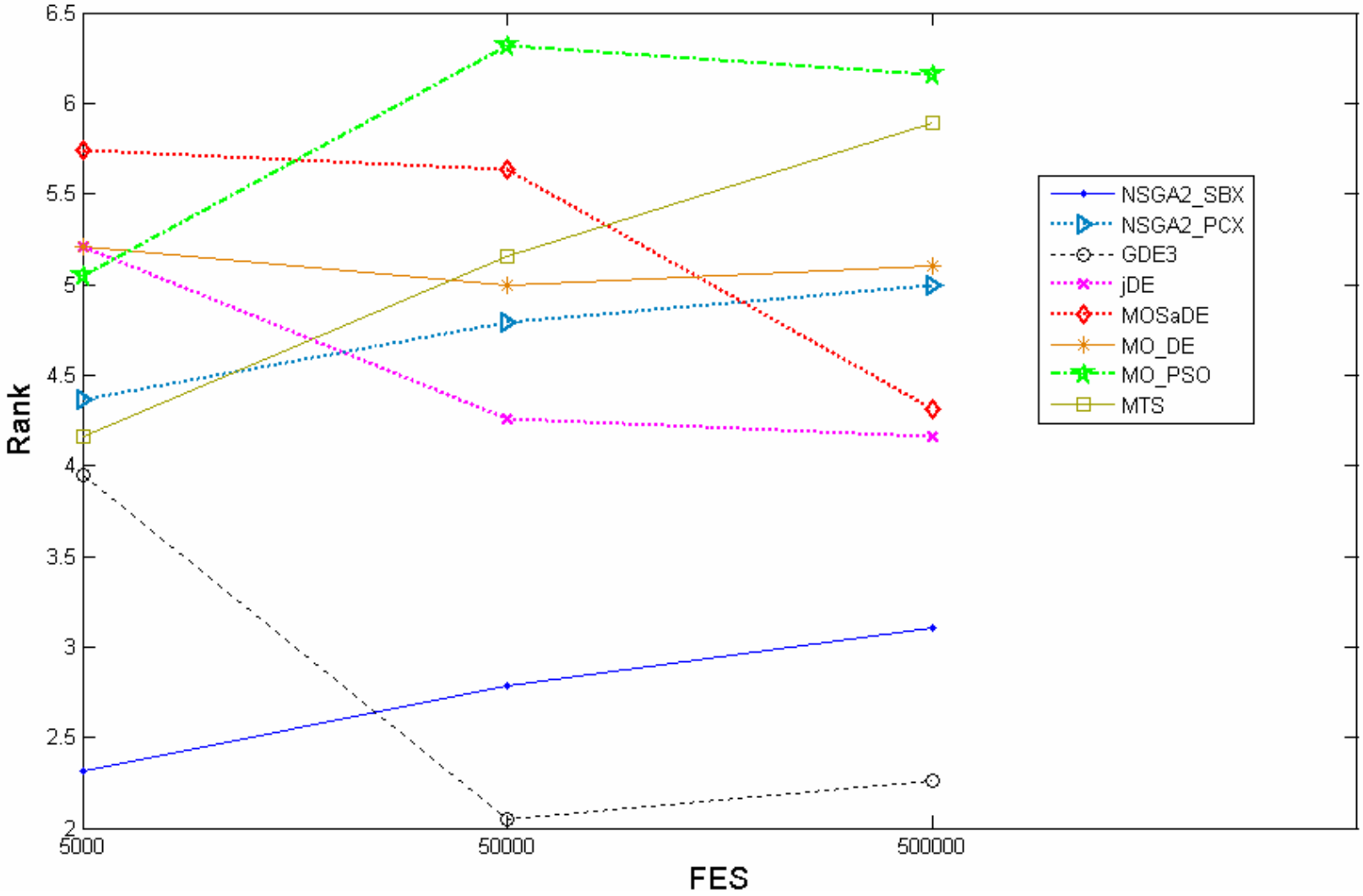
	Total	8. S_DTLZ2	9. R_DTLZ2	10. S_DTLZ3	11. WFG1	12. WFG8	13. WFG9
NSGA2_SBX	**0.53**	3	1	1	3	1	1
NSGA2_PCX	1.74	6	5	8	5	3	6
GDE3	0.74	4	2	2	1	2	3
DEMOwSA	1.37	2	8	3	7	4	2
MOSaDE	1.37	1	4	4	2	7	8
MO_DE	1.95	7	7	7	6	6	4
MO_PSO	1.84	5	6	6	8	5	5
MTS	1.84	8	3	5	4	8	7

CEC07 Summarized Results - Rank by I_{R2}

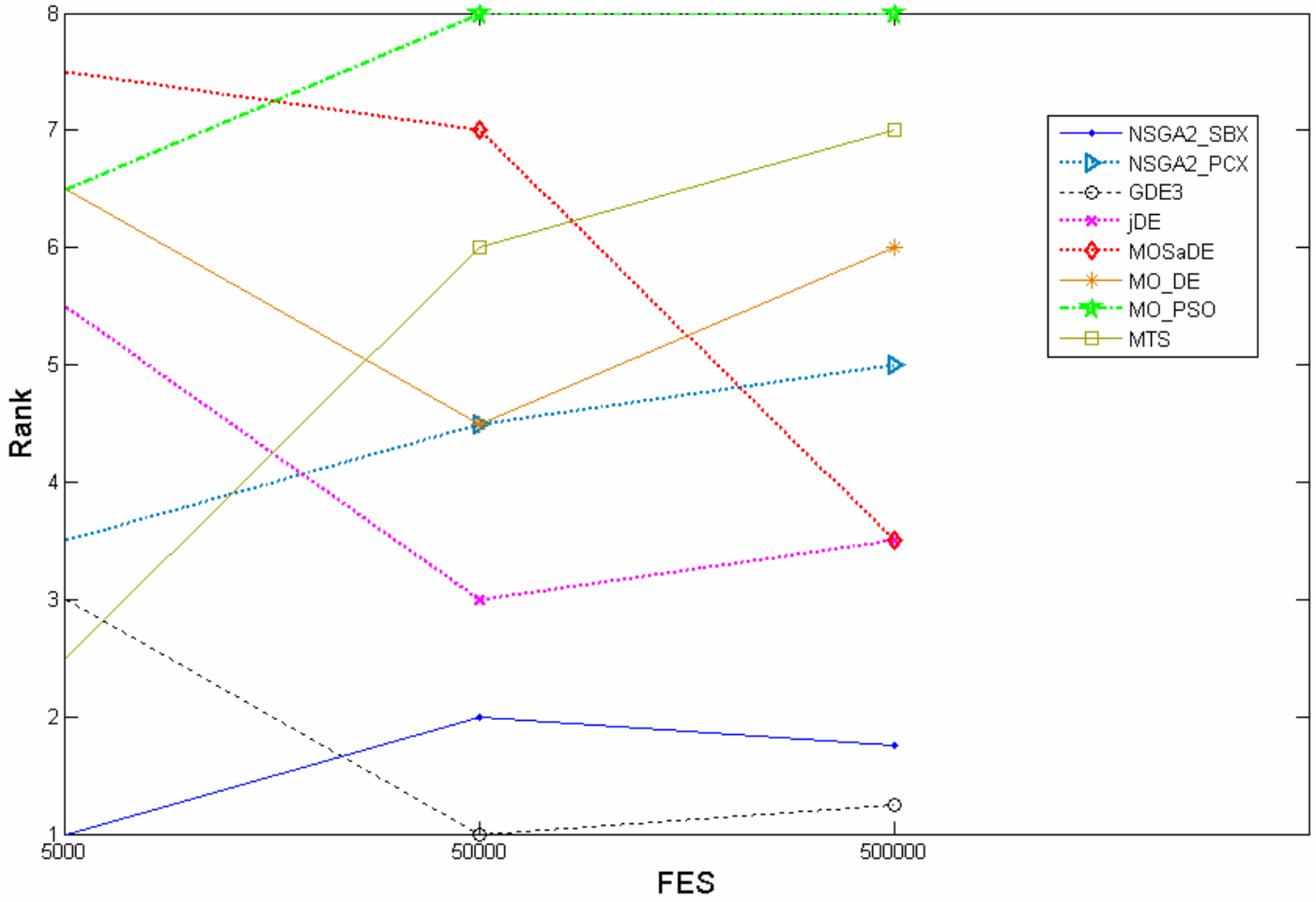


NANYANG TECHNOLOGICAL UNIVERSITY

CEC07 Summarized Results - Rank by $I_{\bar{H}}$



CEC07 Summarized Results - Rank by I_{R2} and $I_{\bar{H}}$



NANYANG TECHNOLOGICAL UNIVERSITY

Rank (I_{R2} and $I_{\bar{H}}$) on all test problems

FES=5000

	R	H	Total
NSGA2_SBX	2.42(1)	2.32(1)	**1**
NSGA2_PCX	4.11(3)	4.37(4)	3.5
GDE3	4.26(4)	3.95(2)	3.5
DEMOwSA	5.16(5)	5.21(6.5)	5.75
MOSaDE	5.58(7)	5.74(8)	7.5
MO_DE	5.42(6)	5.21(6.5)	6.25
MO_PSO	5.63(8)	5.05(5)	6.5
MTS	3.42(2)	4.16(3)	2.5

FES=50000

	R	H	Total
NSGA2_SBX	2.68(2)	2.79(2)	2
NSGA2_PCX	4.89(5.5)	4.79(4)	4.75
GDE3	2.47(1)	2.05(1)	**1**
DEMOwSA	4.37(3)	4.26(3)	3
MOSaDE	5.32(7)	5.63(7)	7
MO_DE	4.63(4)	5.00(5)	4.5
MO_PSO	6.74(8)	6.32(8)	8
MTS	4.89(5.5)	5.16(6)	5.75

FES=500000

	R	H	Total
NSGA2_SBX	2.68(1.5)	3.11(2)	1.75
NSGA2_PCX	4.89(5)	5.00(5)	5
GDE3	2.68(1.5)	2.26(1)	**1.25**
DEMOwSA	4.53(4)	4.16(3)	3.5
MOSaDE	4.11(3)	4.32(4)	3.5
MO_DE	4.95(6)	5.11(6)	6
MO_PSO	6.53(8)	6.16(8)	8
MTS	5.63(7)	5.89(7)	7

Acknowledgement

- Our research into evolutionary algorithms in Singapore is financially supported by an A*Star (Agency for Science, Technology and Research), Singapore

- Thanks to all past and current researchers working with me for their contributions. These slides show the results of their research efforts.

NANYANG TECHNOLOGICAL UNIVERSITY