Recent Advances in Real-Parameter Evolutionary Algorithms

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

IEEE CEC 2013

Cancun, Mexico

20 June 2013

Outline of the Presentation

- Benchmark Test Functions & CEC Benchmarking

- Search Methods Covered:
  - RP-EAs Modeled After the Binary GA
  - CMA-ES / Covariance Matrix Adapted Evolution Strategy
  - PSO / Particle Swarm Optimization
  - DE / Differential Evolution
  - EP / Evolutionary Programming

- Problem Scenarios Considered:
  - Single Objective (Bound Constrained, Dynamic, Constrained, Multimodal, Large Scale)
  - Multiobjective

Some Metaheuristics

- Local Search (LS)
- Ant Colonies Optimization (ACO)
- Artificial Immune Systems (AIS)
- Artificial Bee Colony (ABC)
- Cultural Algorithms (CA)
- Co-evolutionary Algorithms (CEA)
- Covariance Matrix Adaptation Evolution Strategy (CMA-ES)
- Differential Evolution (DE)
- Evolutionary Programming (EP)
- Firefly Algorithm

- Evolution Strategies (ES)
- Genetic Algorithms (GA)
- Guided Local Search (GLS)
- Genetic Programming (GP)
- Particle Swarm Optimization (PSO)
- Simulated Annealing (SA)
- Tabu Search (TS)
- Harmony Search
- Migratory Bird Search
- Gravitational Search
- Cuckoo search
- **The list is growing ….**

Our Public Domain Resources

Software Associated with Our Publications and several Surveys:

Resources available from
http://www.ntu.edu.sg/home/epnsugan

Ensemble Methods for Evolutionary Algorithms to tackle No-Free Lunch Theorem:

Resources available from
http://www.ntu.edu.sg/home/epnsugan/index_files/EEAs-EOAs.htm

Our Resources Useful to Many Researchers

Change photo

P. N. Suganthan Edit

Associate Professor, School of EEE, Nanyang Technological University, Singapore Edit

Computational Intelligence Edit

Verified email at ntu.edu.sg Edit

My profile is public Edit Link Homepage Edit

Citation indices

	All	Since 2008
Citations	8578	7463
h-index	40	38
i10-index	117	108

Citations to my articles

NANYANG TECHNOLOGICAL UNIVERSITY

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
  - Constrained problems
  - Single / Multi-objective problems
  - Static / Dynamic optimization problems
  - Multimodal problems (niching, crowding, etc.)

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 kinds of benchmark functions, different rotation matrices to compose a single test problem.

- Mix different properties of different basic test functions together to destroy repetitive structures and to construct novel **Composition Functions**

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:

J. J. Liang, P. N. Suganthan and K. Deb, "Novel composition test functions for numerical global optimization," *Proc. of IEEE International Swarm Intelligence Symposium*, pp. 68-75, 2005.

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:

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

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

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,

$bias_i$ define which optimum is global optimum.

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

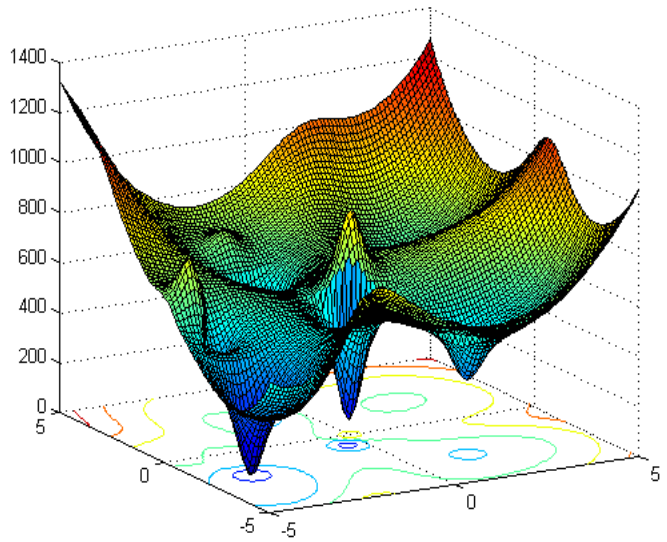
In order to get a better mixture, we estimate the biggest function value f_{max},

then normalize each basic function to similar height as below:

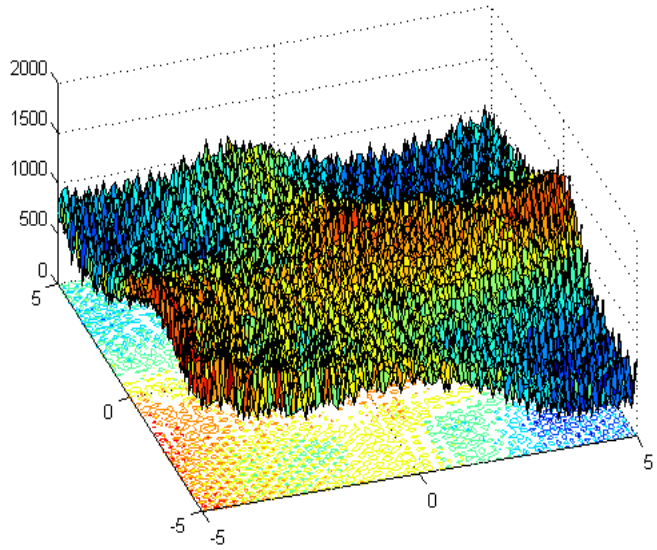
$f_i'(x) = C^* f_i(x) / |f_{maxi}|$, C is a predefined constant.

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

CEC Benchmarking Competitions

CEC'05 Special Session / Competition on Evolutionary RP single objective optimization.

CEC'06 Special Session / Competition on Evolutionary Constrained RP single objective optimization.

CEC'07 Special Session / Competition on Performance Assessment of RP MOEAs.

CEC'08 Special Session / Competition on large scale single objective global optimization with bound constraints.

CEC'09 Special Session / Competition on Dynamic Optimization.

CEC09 Special Session / Competition on Performance Assessment of RP MOEAs.

CEC10 Special Session / Competition on large-scale single objective global optimization with bound constraints.

CEC10 Special Session / Competition on Evolutionary Constrained RP single objective optimization.

CEC10 Special Session on Niching Introduces novel scalable test problems:
B. Y. Qu and P. N. Suganthan, "Novel Multimodal Problems and Differential Evolution with Ensemble of Restricted Tournament Selection", *IEEE CEC,* Barcelona, Spain, July 2010.

CEC11 Competition on Testing Evolutionary Algorithms on Real-world Numerical Optimization Problems.

CEC'13 Special Session / Competition on single objective optimization.

RP-EAs Modeled After Binary GA

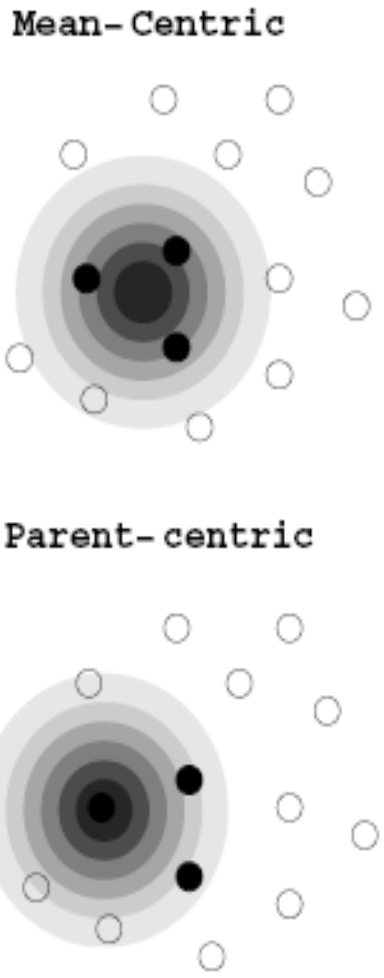
(Not as competitive as DE or PSO)

Major Steps in RP-GA

- Population initialization
- Fitness evaluation
- Selection of parents for crossover
- Crossover
- Mutation
- Replacement

Mean-Centric Vs Parent-Centric Recombination

- Individual recombination preserves the mean of parents (mean-centric) – *works well if the population is randomly initialized within the search range and the global optimum is near the centre of the search range.*
- Individual recombination biases offspring towards a particular parent, but with equal probability to each parent (parent-centric)



Mean-Centric Recombination

- SPX : Simplex crossover (Tsutsui et al. 1999)
- UNDX : Unimodal Normally distributed crossover (Ono et al. 1997, Kita 1998)
- BLX : Blend crossover (Eshelman et al 1993)
- Arithmetic crossover (Michalewicz et al 1991) (a variant is used in DE)

Parent-Centric Recombination

- SBX (Simulated Binary crossover) : variable-wise
- Vector-wise parent-centric recombination (PCX)

- And many more …

Minimal Generation Gap (MGG) Model

1. Select μ parents randomly
2. Generate λ offspring using a recombination scheme
3. Choose two parents at random from the population
4. One is replaced with the best of λ offspring and other with a solution by roulette-wheel selection from a combination of λ offspring and two parents

- Steady-state and elitist GA
- Typical values to use: population N = 300, λ=200,
- μ=3 (UNDX) and μ=n+1 (SPX) (*n*: dimensions)

- Satoh, H, Yamamura, M. and Kobayashi, S.: Minimal generation gap model for GAs considering both exploration and exploitation. *Proc. of the IIZUKA* 96, pp. 494-497 (1997).

Generalized Generation Gap (G3) Model

1. Select the best parent and μ-1 other parents randomly
2. Generate λ offspring using a recombination scheme
3. Choose two parents at random from the population
4. Form a combination of two parents and λ offspring, choose best two solutions and replace the chosen two parents

- Parameters to tune: λ, μ and N

Deb, K., Anand, A. and Joshi, D. (2002). A computationally efficient evolutionary algorithm for real-parameter optimization. *Evolutionary Computation Journal* 10(4), 371–395.

Mutation

- In RP-EAs, the main difference between crossover and mutation may be the number of parents used for the operation. In mutation, just 1. In crossover, 2 or more (excluding DE's terminology as DE mutation can involve up to 5 parents).
- There are several mutation operators developed for real parameters
  - Random mutation : the same as re-initialization of a variable
  - Gaussian mutation: $X_{new}=X_{old}+N(0, \sigma)$
  - Cauchy mutation with EP (Xin Yao *et. al* 1999)
  - Levy mutation / adaptive Levy mutation with EP (Lee, Xin Yao, 2004)
- By making use of these RP-crossover (recombination), RP-mutation operators, RP-EAs can be developed in a manner similar to the Binary GA.

Covariance Matrix Adapted Evolution Strategy CMAES

By Nikolaus Hansen & collaborators

Extensive Tutorials / Codes available from
http://www.bionik.tu-berlin.de/user/niko/cmaesintro.html
http://www.bionik.tu-berlin.de/user/niko/cmaes_inmatlab.html

It **was** the best for single objective bound constrained optimization in 2005.

For other scenarios, not at all competitive

Basic Evolution Strategy [BFM-02]

- Randomly generate an initial population of M solutions. Compute the fitness values of these M solutions.
- Use all vectors as parents to create n_b offspring vectors by Gaussian mutation.
- Calculate the fitness of n_b vectors, and prune the population to M fittest vectors.
- Go to the next step if the stopping condition is satisfied. Otherwise, go to Step 2.
- Choose the fittest one in the population in the last generation as the optimal solution.

Covariance Matrix Adapted Evolution Strategy CMAES

- The basic ES generates solutions around the parents
- CMAES keeps track of the evolution path.
- It updates mean location of the population, covariance matrix and the standard deviation of the sampling normal distribution after every generation.
- The solutions are ranked which is not a straight forward step in multi-objective / constrained problems.
- Ranked solutions contribute according to their ranks to the updating of the mean position, covariance matrix, and the step-size parameter.
- Covariance analysis reveals the interactions / couplings among the parameters.

CMAES Iterative Equations [PPSN 2006 tutorials]

Summary of Equations

The Covariance Matrix Adaptation Evolution Strategy

Initialize $m \in \mathbb{R}^n$, $\sigma \in \mathbb{R}_+$, $\mathbf{C} = \mathbf{I}$, and $p_\mathrm{c} = \mathbf{0}$, $p_\sigma = \mathbf{0}$,
set $c_\mathrm{c} \approx 4/n$, $c_\sigma \approx 4/n$, $c_\mathrm{cov} \approx \mu_\mathrm{eff}/n^2$, $\mu_\mathrm{cov} = \mu_\mathrm{eff}$, $d_\sigma \approx 1 + \sqrt{\frac{\mu_\mathrm{eff}}{n}}$,
λ, and $w_i, i = 1, \ldots, \mu$ such that $\mu_\mathrm{eff} \approx 0.3\,\lambda$, where $\mu_\mathrm{eff} = \frac{1}{\sum_{i=1}^{\mu} w_i^2}$

While not terminate

$$\boldsymbol{x}_i = m + \sigma z_i, \quad z_i \sim \mathcal{N}_i(\mathbf{0}, \mathbf{C}), \qquad \text{sampling}$$

$$m \leftarrow m + \sigma \langle z \rangle_\mathrm{sel} \quad \text{where } \langle z \rangle_\mathrm{sel} = \textstyle\sum_{i=1}^{\mu} w_i z_{i:\lambda} \qquad \text{update mean}$$

$$p_\mathrm{c} \leftarrow (1 - c_\mathrm{c}) p_\mathrm{c} + \mathbf{1}_{\{\|p_\sigma\| < 1.5\sqrt{n}\}} \sqrt{1 - (1 - c_\mathrm{c})^2} \sqrt{\mu_\mathrm{eff}}\, \langle z \rangle_\mathrm{sel} \qquad \text{cumulation for } \mathbf{C}$$

$$\mathbf{C} \leftarrow (1 - c_\mathrm{cov})\, \mathbf{C} + c_\mathrm{cov} \tfrac{1}{\mu_\mathrm{cov}} p_\mathrm{c} p_\mathrm{c}^\mathrm{T} + c_\mathrm{cov} \left(1 - \tfrac{1}{\mu_\mathrm{cov}}\right) \mathbf{Z} \qquad \text{update } \mathbf{C}$$

$$\text{where } \mathbf{Z} = \textstyle\sum_{i=1}^{\mu} w_i z_{i:\lambda} z_{i:\lambda}^\mathrm{T}$$

$$p_\sigma \leftarrow (1 - c_\sigma) p_\sigma + \sqrt{1 - (1 - c_\sigma)^2} \sqrt{\mu_\mathrm{eff}}\, \mathbf{C}^{-\frac{1}{2}} \langle z \rangle_\mathrm{sel} \qquad \text{cumulation for } \sigma$$

$$\sigma \leftarrow \sigma \times \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\|p_\sigma\|}{\mathrm{E}\|\mathcal{N}(\mathbf{0},\mathbf{I})\|} - 1\right)\right) \qquad \text{update of } \sigma$$

CMAES

- Performs well on rotated / linked problems with bound constraints.
- It learns global linkages between parameters.
- The cumulation operation integrates a degree of forgetting.
- The CMAES implementation is self-adaptive.
- CMAES is not highly effective in solving constrained problems, multi-objective problems, etc.
- CMAES is not as simple as PSO / DE.

N. Hansen (2006), 'Covariance Matrix Adaptation (CMA) Evolution Strategy,' Tutorial at PPSN-06, Reykjavik, Iceland, http://www.bionik.tu-berlin.de/user/niko/cmaesintro.html

N. Hansen, A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies", *Evolutionary Computation*, 9(2), pp. 159-195, 2004.

PSO

- PSO
- PSO variants
- CLPSO
- OLPSO
- DMS-L-PSO

Kennedy, J. and Eberhart, R. C. (1995), "Particle swarm optimization," *Proc. of IEEE International Conference on Neural Networks,* Piscataway, NJ, pp. 1942-1948.

Particle Swarm Optimizer

- Introduced by Kennedy and Eberhart in 1995
- Emulates flocking behavior of birds to solve optimization problems
- Each solution in the landscape is a particle
- All particles have fitness values and velocities

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.**

Particle Swarm Optimizer

- where c_1 and c_2 in the equation are the acceleration constants $rand1_i^d$ and $rand2_i^d$ are two random numbers in the range [0,1];
- $\mathbf{X}_i = (X_i^1, X_i^2, ..., X_i^D)$ represents the position of the *ith* particle;
- $\mathbf{V}_i = (V_i^1, V_i^2, ..., V_i^D)$ represents the rate of the position change (velocity) for particle *i.*
- $\mathbf{pbest}_i = (pbest_i^1, pbest_i^2, ..., pbest_i^D)$ represents the best previous position (the position giving the best fitness value) of the *ith* particle;
- $\mathbf{gbest} = (gbest^1, gbest^2, ..., gbest^D)$ represents the best previous position of the population;
- $\mathbf{lbest}_k = (lbest_k^1, lbest_k^2, ..., lbest_k^D)$ represents the best previous position achieved by the neighborhood of the 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)
  - … …
- 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.
  - 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 + cooperative approach (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)

Comprehensive Learning PSO

- **Comprehensive Learning Strategy** :

$$V_i^d \leftarrow w * V_i^d + c * rand_i^d * (pbest_{fi(d)}^d - X_i^d)$$

- where $\boldsymbol{f_i} = [f_i(1), f_i(2), ..., f_i(D)]$ defines which particle's **pbest** particle *i* should follow. $pbest_{fi(d)}^d$ can be the corresponding dimension of any particle's **pbest** including its own **pbest**, and the decision depends on probability *Pc*, referred to as the learning probability, which can take different values for different particles.

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

Comprehensive Learning PSO

- The tournament selection (2) procedure is employed to choose the exemplar for each dimension of each particle.

- To ensure that a particle learns from good exemplars and to minimize the time wasted on poor directions, we allow the particle to learn from the exemplars until the particle ceases improving for a certain number of generations called the refreshing gap *m* (7 generations), after which we re-assign the dimensions of this particle.

Comprehensive Learning PSO

- We observe three main differences between the CLPSO and the original PSO:
  - Instead of using particle's own **pbest** and **gbest/lbest** as the exemplars, all particles' **pbest**s can potentially be used as the exemplars to guide a particle's flying direction.
  - Instead of learning from the same exemplar particle for all dimensions, each dimension of a particle in general can learn from different **pbest**s for different dimensions for some generations. In other words, in any one iteration, each dimension of a particle may learn from the corresponding dimension of different particle's **pbest**.
  - Instead of learning from two exemplars (**gbest/lbest** and **pbest**) at the same time in every generation as in the original PSO, each dimension of a particle learns from just one exemplar for some generations.
  - All these dimensions together represent a point in the sarch space and may be called as the exemplar vector or guidance vector.

Comprehensive Learning PSO

- Using two exemplars (**pbest & lbest/gbest**) is not effective as a particle is directed between two good positions! There is no certainty that the in-between region is good.
- As each dimension a select **pbest** position possibly from a different particle's **pbest**, there are a very large number of potential exemplar positions thereby increasing the diversity substantially.
- Experiments have been conducted on16 test functions which can be grouped into four classes. The results demonstrated good performance of the CLPSO in solving multimodal problems when compared with eight other recent variants of the PSO.

Liang, J. J., Suganthan, P. N., Qin, A. K. and Baskar, S (2006). "Comprehensive Learning Particle Swarm Optimizer for Global Optimization of Multimodal Functions," *IEEE Transactions on Evolutionary Computation*, June 2006. Matlab codes are also available

Orthogonal Learning PSO (OLPSO)

- In OLPSO, guidance vector (P_o) is used to update the particle's flying velocity v_{id}.

$$v_{id} = wv_{id} + cr_d(p_{od} - x_{id})$$
$$x_{id} = x_{id} + v_{id}$$

- Guidance vector (P_o) is constructed by orthogonal learning (OL) strategy that combines the information of personal best position (P_i) and neighborhood's best position (P_n).

$$P_o = P_i \oplus P_n$$

where, the symbol $\oplus$ for the orthogonal experiment design (OED) operation.

Zhi-Hui Zhan, Jun Zhang ; Yun Li Yu-Hui Shi, **Orthogonal Learning Particle Swarm Optimization,** IEEE Transactions on Evolutionary Computation, **Volume:** 15 , Issue: 6 **pp** 832 – 847, Dec. 2011

Orthogonal Learning (OL) Strategy

- In OL strategy, OED method is used to test the combinations of (P_i) and (P_n) for constructing a guidance vector (P_o).

- OED operation :

1) Orthogonal Array with Q Levels per factor : $L_M(Q^N)$

where, L denotes orthogonal array, M is number of combination of test solutions, $Q = 2\ levles\ (P_i\ and\ P_n),\ \ N = D$

2) Factor Analysis to find the best combination of levels

$$S_{nq} = \frac{\sum_{m=1}^{M} f_m \times z_{mnq}}{\sum_{m=1}^{M} z_{mnq}}$$

where, S_{nq} denotes the effect of the qth level in the nth factor , f_m denotes the experimental results of the mth combination, $z_{mnq} = 1$

NANYANG TECHNOLOGICAL UNIVERSITY

Constructing Exemplar Position Using OL

Step 1: An orthogonal array (OA) is generated as $L_M(2^D)$where $M = 2^{\lceil log_2 (D+1) \rceil}$.

Step 2: Make up M tested solutions by selecting the corresponding value from P_i or P_n according to the OA.

Step 3: Evaluate each tested solution and record the best (with best fitness) solution, X_b.

Step 4: Calculate the effect of each level on each factor and determine the best level for each factor .

Step 5: Derive a predictive solution with the levels determined in Step 4 and evaluate X_p.

Step 6: Compare $f(X_b)$ and $f(X_p)$ and the level combination of the better solution is used to construct the vector . P_o .

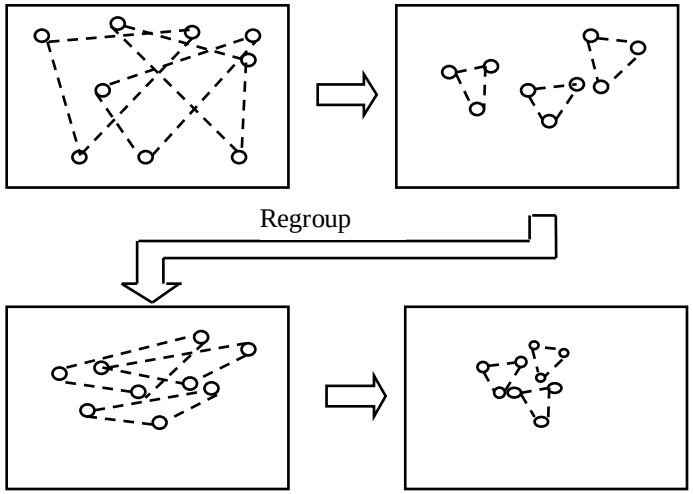
Dynamic Multi-Swarm PSO

- Constructed based on the local version of PSO with a novel neighborhood topology.

- This new neighborhood structure has two important characteristics:
  - Small Sized Swarms
  - Randomized Regrouping Schedule

J. J. Liang, and P. N. Suganthan, "Dynamic Multi-Swarm Particle Swarm Optimizer," *IEEE Swarm Intelligence Symposium*, pp. 124-129, Pasadena, CA, USA, June 2005.

Dynamic Multi-Swarm PSO



- The population is divided into small swarms randomly.
- These sub-swarms use their own particles to search for better solutions and converge to good solutions.
- The whole population is regrouped into new sub-swarms periodically. New sub-swarms continue the search.
- This process is continued until a stop criterion is satisfied

Dynamic Multi-Swarm PSO

Adaptive Self-Learning Strategy

Each Particle has a corresponding Pc_i. Every R generations, $keep\_id_i$ is decided by Pc_i. If the generated random number is larger than or equal to Pc_i, this dimension will be set at the value of its own *pbest*, $keep\_id_i(d)$ is set to1. Otherwise $keep\_id_i(d)$ is set to 0 and it will learn from the **lbest** and its own **pbest** as the PSO with constriction coefficient:

If $keep\_id_i^d = 0$

$$V_i^d \leftarrow 0.729 * V_i^d + 1.49445 * rand1_i^d * (pbest_i^d - X_i^d) + 1.49445 * rand2_i^d * (lbest_k^d - X_i^d)$$

$$V_i^d = \min(V_{\max}^d, \max(-V_{\max}^d, V_i^d))$$

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

Otherwise

$$X_i^d \leftarrow pbest_i^d$$

Somewhat similar to DE

Dynamic Multi-Swarm PSO

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.
- When the sub-swarms are regrouped, the mean of normal distribution of **Pc** is recalculated according to all the recorded **Pc** values corresponding to successful movements during this period. The record of the successful **Pc** values will be cleared once the normal distribution's mean is recalculated.
- As a result, the proper **Pc** value range for the current problem can be learned to suit the particular problem.

DMS-PSO with Local Search

- Since we achieve a larger diversity using DMS-PSO, at the same time, the convergence rate is reduced. In order to alleviate this problem and to enhance local search capabilities, a local search is incorporated:
  - Every L generations, the **pbest**s of five randomly chosen particles will be used as the starting points and the Quasi-Newton method (fminunc(…,…,…)) is employed to do the local search with the L_FEs as the maximum FEs.
  - In the end of the search, all particles form a single swarm to become a global PSO version. The best solution achieved so far is refined using Quasi-Newton method every *L* generations with the 5*L_FEs as the maximum FEs.
  - If local search results in improvements, the nearest **pbest** is replaced / updated.

Differential Evolution

Possibly the best real-parameter Optimizer overall according to CEC competitions

S. Das and P. N. Suganthan, "Differential Evolution: A Survey of the State-of-the-art", *IEEE Trans. on Evolutionary Computation,* Vol. 15, No. 1, pp. 4-31, Feb. 2011

DE Special issue in *IEEE Trans. on Evolutionary Computation,* in Feb 2011.

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 satisfactorily the Chebychev problem.

- 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.

- 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
  - Easy parallelization
  - Floating-point encoding
- **Basic components**
  - Initialization
  - Trial vector generation
    - Mutation
    - Recombination
  - Replacement

Insight into classic 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 classic 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:

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

Replacement

If the trial vector $\mathbf{u}_{i,g}$ has the 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 classic DE in 2D

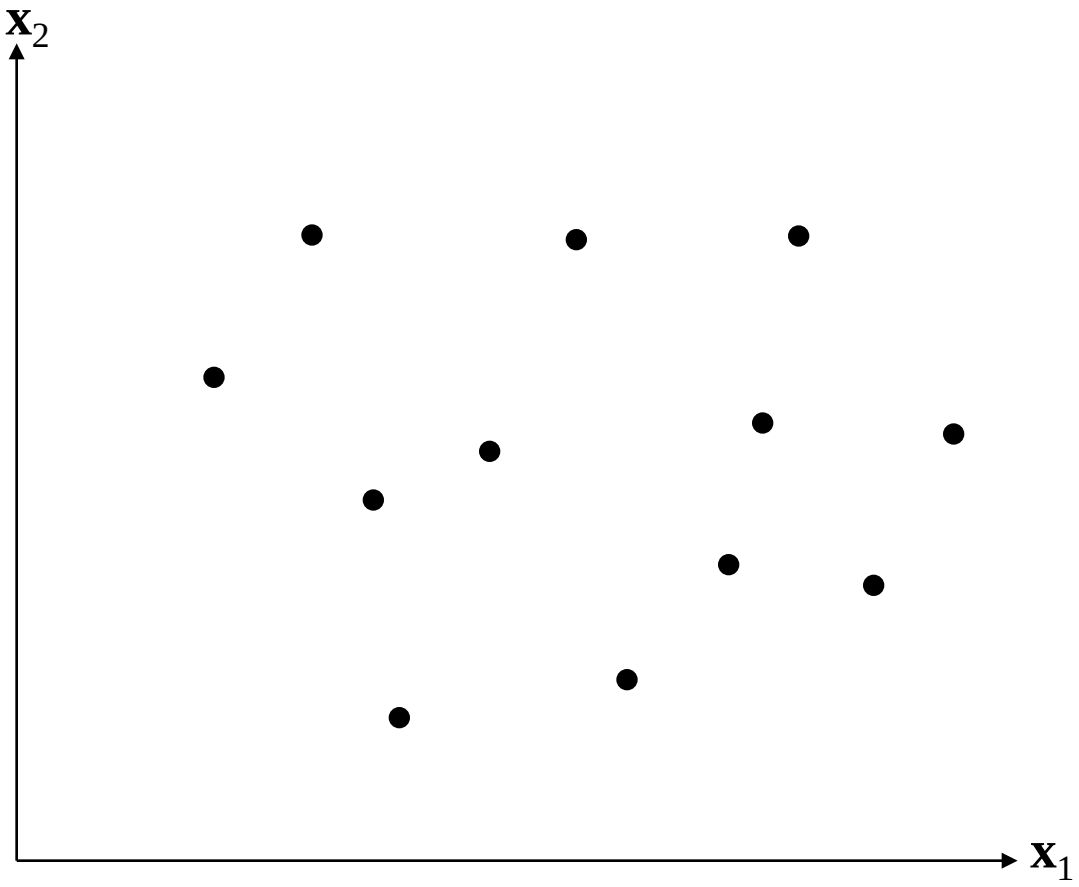


Illustration of classic DE

Illustration of classic DE in 2D

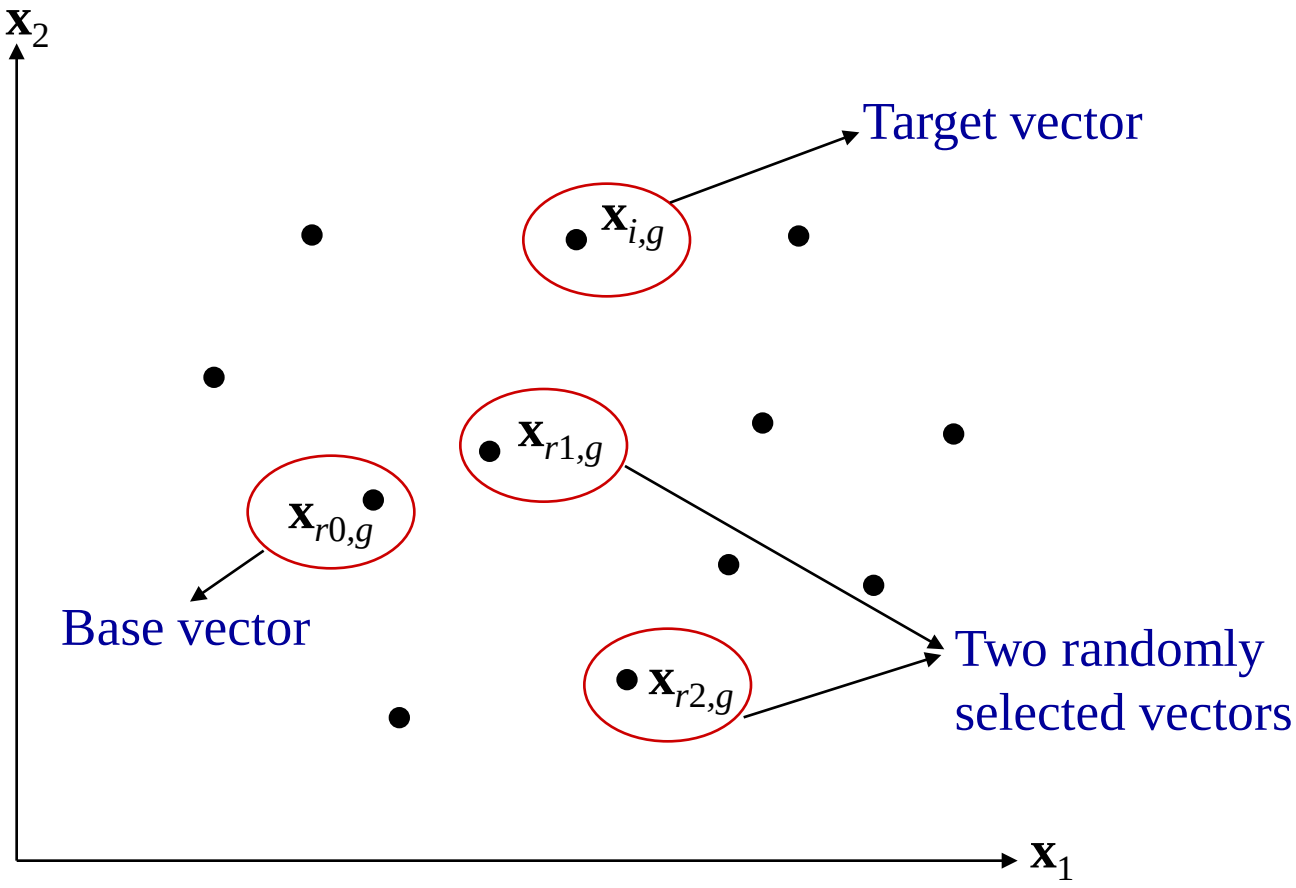
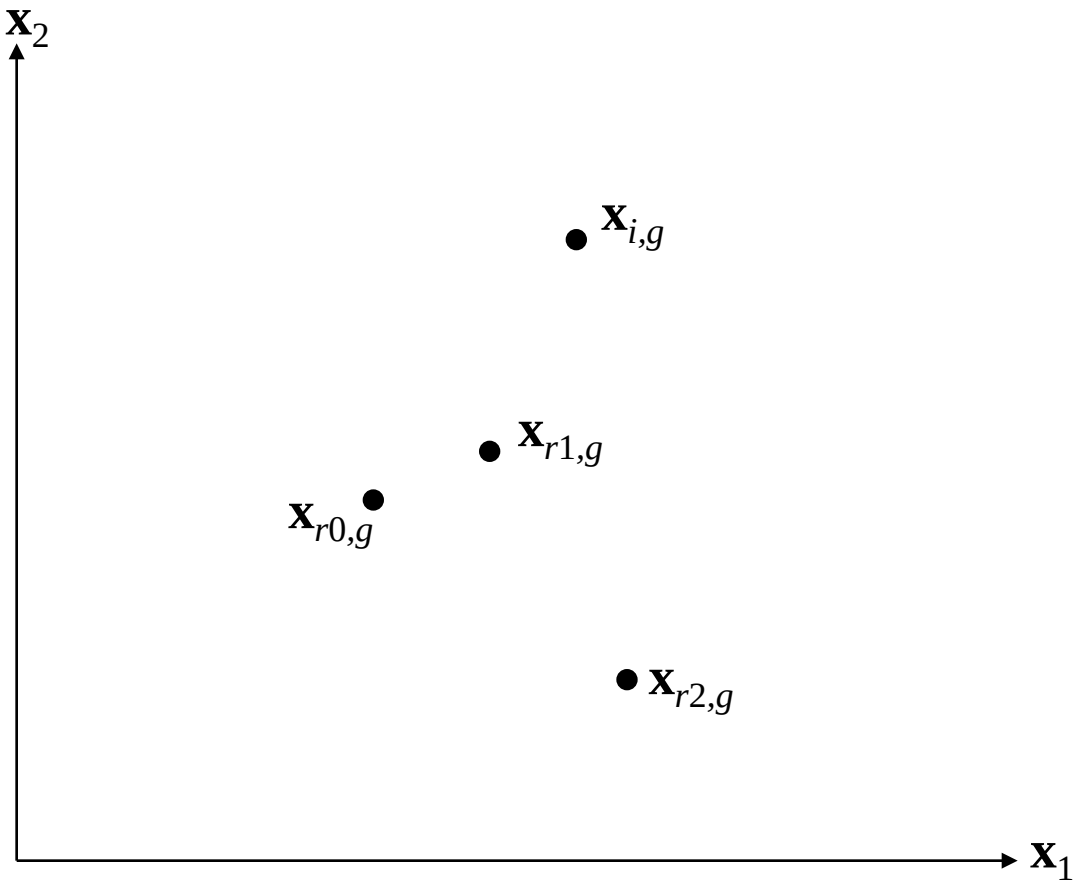


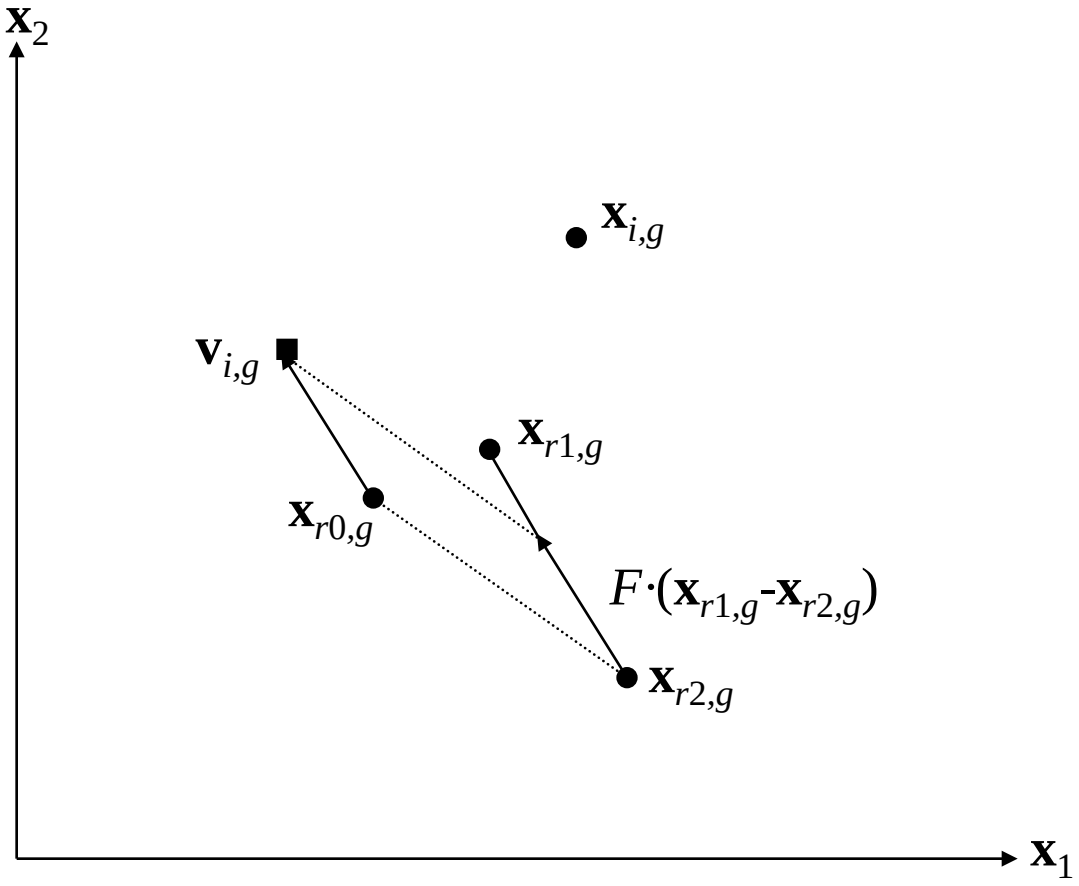
Illustration of classic DE

Illustration of classic DE in 2D



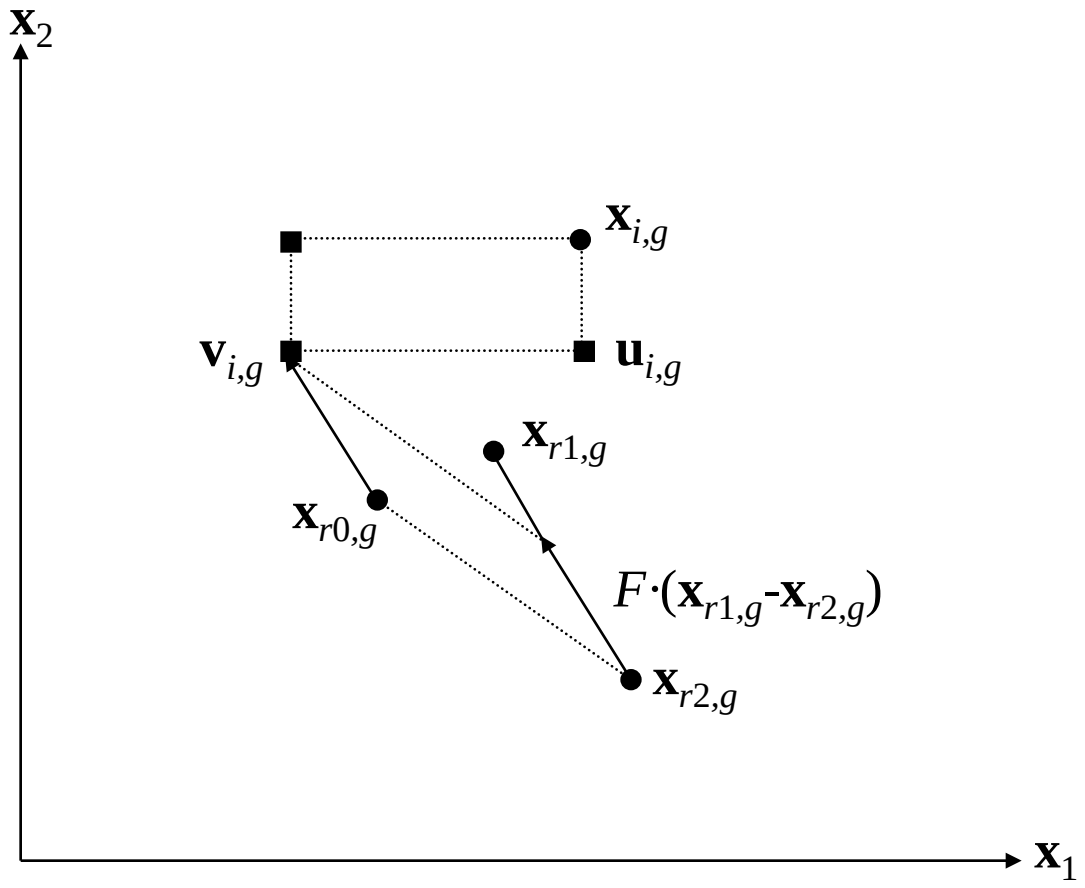
Four operating vectors in 2D continuous space

Illustration of classic DE in 2D



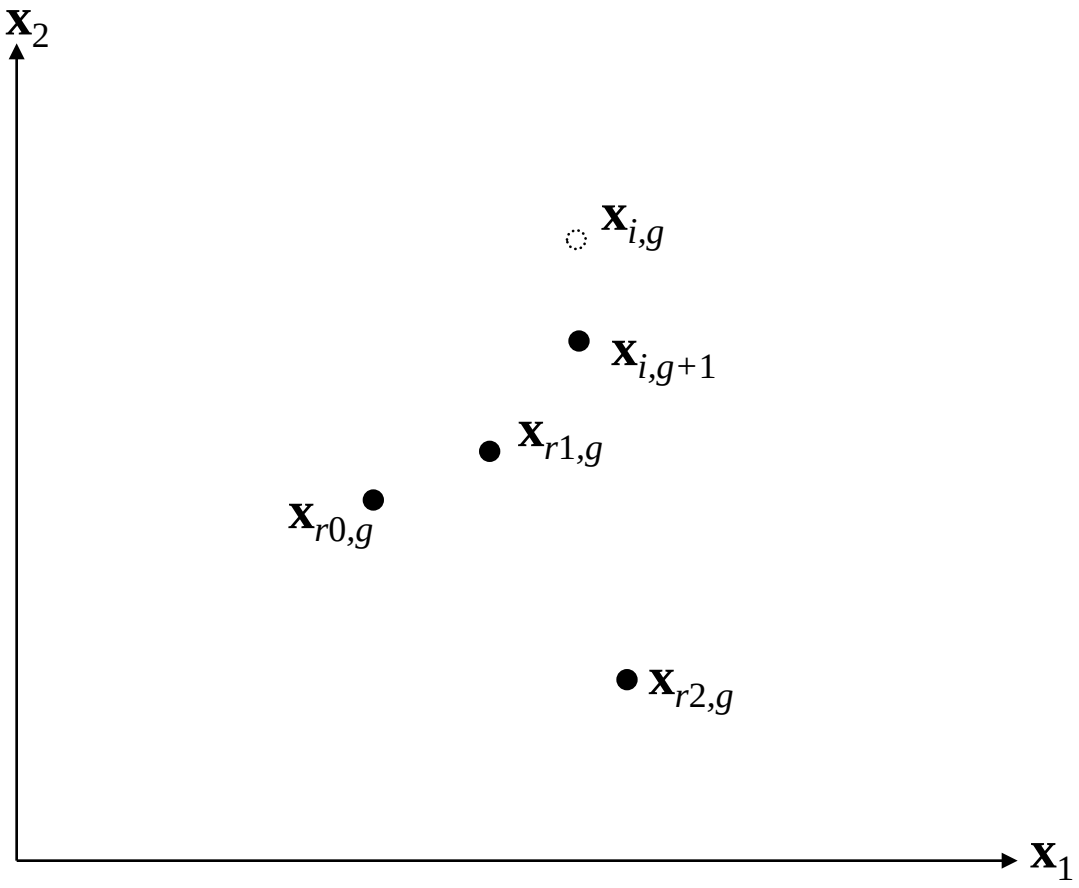
Mutation

Illustration of classic DE in 2D



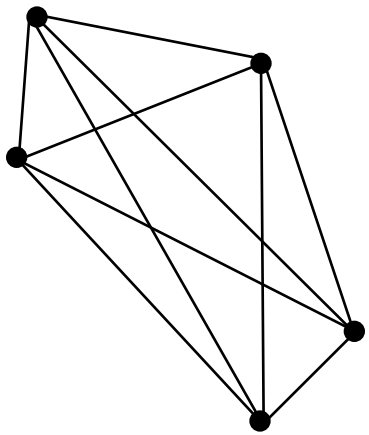
Crossover

Illustration of classic DE in 2D

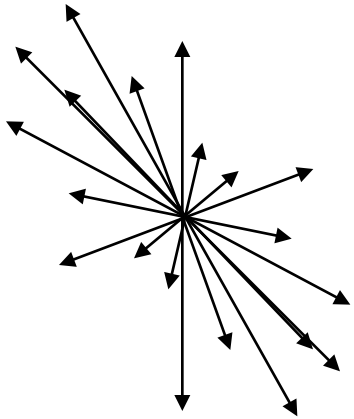


Replacement

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

- 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 and better

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* (rotation variant) and *dither* (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: $$\mathbf{V}_{i,G} = \mathbf{X}_{best,G} + F \cdot \left(\mathbf{X}_{r_1,G} - \mathbf{X}_{r_2,G}\right)$$

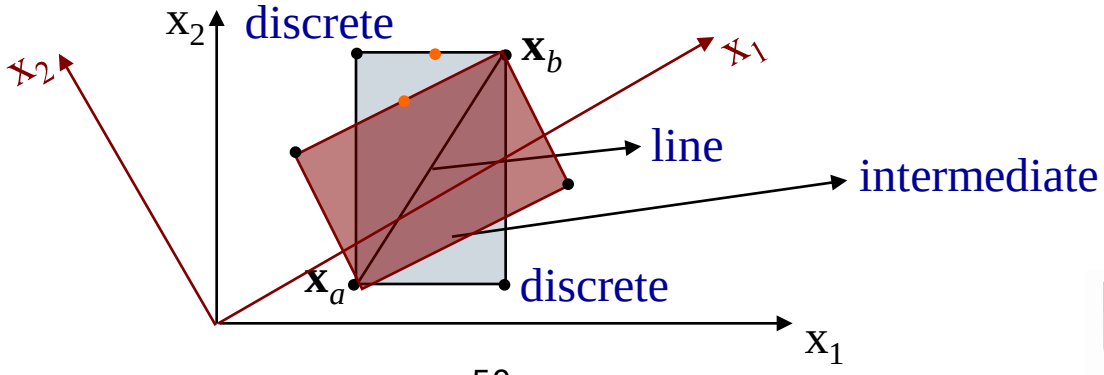
DE/current-to-best/1: $$\mathbf{V}_{i,G} = \mathbf{X}_{i,G} + F \cdot \left(\mathbf{X}_{best,G} - \mathbf{X}_{i,G}\right) + F \cdot \left(\mathbf{X}_{r_1,G} - \mathbf{X}_{r_2,G}\right)$$

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
    - Binominal (uniform)
  - Arithmetic recombination
    - Line recombination (rotation invariant)
    - Intermediate recombination (rotation variant)
    - Extended intermediate recombination (rotation variant)



DE variants

- Crossover rate $CR \in [0,1]$

  Decomposable (small CR) and indecomposable functions (large CR)

- Degenerate cases in the trial vector generation

For example, in classic DE, $r_1=r_2$, $r_0=r_1$, $r_0=r_2$, $i=r_0$, $i=r_1$, $i=r_2$

Better to generate mutually exclusive indices for target vector, base vector and vectors constituting the difference vector

Replacement

- One-to-one replacement

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 favor 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-adaptive DE (2) [BGBMZ06]: encode mutation scale factor and crossover rate in each individual, which are reinitialized according to two new probability variables.

Self-adaptive DE (SaDE)

DE with strategy and parameter self-adaptation:

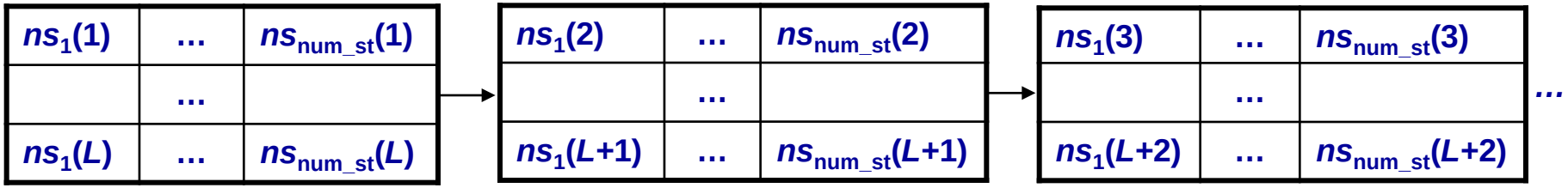
Strategy adaptation: select one strategy from a pool of candidate strategies with the probability proportional to its previous successful rate to generate effective trial vectors during a certain learning period

Steps:

- Initialize selection probability $p_i=1/num\_st$, $i=1,\ldots,num\_st$ for each strategy
- According to the current probabilities, we employ ***stochastic universal selection*** to assign one strategy to each target vector in the current population
- 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)"
- 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

K. Qin, V. L. Huang, and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization", *IEEE Trans. On Evolutionary Computations*, pp. 398-417, April, 2009

Self-adaptive DE (SaDE)



- 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

Ensemble of Parameters and Mutation and Crossover Strategies in DE (EPSDE)

- Motivation
  - Empirical guidelines are given to select parameters (tuning problem)
  - Fixed single mutation strategy & parameters – may not be the best always over the whole evolution.
  - Variations in optimization problems (Ex: uni-modal & multimodal)
  - To introduce Adaptation

- Implementation
  - Contains a pool of mutation strategies & parameter values
  - They compete to produce successful offspring population.
  - Candidate pools must be restrictive to avoid unfavorable influences such as premature convergence
  - The pools should be diverse

R. Mallipeddi, P. N. Suganthan, Q. K. Pan and M. F. Tasgetiren, "Differential Evolution Algorithm with ensemble of parameters and mutation strategies," *Applied Soft Computing*, 11(2):1679–1696, March 2011.

EPSDE

- Selection of pool of mutation strategies
  1. strategies without crossover (DE/current-to-rand/1/bin)
  2. strategies with crossover
     1. individuals of mutant vector randomly selected (DE/rand/1/bin)
     2. rely on the best found so far (DE/best/2/bin)
- Selection of pool of parameters

  F = {0.4, 0.5,0.6,0.7,0.8,0.9} CR = {0.1,0.2,0.3,0.4, 0.5,0.6,0.7,0.8,0.9}
- Initial population randomly assigned with a mutation strategy & parameters
- Trial vector better than target vector - retain setting
- Trial vector worse than target vector - re-initialize setting
- Increased probability of offspring production by better combination

EPSDE

- 14 problems – (10D, 30D and 50D)
- 30 runs (100000, 300000 and 500000FEs for 10D, 30D and 50D)
- *NP* = 50 (all algorithms)

EPSDE	*D*	SaDE	jDE	ADE	SDE	JADE	Total
inferior	10	1	1	0	0	1	3
	30	1	1	0	1	1	4
	50	1	1	0	1	2	5
equal	10	12	9	9	6	9	45
	30	6	6	8	0	6	26
	50	3	4	9	1	5	22
better	10	1	4	5	8	4	**22**
	30	7	7	6	13	7	**40**
	50	10	9	5	12	7	**43**

- Scalability

EPSDE -SEMCCO 2010

- Crossover: Binomial and Exponential
- DE/rand/1/bin replaced with JADE mutation
- Test Problems: 25 benchmark problems of CEC 2005 (10D & 30D)
- Runs : 25
- Statistical *t*-test
- Comparative Results with JADE (Jingqiao's & Arthur's DE, i.e. JADE):
  - EPSDE better, similar and worst in 13, 8 and 4 in 10D
  - EPSDE better, similar and worst in 13, 9 and 3 in 30D

R. Mallipeddi and P. N. Suganthan, "Differential Evolution Algorithm with Ensemble of Parameters and Mutation and Crossover Strategies", Swarm Evolutionary and Memetic Computing Conference, pp. 71-78, LNCS, Vol. 6466, Chennai, India 2010.

- Most Recent variant includes Adaptation, i.e. SA-EPSDE.

JADE (Zhang and Sanderson, TEC, 2009)

1) Uses DE/current-to-*p*best strategy as a less greedy generalization of the DE/current-to-best/ strategy. Instead of only adopting the best individual in the DE/current-to-best/1 strategy, the current-to-pbest/1 strategy utilizes the information of other good solutions.

Denoting $\vec{X}^{\,p}_{best,G}$ as a randomly chosen vector from the top 100*p*% individuals of the current population,

DE/current-to-*p*best/1 without external archive: $\vec{V}_{i,G} = \overset{\sqcup}{X}_{i,G} + F_i \cdot (\vec{X}^{\,p}_{best,G} - \overset{\sqcup}{X}_{i,G}) + F_i \cdot (\vec{X}_{r_1^i,G} - \overset{\sqcup}{X}_{r_2^i,G})$

2) JADE can optionally make use of an external archive (A), which stores the recently explored inferior solutions. In case of DE/current-to-pbest/1 with archive, $\vec{X}_{i,G}$, $\vec{X}^{\,p}_{best,G}$, and $\vec{X}_{r_1^i,G}$ are selected from the current population P, but $\overset{\sqcup}{X}_{r_2^i,G}$ is selected from $P \cup A$

JADE (Contd..)

3) JADE adapts the control parameters of DE in the following manner:

A) *Cr* for each individual and at each generation is randomly generated from a normal distribution $N(\mu_{Cr},0.1)$ and then truncated to [0, 1].

The mean of normal distribution is updated as: $\mu_{Cr} = (1-c).\mu_{Cr} + c.mean_A(S_{Cr})$

where S_{Cr} be the set of all successful crossover probabilities Cr_i s at generation *G*

B) Similarly for each individual and at each generation F_i is randomly generated from a Cauchy distribution $C(\mu_F,0.1)$ with location parameter μ_F and scale parameter 0.1.

F_i is truncated if F_i > 1 or regenerated if F_i <= 0

The location parameter of the Cauchy distribution is updated as: $\mu_F = (1-c).\mu_F + c.mean_L(S_F)$

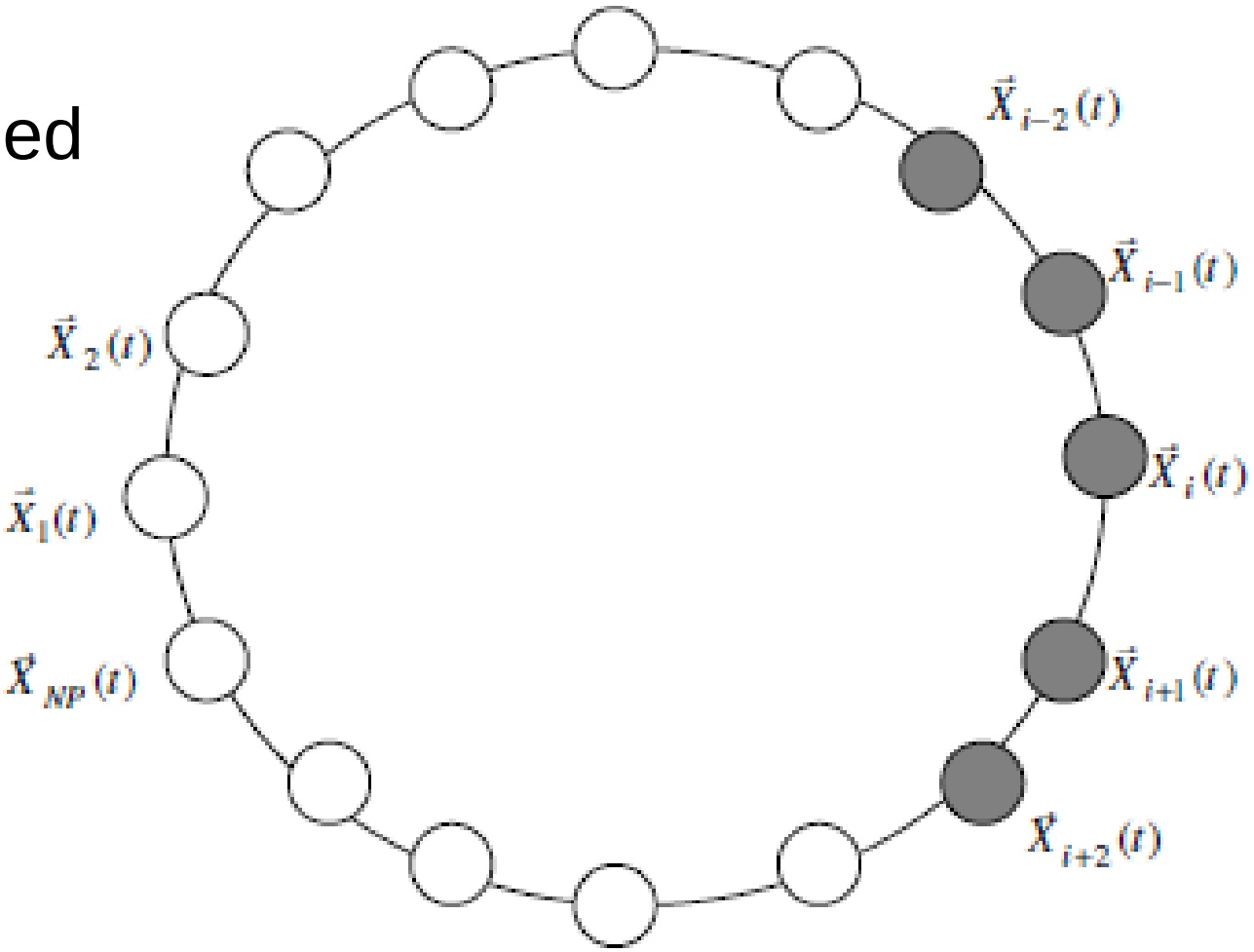
where S_F is the set of all successful scale factors at generation G and $mean_L$ is the Lehmer mean:

$$mean_L(S_F) = \frac{\sum_{F \in S_F} F^2}{\sum_{F \in S_F} F}$$

JADE usually performs best with 1/*c* chosen from [5, 20] and *p* from [5%, 20%]

Differential Evolution with Neighborhood-based Mutation

Uses index based topological Neighborhood.



Swagatam Das, A Abraham, Uday K. Chakraborty, and A Konar, "Differential evolution with a neighborhood based mutation operator: a comparative study", *IEEE Transactions on Evolutionary Computing*, Vol 13, No. 3, June 2009.

NANYANG TECHNOLOGICAL UNIVERSITY

Local Mutation Model:

$$\vec{L}_i(t) = \vec{X}_i(t) + \alpha \cdot (\vec{X}_{n\_best_i}(t) - \vec{X}_i(t)) + \beta \cdot (\vec{X}_p(t) - \vec{X}_q(t))$$

Global Mutation Model:

$$\vec{g}_i(t) = \vec{X}_i(t) + \alpha \cdot (\vec{X}_{g\_best}(t) - \vec{X}_i(t)) + \beta \cdot (\vec{X}_{r_1}(t) - \vec{X}_{r_2}(t))$$

Combined Model for Donor Vector generation:

$$\vec{V}_i(t) = w.\vec{g}_i(t) + (1-w).\vec{L}_i(t)$$

The weight factor *w* may be adjusted during the run or self-adapted through the evolutional learning process.

The MDE_*p*BX Algorithm

- In MDE_*p*BX, we propose a new mutation strategy, a fitness-induced parent selection scheme for the binomial crossover of DE, and a simple but effective scheme of adapting two of its most important control parameters.
- First, a less greedy and more explorative variant of DE/current-to-best/1 is used. We call it DE/current-to-gr_best/1.
- DE/current-to-gr_best/1 utilizes the best member of a dynamic group of *q*% population members to perturb the target vector.
- This overcomes the limitations of fast but less reliable convergence performance of DE/current-to-best/1.

Sk. Minhazul Islam, S. Das, S. Ghosh, S. Roy, and P. N. Suganthan, "An Adaptive Differential Evolution Algorithm with Novel Mutation and Crossover Strategies for Global Numerical Optimization", *IEEE Trans. on* SMC-B, Vol. 42, No. 2, pp. 482-500, 2012.

Algorithmic Components of MDE_*p*BX

- Second, we modify the conventional binomial crossover of DE by introducing a fitness-induced bias in the selection of parents from the current generation.
- The exploitative crossover scheme is referred to as "*p*-best crossover".
- Here a mutant vector is allowed to exchange its components through binomial crossover with a randomly selected member from the *p* top-ranked individuals of the current generation instead of exchanging with the parent of the same index
- Third, we suggest simple schemes to update the values of *F* and *Cr* in each generation, guided by the knowledge of their successful values that were able to generate better offspring in the last generation.

The DCMA-EA Algorithm

- Hybridization aims at combining the operators and methodologies from different Evolutionary Computation paradigms to form a single algorithm that enjoys a statistically superior performance.
- DCMA-EA is an hybridization of CMA-ES and DE algorithm that aims at improving the performance of the CMA-ES algorithm on complicated landscapes, such as noisy and hybrid composition functions.
- In the proposed hybridization, our aim is to incorporate the difference-vector based mutation scheme of DE into CMA-ES as these difference vectors have the ability to adjust to the natural scaling of the problem.
- Further, in order to enhance the diversity among the population members as well as increase the convergence speed, the selection and crossover operators of DE have also been embedded.

S. Ghosh, S. Das, S. Roy, Sk. Minhazul Islam, and P. N. Suganthan "A Differential Covariance Matrix Adaptation Evolutionary Algorithm for Real Parameter Optimization", *Information Sciences,* Vol. 182, No. 1, pp 199-219 Jan. 2012.

Prominent Real-world Applications of DE:

Sub areas and details	Types of DE applied and references
Electrical Power Systems	
Economic dispatch Optimal power flow Power system planning, generation expansion planning Capacitor placement Distribution systems' network reconfiguration Power filter, power system stabilizer	Chaotic DE [S31], Hybrid DE with acceleration and migration [S87], DE/rand/1/bin [S88], hybrid DE with improved constraint handling [S89], variable scaling hybrid DE [S90] DE/target-to-best/1/bin [S91], Cooperative Co-evolutionary DE [S92], DE/rand/1/bin with non-dominated sorting [S93], conventional DE/rand/1/bin [S94, S96], DE with Random Localization (DERL) [S95]. Modified DE with fitness sharing [S97], conventional DE/rand/1/bin [S98], comparison of 10 DE strategies of Storn and Price [S99], Robust Searching Hybird DE (RSHDE) [S100] Hybrid of Ant System and DE [S49] Hybrid DE with variable scale factor [S101], mixed integer hybrid DE [S185]. Hybrid DE with acceleration and migration operators [S102], DE/target-to-best/1/bin [S103], hybrid of DE with ant systems [S104]
Electromagnetism, Propagation, and Microwave Engineering	
Capacitive voltage divider Electromagnetic inverse scattering Design of circular waveguide mode converters Parameter estimation and property analysis for electromagnetic devices, materials, and machines electromagnetic imaging Antenna array design	Multi-Objective DE (MODE) and NSDE (DE with Non-dominated Sorting) [S105] DE/rand/1/bin [S106], conventional DE with individuals in groups (GDES) [S107], Dynamic DE [77] DE/rand/1/bin [S108] DE/rand/1/bin [S109 – S111, S113], DE/target-to-best/1/bin [S112] conventional DE/rand/1/bin [S114, S115], DE/best/1/bin [S116] multi-member DE (see [93] for details) [S117], hybrid real/integer-coded DE [S118], DE/rand/1/bin [S119, S120], modified DE with refreshing distribution operator and fittest individual refinement operator [S121], DE/best/1/bin [S122], MOEA/D-DE [68,69]
Control Systems and Robotics	
System identification Optimal control problems Controller design and tuning Aircraft control nonlinear system control	Conventional DE/rand/1/bin [S123 – S126] DE/rand/1/bin and DE/best/2/bin [S127], modified DE with adjustable control weight gradient methods [S128]. Self adaptive DE [S129], DE/rand/1 with arithmetic crossover [S130], DE/rand/1/bin with random scale factor and time-varying *Cr* [S131]. Hybrid DE with downhill simplex local optimization [55]. Hybrid DE with convex mutation [15].

From TEVC DE-survey, Feb 2011.

Sub areas and details	Types of DE applied and references
Bioinformatics	
Gene regulatory networks	DE with adaptive local search (see [22] for details) [63], hybrid of DE and PSO [S137]
Micro-array data analysis	Multi-objective DE-variants (MODE, DEMO) [S138]
Protein folding	DE/rand/1/bin [S139]
Bioprocess optimization	DE/rand/1/bin [S140]
Chemical Engineering	
Chemical process synthesis and design Phase equilibrium and phase study Parameter estimation of chemical process	Modified DE with single array updating [S141, 7], 10 DE-variants of Storn and Price (see [74,75]) compared in [S142, S144], multi-objective DE [S143], hybrid DE with migration and acceleration operators [S145]. DE/rand/1/bin [S146]. Hybrid DE with geometric mean mutation [S147], DE/target-to-best/1/exp [S148].
Pattern Recognition and Image Processing	
Data clustering Pixel clustering and region-based image segmentation Feature extraction Image registration and enhancement Image Watermarking	DE/rand/1/bin [S149], DE with random scale factor and time-varying crossover rate [20], DE with neighborhood-based mutation [S150] Modified DE with local and global best mutation [S151], DE with random scale factor and time-varying crossover rate [S152]. DE/rand/1/bin [S153] DE/rand/1/bin [S154], DE with chaotic local search [S155] DE/rand/1/bin and DE/target-to-best/1/bin [S156]
Artificial neural networks (ANN)	
Training of feed-forward ANNs Training of wavelet neural networks (WNNs) Training of B-Spline neural networks	DE/rand/1/bin [S157, S160], generalization-based DE (GDE) [S158], DE/target-to-best/1/bin [S159] DE/rand/1/bin [S161] DE with chaotic sequence-based adjustment of scale factor *F* [S162]
Signal Processing	
estimation Digital filter design Fractional order signal processing	Dynamic DE (DyDE) [S163] DE/rand/1/bin [S164, S165], DE with random scale factor [S166] DE/rand/1/bin [S167]

Evolutionary Programming

One of the oldest Methods, not as competitive as DE or PSO.

Evolutionary Programming (EP)

- EP was proposed in the 1960s by L. Fogel to evolve intelligent systems [FOW66], [FOW95], [Fogel98].
- EP for numerical optimization was proposed by D. Fogel [Fogel91], [SNF95].
- Basic EP-steps are
  - Population Initialization
  - Mutation (EP has no cross-over)

    Gaussian (Classical EP or CEP [Fogel91]

    Cauchy (Fast EP or FEP [Yao96]

    Levy [LY04], or mixture of others
  - Selection (Tournament selection)

Evolutionary Programming

- In EP each parent (x_i, η_i) creates a single offspring (x_i', η_i') by mutation:

$$\eta_i'(j) = \eta_i(j)\exp(\tau' N(0,1) + \tau N_j(0,1))$$

$$x_i'(j) = x_i(j) + \eta_i(j) N_j(0,1)$$

τ and τ' are set to $\left(\sqrt{2\sqrt{n}}\right)^{-1}$ and $\left(\sqrt{2n}\right)^{-1}$ where *n* is the dimension of the problem

The initial value of η is set to 3.

Evolutionary Programming

- In CEP the η parameter is varied randomly independent of the problem, the performances of the individuals can be exploited more effectively.
- This may cause the η parameter to become too large or small and thus lead to oscillation or stagnation.
- A lower bound should be used which is problem dependant. Adaptation of lower bound was proposed in [LYN01]

Adaptive Evolutionary Programming [MS08a]

- In Adaptive EP (AEP) [MS08a] the η parameter is adapted based on the η values of the successful offspring [QHS09]
- The η is randomly initialized in the search range as

$$\eta_i = \lambda * rand(1,n).*(X\max - X\min)$$

where *Xmin, Xmax* are the minimum and maximum values of the search space

λ =0.8/sqrt(*D*), *D* is the dimension of the problem [WCZZ08]

R. Mallipeddi, P. N. Suganthan, "Evaluation of Novel Adaptive Evolutionary Programming on Four Constraint Handling Techniques ", *IEEE Congress on Evolutionary Computation,* pp. 4045-4052, Hong Kong, June 2008.

Adaptive Evolutionary Programming

- During the learning period (lp), the η values of the successful offspring which becomes parents in the next generation are saved in *sneta* η memory in every generation. After each learning period, the parameter will be updated based on the *sneta* of the previous (lp) generations as:

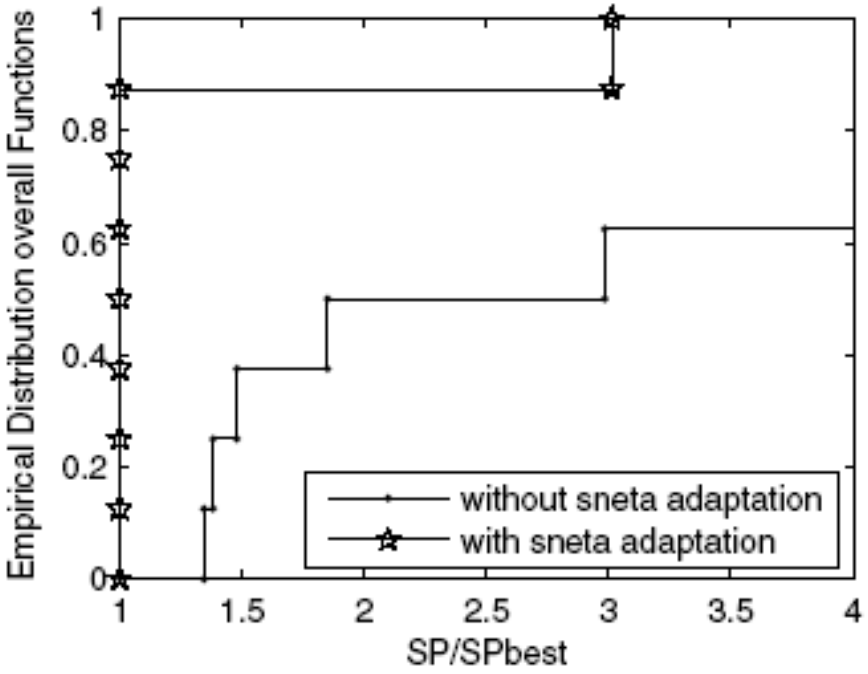
$$\eta_i = msneta + P * (2 * rand(1, n) - 1) * msneta$$

where *msneta* is the mean of *sneta. P* is a small perturbation factor.

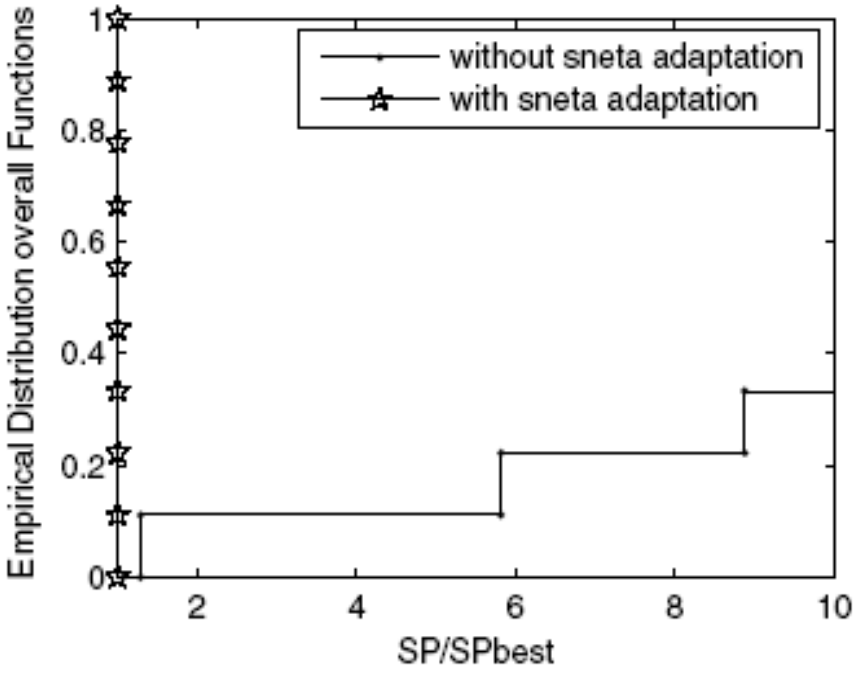
Test Problems [MS08a]

- First 13 test problems of CEC 2006 were considered to compare the performance of AEP and CEP.
- Each algorithm was run 25 times on each test problem.
- The success performance graphs [SHLDCAT05] are on the next slides.

Results



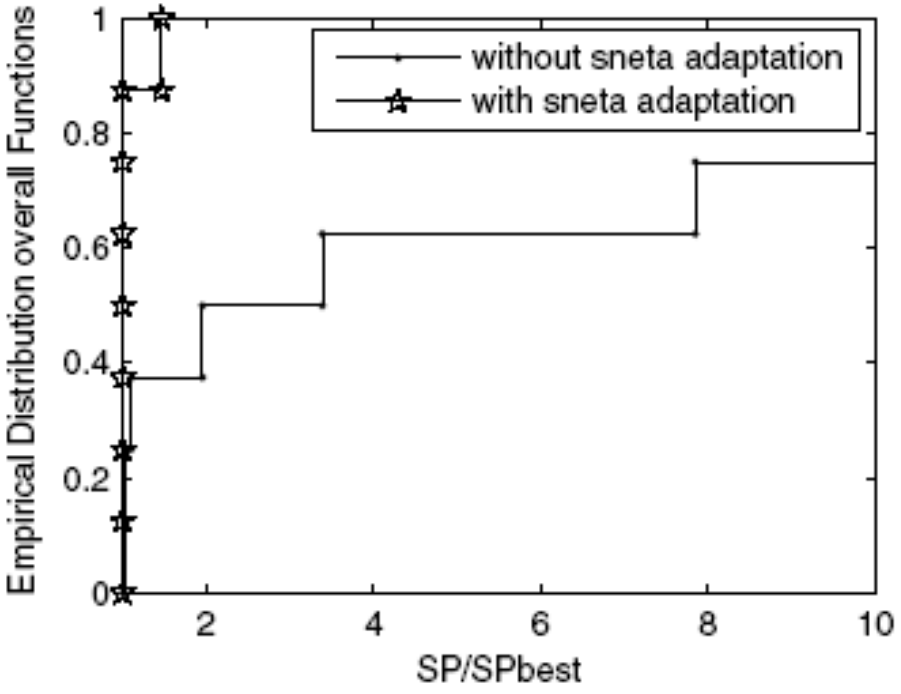
Superiority of Feasible



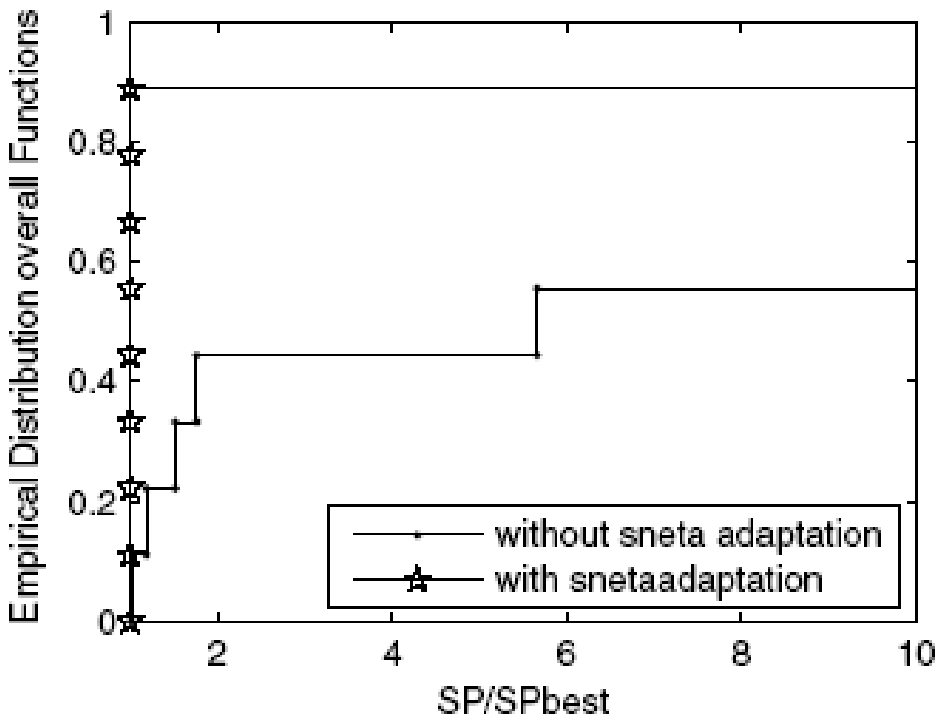
Self-Adaptive Penalty

Normalized Success Performance (Pl. refer to [SHLDCAT05] for details)

Results



Epsilon Constraint



Stochastic Ranking

Normalized Success Performance

NANYANG TECHNOLOGICAL UNIVERSITY

Ensemble of Mutation Strategies

- Different mutation strategies (Gaussian, Cauchy and Levy) have their own advantages and disadvantages
- Gaussian- with smaller step size useful for local search
- Levy and Cauchy- larger step size useful for global search

Cauchy Distribution

- The one-dimensional density function centered at the origin is defined by:

$$f_t(x) = \frac{1}{\pi} \frac{t}{t^2 + x^2}, \quad -\infty < x < \infty$$

- where $t > 0$ is the scale parameter
- The corresponding distribution function is

$$F_t(x) = \frac{1}{2} + \frac{1}{\pi} \arctan\left(\frac{x}{t}\right)$$

Cauchy Distribution

- The shape of $f_t(x)$ resembles that of the Gaussian density function but approaches the *x*-axis so slowly that an expectation does not exist. As a result, the variance of the Cauchy distribution is infinite.

Mixed Mutation Strategies

- Performance of the EP can be improved by using different mutation operators simultaneously to benefit from their advantages.
- Such a strategy can be referred to as a mixed mutation strategy [Chellapilla98], [DHHH07], [SF97], [WHY05], [YL98], [YLL99].

Designing Mixed Mutation Strategies

- Different ways of designing mixed mutation strategies [YL98],[DHHH07]

1. linear combination of Gaussian and Cauchy distributions [chellapilla98]
2. Improved Fast EP (IFEP) [YLL99] implements Cauchy and Gaussian mutations simultaneously and generates two individuals; the better one will be chosen to compete with the parent population by tournament selection.

Designing Mixed Mutation Strategies

- The idea of IFEP was also applied in mixing Levy's mutations with various scaling parameters [LY04]
- The idea of IFEP using all the three mutation operators was also proposed [WHY05]
- A reinforcement learning can be used to adopt the mutation operator [ZL08]

Ensemble of Mutation Strategies [MMS08]

- In the ensemble the 2 mutation strategies used are

1. Gaussian
2. Cauchy

- Each mutation operator has one population associated with it.
- In every generation the population members produce offspring population by the respective mutation operator

$$\eta'_{G,i}(j) = \mathrm{P_g} * successneta_G(j) * [2 * rand - 1] + successneta_G(j)$$

$$x'_{G,i}(j) = x_{G,i}(j) + \eta_{G,i}(j) N_j(0,1)$$

$$\eta'_{C,i}(j) = \mathrm{P_c} * successneta_c(j) * [2 * rand - 1] + successneta_c(j)$$

$$x'_{C,i}(j) = x_{C,i}(j) + \eta_{C,i}(j) \delta_j$$

R. Mallipeddi, S. Mallipeddi, P. N. Suganthan, "Ensemble strategies with adaptive evolutionary programming", *Information Sciences*, vol. 180, no. 9, May 2010, pp. 1571-1581

Ensemble of Mutation Strategies

- In the selection process the parents have to compete not only with offspring population of itself but also with the offspring population of other mutation operator.
- Hence, in ensemble every function call is utilized by each population associated with mutation operators in the ensemble.
- Due to this, the offspring population produced by a particular mutation operator may dominate the others at a particular stage of the optimization process. Furthermore, an offspring produced by a particular mutation operator may be rejected by its own population, but could be accepted by the population of another mutation operator.

- Therefore, the ensemble transforms the burden of choosing a particular mutation operator and tuning the associated parameter values for a particular problem into an advantage.

Experimental Setup [MMS08]

- To compare the performance of the algorithms, they are tested on 10D and 30D versions of the CEC2005 problems [SHLDCAT05].
- The dynamic lower bound adaptation (DLB2) in [LYN01] is used to adapt the lower bound.
- **CEP & FEP** [5]

1. Population size $\mu = 10$
2. Tournament size $q = 10$
3. Initial standard deviations: Initial Lower Bound is 0.1 (for both CEP and FEP)
4. Maximum Generations: 10000 for 10D problems and 30000 for 30D problems (total function evaluations are kept the same).

Experimental Setup

- The lower bound is updated after every 5 generations as given in [LYN01].
- To compare the performance of the CEP and FEP with AEP, the population size and the tournament size are kept the same.
- **Adaptive Evolutionary Programming**

1. Population size μ=10
2. Tournament size q=10
3. Initial lower bound for AEP with Gaussian: 0.1
4. Initial lower bound for AEP with Cauchy: 0.001
5. Maximum Generations: 10000 for 10D problems and 30000 for 30D problems

Experimental Setup

- Different values of P_g and P_c are used to show that AEP performs consistently better than the CEP and FEP. The AEP with different values of P_g and P_c are referred to as:

1. AEP1- AEP with Gaussian mutation operator and $P_g = 0.3$
2. AEP2- AEP with Gaussian mutation operator and $P_g = 0.5$
3. AEP3- AEP with Cauchy mutation operator and $P_c = 0.5$
4. AEP4- AEP with Cauchy mutation operator and $P_c = 0.7$

Experimental Setup

- **Ensemble of Mutation Strategies**

1. Population size μ=10 for Gaussian and Cauchy
2. Tournament size q=10
3. Initial lower bound for Gaussian: 0.1
4. Initial lower bound for Cauchy: 0.001
5. Maximum Generations 5000 for 10D problems and 15000 for 30D problems (total function evaluations are kept the same).

- The maximum generations are halved in the ensemble to allow fair comparison between algorithms with single mutation operators and ensemble of mutation operators with the same number of total function evaluations. The different ensembles used are:

1. EN1- (AEP1+AEP3)
2. EN2- (AEP1+AEP4)

Comparisons [MMS08]

- Comparing the results of CEP and FEP with the results of AEP in Tables in the next slides, it can be observed that the performance of AEP with Gaussian and Cauchy mutations with different parameter settings are better than their counterparts.
- The better performance of AEP over EP is due to the *successneta* adaptation. The *successneta* value is appropriately changed for the problem being solved, mutation operators used, etc.

Performance Comparison

- To compare the performance of ensemble algorithms with the single mutation strategy algorithms, we employ two types of statistical tests.

1. *t*-test
2. Wilcoxon ranksum test

- *t*-test, being a parametric method, can be used to compare the performances of two algorithms on a single problem. When the performances of two algorithms are compared on multiples problems *t*-test is not valid as the normality assumption can be violated [GMLH09].
- Therefore, to compare the performances of two algorithms over a set of different problems, we can use non-parametric tests, e.g. Wilcoxon ranksum test [GMLH09] .

Performance on 10D Problems

Prob. No.	CEP		FEP	
	Mean	Std	Mean	Std
F01	1.088E-27	3.553E-27	1.354E-27	5.176E-27
F02	3.426-25	1.446E-24	9.046E-20	3.984E-19
F03	4.137E+01	8.458E+01	4.962E+01	1.615E+02
F04	1.827E+01	8.015E+01	5.043E-07	2.352E-06
F05	3.437E+00	3.425E+00	1.009E-14	1.687E-14
F06	4.431E+00	3.541E+01	1.108E-14	2.432E-14
F07	2.072E+02	2.880E-02	7.880E-02	4.490E-02
F08	9.223E+02	2.433E-01	7.140E-02	3.350E-02
F09	1.679E+01	6.937E+00	7.403E+00	3.199E+00
F10	2.153E+01	1.029E+01	1.047E+01	5.189E+00
F11	2.144E+01	8.574E+00	1.000E+01	3.291E+00
F12	1.564E+03	3.995E+02	6.392E+02	2.481E+02
F13	8.400E+01	9.866E+01	4.800E+01	6.532E+01
F14	7.908E+01	7.574E+01	1.759E+01	3.673E+01

Performance on 10D problems

Prob. No.	AEP1		AEP2	
	Mean	Std	Mean	Std
F01	0	0	0	0
F02	8.936E-29	1.067E-28	4.463E-28	1.025E-27
F03	7.141E+01	1.887E+02	1.364E+02	2.847E+02
F04	1.050E-28	1.413E-28	4.783E-12	2.194E-11
F05	2.239E+00	6.193E+00	9.948E-16	1.628E-15
F06	1.198E+01	9.994E+00	3.232E+00	7.558E+00
F07	8.310E-02	3.890E-02	8.580E-02	4.580E-02
F08	7.050E-02	4.890E-02	7.420E-02	3.650E-02
F09	2.826E+00	1.739E+00	4.537E+00	2.741E+00
F10	1.413E+01	5.331E+00	1.524E+01	6.156E+00
F11	1.280E+00	9.363E-01	2.160E+00	1.179E+00
F12	4.305E+02	2.413E+02	4.791E+02	1.983E+02
F13	1.199E+02	1.599E+02	6.265E+01	1.112E+02
F14	2.684E+02	9.465E+01	2.534E+02	1.442E+02

Performance on 10D problems

Prob. No.	AEP3		AEP4	
	Mean	Std	Mean	Std
F01	0	0	0	0
F02	8.936E-29	1.067E-28	8.936E-29	1.067E-28
F03	7.141E+01	1.887E+02	7.141E+01	1.887E+02
F04	1.050E-28	1.413E-28	1.050E-28	1.413E-28
F05	2.239E+00	6.193E+00	2.239E+00	6.193E+00
F06	1.198E+01	9.994E+00	1.198E+01	9.994E+00
F07	8.310E-02	3.890E-02	8.310E-02	3.890E-02
F08	7.050E-02	4.890E-02	7.050E-02	4.890E-02
F09	2.826E+00	1.739E+00	2.826E+00	1.739E+00
F10	1.413E+01	5.331E+00	1.413E+01	5.331E+00
F11	1.280E+00	9.363E-01	1.280E+00	9.363E-01
F12	4.305E+02	2.413E+02	4.305E+02	2.413E+02
F13	1.199E+02	1.599E+02	1.199E+02	1.599E+02
F14	2.684E+02	9.465E+01	2.684E+02	9.465E+01

Performance on 30D

Prob. No.	CEP		FEP	
	Mean	Std	Mean	Std
F01	4.225E-22	9.936E-22	4.225E-22	9.936E-22
F02	6.951E-01	2.426E+00	6.951E-01	2.426E+00
F03	6.179E+01	7.260E+01	6.179E+01	7.260E+01
F04	2.235E+04	1.107E+04	2.235E+04	1.107E+04
F05	1.534E+01	1.980E+00	1.534E+01	1.980E+00
F06	1.227E+01	2.264E+00	1.227E+01	2.264E+00
F07	1.302E+03	1.421E+02	1.302E+03	1.421E+02
F08	4.166E+03	4.650E+02	4.166E+03	4.650E+02
F09	8.332E+01	2.889E+01	8.332E+01	2.889E+01
F10	1.005E+02	3.470E+01	1.005E+02	3.470E+01
F11	8.711E+01	2.540E+01	8.711E+01	2.540E+01
F12	5.380E+03	7.265E+02	5.380E+03	7.265E+02
F13	4.800E+01	7.703E+01	4.800E+01	7.703E+01
F14	1.233E+02	1.066E+02	1.233E+02	1.066E+02

Performance on 30D

Prob. No.	AEP1		AEP2	
	Mean	Std	Mean	Std
F01	1.798E-26	7.178E-26	1.798E-26	7.178E-26
F02	1.245E+03	1.324E+03	1.245E+03	1.324E+03
F03	3.042E+02	3.831E+02	3.042E+02	3.831E+02
F04	1.131E+04	6.133E+03	1.131E+04	6.133E+03
F05	2.343E+00	1.943E+00	2.343E+00	1.943E+00
F06	3.389E+00	2.620E+00	3.389E+00	2.620E+00
F07	6.040E-02	1.367E-01	6.040E-02	1.367E-01
F08	2.190E-02	1.620E-02	2.190E-02	1.620E-02
F09	5.483E+01	1.535E+01	5.483E+01	1.535E+01
F10	6.408E+01	1.561E+01	6.408E+01	1.561E+01
F11	5.832E+01	1.518E+01	5.832E+01	1.518E+01
F12	3.126E+03	5.374E+02	3.126E+03	5.374E+02
F13	6.000E+01	1.154E+02	6.000E+01	1.154E+02
F14	6.779E+01	1.343E+02	6.779E+01	1.343E+02

Performance on 30D

Prob. No.	AEP3		AEP4	
	Mean	Std	Mean	Std
F01	0	0	0	0
F02	1.891E-05	3.193E-05	1.891E-05	3.193E-05
F03	6.633E+01	1.389E+02	6.633E+01	1.389E+02
F04	7.884E+03	3.848E+03	7.884E+03	3.848E+03
F05	1.251E-14	5.140E-15	1.251E-14	5.140E-15
F06	1.342E+00	7.429E-01	1.342E+00	7.429E-01
F07	1.640E-02	2.610E-02	1.640E-02	2.610E-02
F08	1.010E-02	1.080E-02	1.010E-02	1.080E-02
F09	2.181E+01	6.166E+00	2.181E+01	6.166E+00
F10	8.980E+01	2.167E+01	8.980E+01	2.167E+01
F11	1.248E+01	4.611E+00	1.248E+01	4.611E+00
F12	2.228E+00	5.338E+02	2.228E+00	5.338E+02
F13	3.600E+01	7.000E+01	3.600E+01	7.000E+01
F14	3.705E+01	4.208E+01	3.705E+01	4.208E+01

Comparison [MMS08]

- In each column -1, 0 and 1 indicate that the single algorithm is better than the ensemble, equal to the ensemble and worse than the ensemble respectively according to the *t*-test.
- When comparing the two ensembles (EN1 and EN2) with the four single mutation strategy algorithms (AEP1, AEP2, AEP3 and AEP4) over 14, 10D and 30D problems, we have 224 comparisons.
- In the total 224 statistical comparisons between the four single mutation strategy algorithms and the two ensembles, ensemble performs better in 93 cases and equally in 108 cases while it performs worse in 23 cases
- Out of the 23 worst cases 7 cases correspond to F01 (Sphere function) while 10 cases correspond to F05 and F06 (both are Ackley's functions) a relatively easier multimodal problem.

NANYANG TECHNOLOGICAL UNIVERSITY

Prob. No.	10D Problems								30D Problems							
	EN1				EN2				EN1				EN2			
	Mean		Std		Mean		Std		Mean		Std		Mean		Std	
F01	1.636E-12		2.46E-12		2.751E-17		3.947E-17		3.960E-16		5.971E-16		3.661E-17		9.620E-17	
	-1	-1	-1	-1	0	0	0	0	-1	0	-1	-1	0	0	0	0
F02	2.554E-06		1.276E-05		6.480E-14		1.647E-13		2.600E-03		7.900E-03		3.200E-03		8.100E-03	
	0	0	0	0	0	0	0	0	1	1	0	0	1	1	0	0
F03	1.299E+01		2.465E+01		1.021E+01		2.278E+01		1.191E+02		1.561E+02		1.487E+02		2.547E+02	
	0	0	0	1	0	0	0	1	1	1	0	0	0	1	0	0
F04	2.521E-09		4.392E-09		5.152E-14		8.040E-14		2.547E+03		2.248E+03		5.199E+03		4.247E+03	
	0	0	-1	-1	0	0	-1	0	1	1	1	1	1	1	1	1
F05	3.799E-07		4.161E-07		4.677E-09		6.949E-09		0		0		0		6.647E-10	
	-1	0	0	-1	-1	0	0	-1	1	1	-1	-1	1	1	-1	-1
F06	6.251E-07		1.271E-06		3.704E-09		5.318E-09		1.604E+00		7.962E-01		1.453E+00		1.004E+00	
	-1	0	1	1	-1	0	1	1	1	1	0	0	1	1	0	0
F07	6.420E-02		3.670E-02		7.210E-02		3.390E-02		1.510E-02		1.500E-02		1.380E-02		1.380E-02	
	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0

Prob. No.	10D Problems								30D Problems							
	EN1				EN2				EN1				EN2			
	Mean		Std		Mean		Std		Mean		Std		Mean		Std	
F08	6.350E-02		3.120E-02		7.860E-02		4.950E-02		1.570E-02		1.210E-02		2.190E-02		1.760E-02	
	0	0	0	0	0	0	0	0	0	1	0	0	0	1	-1	0
F09	2.507E+00		1.576E+00		2.070E+00		1.375E+00		1.380E+01		5.453E+00		1.327E+01		3.962E+00	
	-1	1	0	1	1	1	0	1	1	1	1	1	1	1	1	1
F10	9.928E+00		3.777E+00		1.257E+01		4.474E+00		6.192E+01		1.890E+01		5.804E+01		1.214E+01	
	0	1	1	1	0	0	0	0	0	1	1	1	0	1	1	1
F11	6.000E-01		7.638E-01		4.400E-01		8.699E-01		7.400E+00		4.282E+00		6.840E+00		2.897E+00	
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
F12	2.132E+02		1.709E+02		3.460E+02		1.566E+02		1.680E+03		4.290E+02		1.816E+03		4.651E+02	
	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1
F13	2.400E+01		4.359E+01		2.800E+01		4.583E+01		5.200E+01		1.046E+02		3.600E+01		9.074E+01	
	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0
F14	9.354E+00		2.731E+01		5.360E+00		1.975E+01		2.018E+01		2.658E+01		1.962E+01		2.783E+01	
	0	0	1	1	0	0	1	1	0	1	0	0	0	1	0	0

Comparison

- To compare the performances of the ensemble and single mutation strategy algorithms over all fourteen 10D and 30D problems, we use Wilcoxon test. Wilcoxon test is a rank based test, where all observations are ranked and the algorithm with the superior rank is considered the best algorithm. Wilcoxon test is performed in two different ways:

1. The standard Wilcoxon test with ranking all numerical observations.
2. Wilcoxon test does not consider the absolute difference between the neighboring values. Hence, we rank the data such that the neighboring data items having an absolute difference between them less than a specified value (e-5) are given the same rank.

- In each row 0 and 1 represent that the performance of that particular ensemble is equal and better than the algorithm with a single mutation strategy respectively.

Wilcoxon Ranksum Results

Test	EN1				EN2			
General Wilcoxon Test	1	1	0	0	1	1	1	1
Wilcoxon with specified tolerance value (e-5)	1	1	1	1	1	1	1	1

Multi-Objective Optimization

Currently Decomposition Based MOEAs are competitive.

CEC'07 Special Session / Competition on Performance Assessment of real-parameter MOEAs

CEC09 Special Session / Competition on Performance Assessment of real-parameter MOEAs

A. Zhou, B-Y. Qu, H. Li, S-Z. Zhao, P. N. Suganthan, Q. Zhang, "Multiobjective Evolutionary Algorithms: A Survey of the State-of-the-art", *Swarm and Evolutionary Computation*, Vol. 1, No. 1, pp. 32-49, Mar 2011.

MO Problems

- In real world applications numerous optimization problems have more than one objective in conflict with each other.
- The aim is to find Pareto optimal trade-off solutions that represent the best possible compromises among the different objectives
- A bound constrained multi-objective optimization problem (MOP) can be described as follows:

Minimize/maximize $F(x) = (f_1(x), f_2(x), \ldots, f_m(x))^T$

Subject to $x_i^L \leq x_i \leq x_i^U$

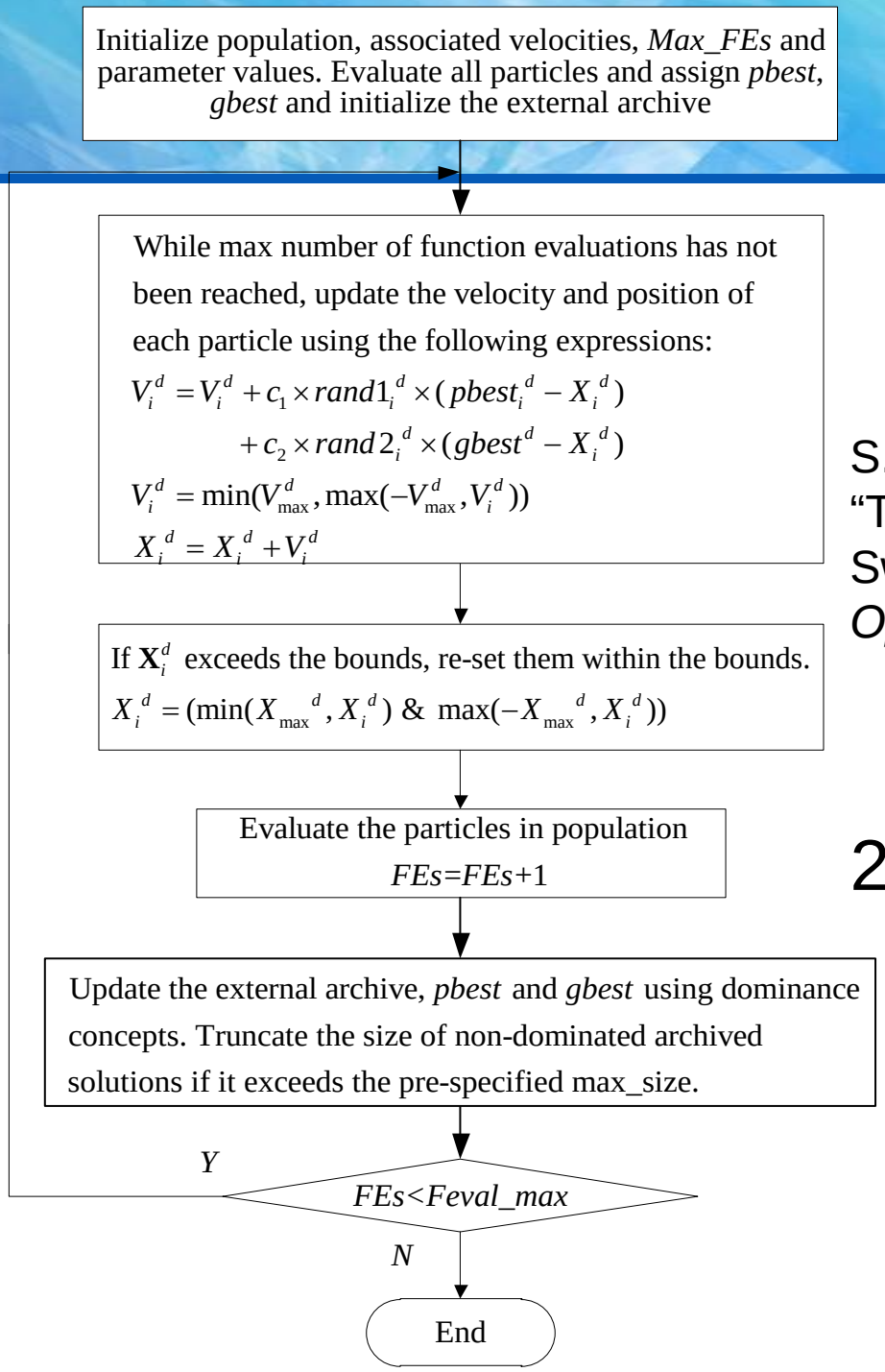
Non-domination sorting

A solution x_1 is said to dominate another solution x_2 if the both conditions specified below are satisfied:

1. The solution x_1 is no worse than x_2 in all objectives.
2. The solution x_1 is strictly better than x_2 in at least one objective.

The process of non-domination sorting is time-consuming.

In addition, purely non-domination based selection is a greedy approach and may lead to a locally optimal solution set.



S. Z. Zhao and P. N. Suganthan, "Two-lbests Based Multi-objective Particle Swarm Optimizer", *Engineering Optimization*, Vol. 43 No. 1 pp. 1-17, 2011.

2LB-MOPSO

2LB-MOPSO

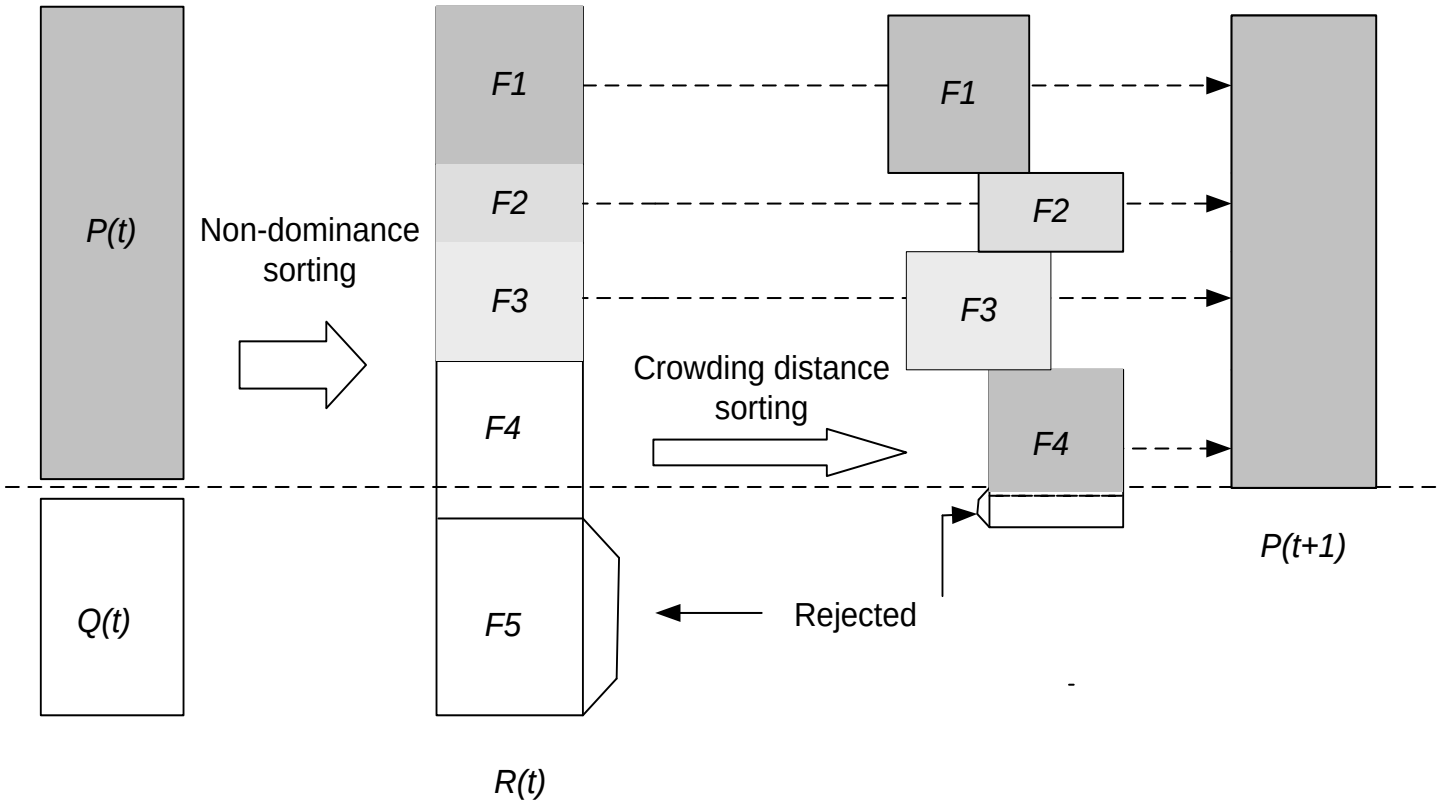
- **All MOPSO implementations select the global best (*gbest*) or local best (*lbest*) from only the non-dominated solutions in the external archive in the current iteration. This approach emphasizes the elitism at the expense of diversity with possible adverse consequences due to the potential loss of diversity if the size of the non-dominated solutions is very small for several consecutive iterations.**

- **We proposed that the *gbest*s or *lbest*s should be selected from the top fronts in a non-domination sorted external archive of reasonably large size.**

- **A novel two local bests (*lbest*s) based MOPSO version (2LB-MOPSO) is also proposed to effectively focus the search around a small region in the parameter space in the vicinity of the best existing fronts whereas in the current MOPSO variants, the *pbest* and *gbest* (or *lbest*) of a particle can be far apart in the parameter space due to the random nature of their selection process thereby potentially resulting in a chaotic search process.**

- **The same two exemplars are used for a few iterations.**

The members in *P(t+1)* to be used as lbest guides

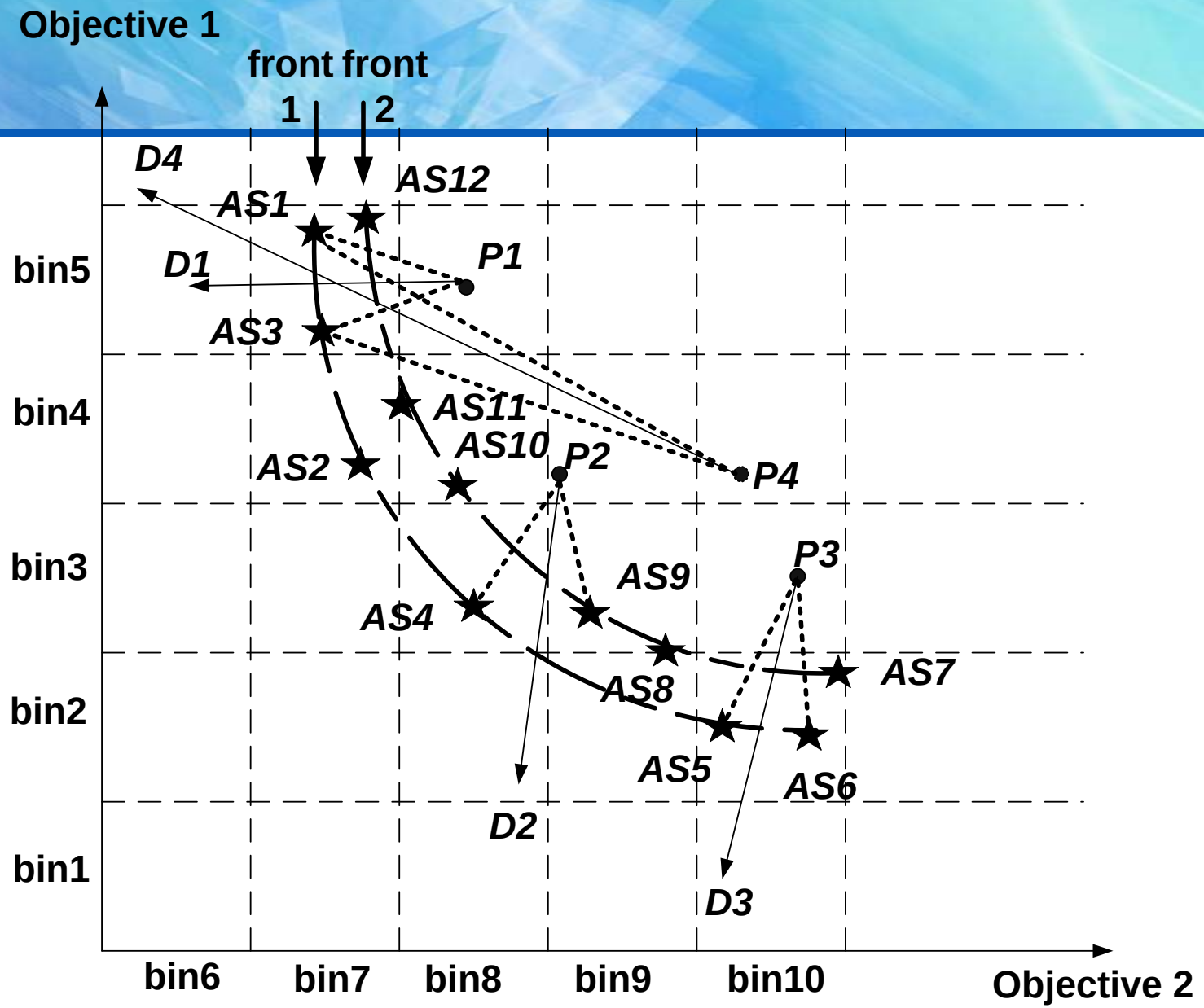


P(t): External archive of generation *t*

Q(t): New solutions of generation *t*

R(t): Mixed-temporary archive

P(t+1): New external archive of generation *t+1*



★ Archived Solution (*AS*)

● Particle (*P*)

← Moving Direction of Particle (*D*)

NANYANG TECHNOLOGICAL UNIVERSITY

Experimental Procedures

- **19 CEC2007 multi-objective benchmark problems (2 ,3 and 5 objectives), 25 runs.**

- **2LB-MOPSO is compared against two MOPSO variants and seven MOEA variants. Eight of these MOEA variants were from the CEC07 Special Session and Competition.**

- **The R indicator and the hypervolume indicator capture both convergence to the Pareto front and diversity of the approximated set of solutions are used as the comparison criteria.**

- **Additional experiments were conducted**

✓ **to compare the performance when the two *lbest*s are selected from a) a non-domination sorted external archive of a reasonably large size, b) only non-dominated (front 1) solutions.**

✓ **to show the benefit of selecting *gbest* (or *lbest*) from the neighborhood in both parameter space and objective space and the advantage of retaining the *gbest* or *lbest* for every particle in the same neighborhood for a few iterations (i.e. count)**

NANYANG TECHNOLOGICAL UNIVERSITY

Summation of ranks based on R & H indicators on all test problems

Algorithms:

A1- 2LB-MOPSO

A2- MOCLPSO

A3- MO_PSO

A4- NSGA2_SBX

A5- NSGA2_PCX

A6- GDE3

A7- MO_DE

A8- MOSaDE

A9- DEMOwSA

A10- MTS

Algorithm	FEs=5000 (*R+H*)			Fes=50000 (*R+H*)			Fes=500000 (*R+H*)			Overall Summation		
	M=2	M=3	M=5	M=2	M=3	M=5	M=2	M=3	M=5	M=2	M=3	M=5
A1	**22**	**18**	**37**	**40**	**30**	**40**	**52**	**42**	**52**	**114**	**90**	**129**
	(1)	**(1)**	**(2)**	**(2)**	**(1)**	**(3)**	**(2)**	**(2)**	**(3)**	**(1)**	**(1)**	**(2)**
A2	120	79	60	118	92	80	129	100	87	369	271	227
	(10)	(7)	(5)	(10)	(10)	(8)	(10)	(10)	(9)	(10)	(10)	(6)
A3	84	80	96	106	88	95	111	88	96	301	256	287
	(6)	(8)	(9)	(9)	(9)	(10)	(9)	(9)	(10)	(9)	(9)	(10)
A4	**35**	**49**	**33**	**42**	**49**	**27**	63	**50**	**18**	**140**	**148**	**78**
	(2)	**(3)**	**(1)**	**(3)**	**(3)**	**(1)**	(4)	**(3)**	**(1)**	**(2)**	**(3)**	**(1)**
A5	94	68	**41**	93	73	77	71	71	81	258	212	199
	(8)	(5)	**(3)**	(6)	(5)	(6)	(6)	(7)	(7)	(6)	(5)	(5)
A6	77	61	68	**32**	**31**	**37**	**46**	**27**	**40**	**155**	**119**	**145**
	(5)	(4)	(6)	**(1)**	**(2)**	**(2)**	**(1)**	**(1)**	**(2)**	**(3)**	**(2)**	**(3)**
A7	93	81	85	98	79	78	78	70	84	269	230	247
	(7)	(9)	(7)	(8)	(7)	(7)	(7)	(6)	(8)	(7)	(6)	(8)
A8	75	103	99	59	88	91	**58**	63	66	192	254	256
	(4)	(10)	(10)	(4)	(8)	(9)	**(3)**	(5)	(4)	(4)	(8)	(9)
A9	108	**49**	94	97	54	72	65	62	66	270	165	232
	(9)	**(2)**	(8)	(7)	(4)	(5)	(5)	(4)	(5)	(8)	(4)	(7)
A10	**61**	72	47	91	76	63	99	87	70	251	235	180
	(3)	(6)	(4)	(5)	(6)	(4)	(8)	(8)	(6)	(5)	(7)	(4)

Conclusions

- We propose a novel two local best (lbests) based MOPSO in which the two lbests are located close to each other in the vicinity of the best front sections. 2LB-MOPSO keeps the *lbest*s in a particular region for a few iterations. Therefore, the 2LB-MOPSO was able to focus the search around small regions in the vicinity of the best front sections while maintaining the diversity due to the usage of two lbests and binning.

- 2LB-MOPSO is applied to design multi-objective robust PID controllers. Comparative studies reveal that 2LB-MOPSO results are better than earlier reported results (Riccati, IGA and OSA methods) and the results obtained by NSGA-II.

- 2LB-MOPSO is also applied over different instances of the design problem of monopulse antenna arrays. For the purpose of keeping minimum Maximum Sidelobe Level (MSLL) and principal lobe Beam Width (BW) as two design objectives to be simultaneously achieved.

NANYANG TECHNOLOGICAL UNIVERSITY

MOEP with Ranksum Sorting and Diversified Selection

- Ranksum Sorting

  Divide every objective's range into N ranks/bins and sum the corresponding ranks for all objectives. The population is assigned with the rank-sum according to its position in the search space w. r. t. the summation of ranks/bins.

B . Y. Qu, P. N. Suganthan, "Multi-objective Evolutionary Programming without Non-domination Sorting is up to Twenty Times Faster", *IEEE Congress on Evolutionary Computation,* pp. 2934-2939, Norway, May 2009.

- The process can be described in the following steps:

Computing Ranksum [QS09]

Step 1: Select one unranked objective

Step 2: Get the maximum and minimum value of the selected objective to calculate the range for this objective

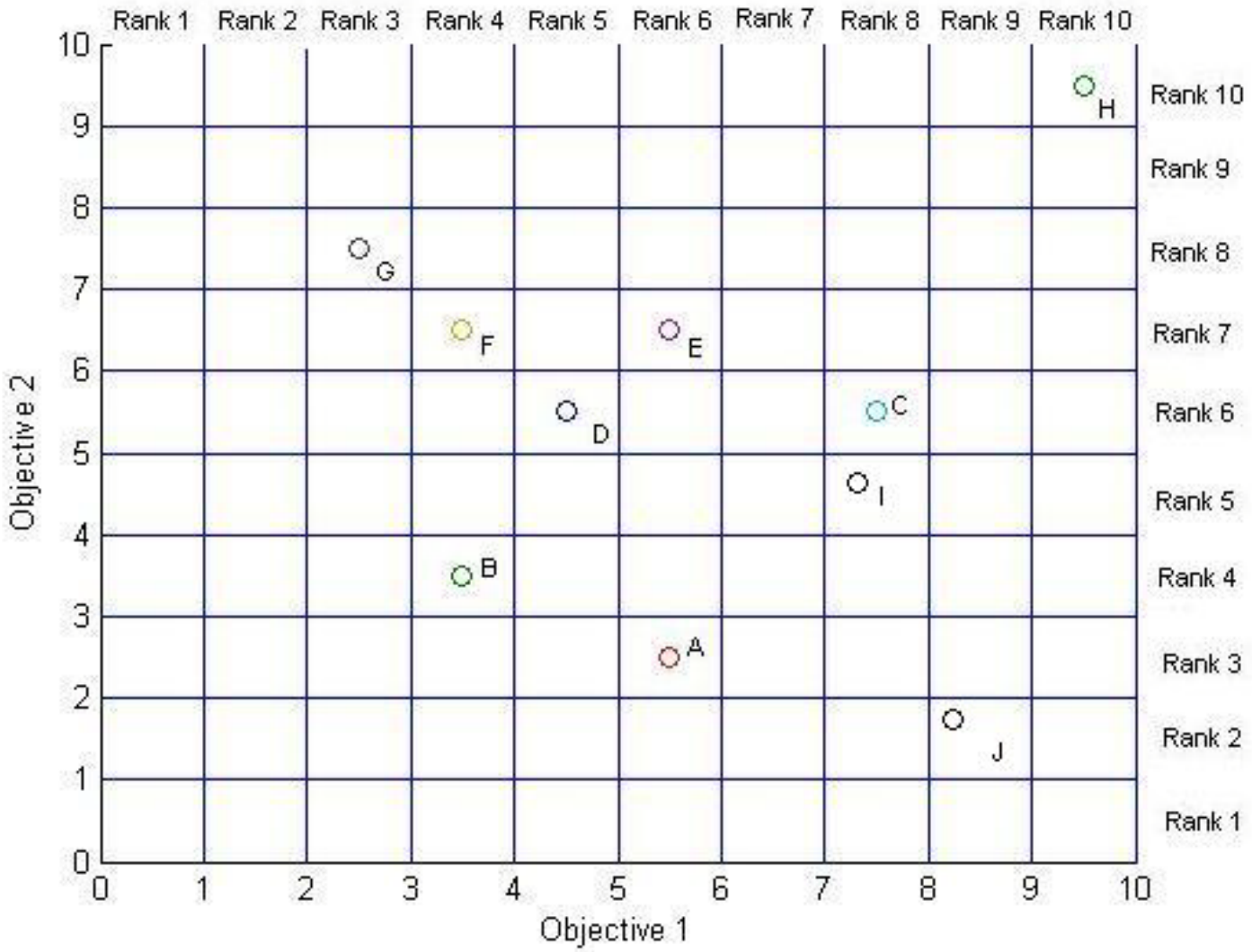
Step 3: Divide the searching range of this objective into N grids.

Step 4: For every point in the search space, identify which grid it belongs to and assign the corresponding rank to the point for the selected objective.

Step 5: If all objectives have been selected go to step 6, otherwise repeat Steps 1-4.

Step 6: Sum the ranks of every objectives of each solution to obtain the rank-sum of the solution. Similarly, obtain the rank-sum for all solutions in population.

Example


Rank 1
Rank 2
Rank 3
Rank 4
Rank 5
Rank 6
Rank 7
Rank 8
Rank 9
Rank 10
Objective 2
Objective 1
H
G
F
E
D
C
I
B
A
J

Example

Point	Objective 1 rank	Objective 2 rank	Rank-sum
A	6	3	9
B	4	4	8
C	8	6	14
D	5	6	11
E	6	7	13
F	4	7	11
G	3	8	11
H	10	10	20
I	8	5	13
J	9	2	11

MOEP with rank-sum sorting and diversified selection

- Diversified Selection

  In order to keep the diversity of the population during the selection process, the population will be divided into 2 sets, preferential set and backup set.

- The steps of getting the preferential set are as follow:

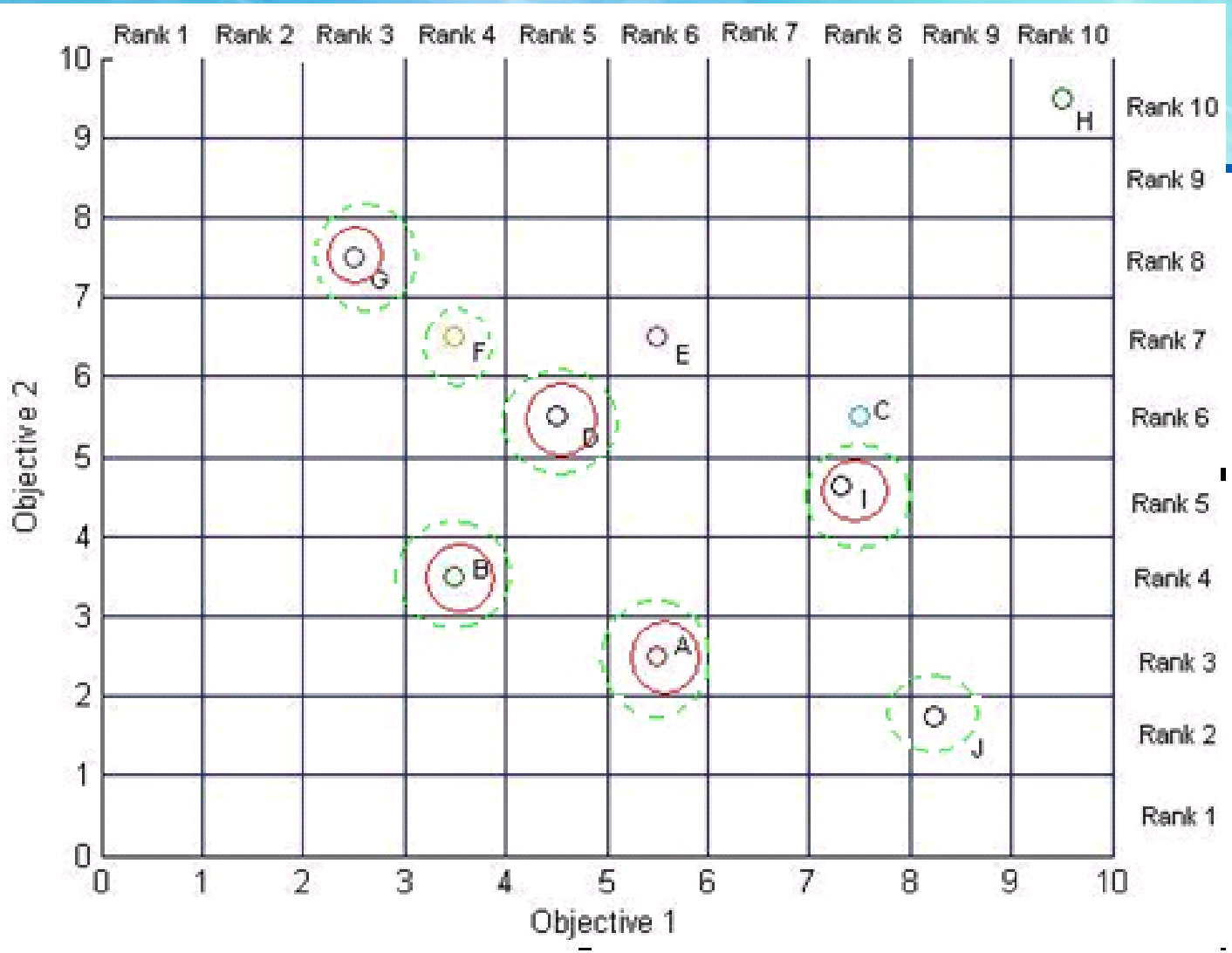
Diversified Selection

Step 1 Select one unselected objective

Step 2 For the selected objective, scan *P* percentage of the total rank (i.e. from rank 1 to rank *P*, *P* can be set to 80 or 90). For each rank (if this rank is not empty, otherwise just continue to the next rank) of the selected objective, the solutions with the lowest rank-sum will be chosen to enter preferential set.

Step 3 If all objectives have been selected, go to step 4. Otherwise repeat Steps 1-2.

Step 4 Collect the solutions not inside the preferential set and put them in the backup set.



The solutions chosen to be in the preferential set according to objective 1 are G, B, D, A, and I which are shown by solid circles. Similarly, G, F, D, I, B, A and J are chosen according to objective 2.

MODE based on summation of normalized objective values and diversified selection

Step 1	For m = 1 to M (*number_of_objectives)*
Step 2	Find the maximum and minimum objective values of the m^{th} objective and calculate the range of the m^{th} objective.
Step 3	Normalize the m^{th} objective values of all members using the equation below: $$f_m'(x) = \frac{f_m(x) - f_{\min}}{f_{\max} - f_{\min}}$$ where f_m' is the normalized m^{th} objective value.
Step 4	Endfor
Step 5	For i = 1 to NP (*population_size)*
Step 6	Sum all normalized objective values of the member to obtain a single value.
Step 7	Endfor

B. Y. Qu, P. N. Suganthan, "Multi-Objective Evolutionary Algorithms based on the Summation of Normalized Objectives and Diversified Selection", *Information Sciences*,180(17):3170-3181.

B. Y. Qu and P. N. Suganthan, "Multi-objective differential evolution based on the summation of normalized objectives and improved selection method," SDE-2011 *IEEE Symposium on Differential Evolution*, Paris, France, April 2011.

Diversified selection

TABLE III THE STEPS OF REMOVING BAD INDIVIDUALS

Step 1	Indentify the reference point using equation 1.
Step 2	Find the closest individual in population to the reference point and set it as reference individual.
Step 3	Remove all the individuals that dominated by the reference individual.

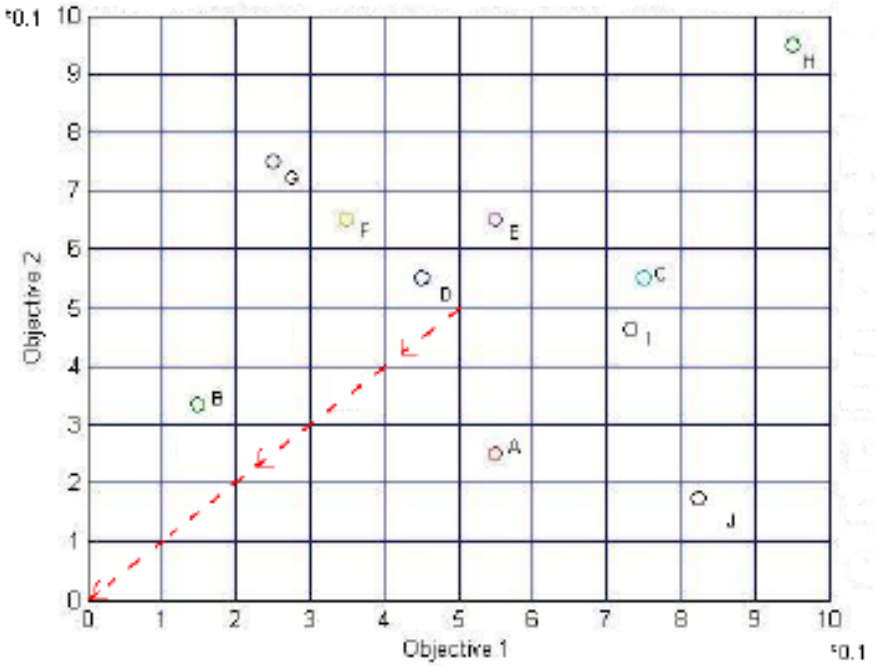


Fig. 2 Pre-selection for two objectives

Step 1	For m = 1 to M (*number_of_objectives)*
Step 2	(i) Evenly divide the range of the objective space into 100 bins.
Step 3	(ii) Scan P percentage of the 100 bins (*i.e.* from bin 1 to P, P may be chosen as 80 or 90).
Step 4	(iii) For each scanned bin (if this bin is empty, otherwise just continue to next bin), the solution with the smallest summation of normalized objective values will be chosen to enter preferential set.
Step 5	End For
Step 6	Accumulate the solutions excluded from the preferential set and store them in backup set.

Ensemble Methods for Multi-objective Optimization

Techbycheff Approach in MOEA/D

In this approach, the scalar optimization problems are in the form:

$$\begin{aligned} \text{minimize} \quad & g^{te}(x|\lambda, z^*) = \max_{1\leq i\leq m}\{\lambda_i|f_i(x) - z_i^*|\} \\ \text{subject to} \quad & x \in \Omega \end{aligned}$$

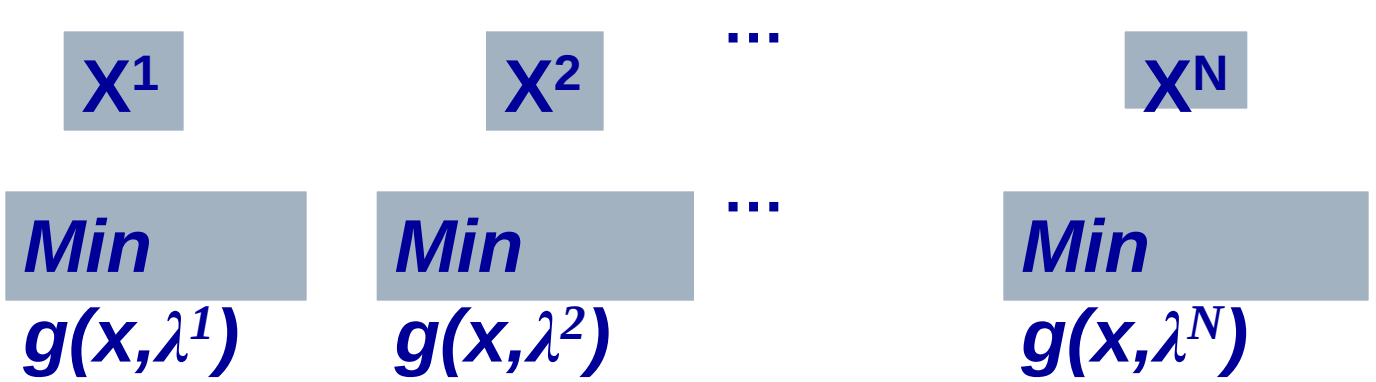
where m is the number of objectives. λ is a set of well-spread weight vectors ($\lambda^1 \ldots \lambda^N$) and z_i^* is the reference/best fitness of each objective. The problem of approximation of the PF is decomposed into N scalar optimization sub-problems. Solve these N sub-problems one by one. The distribution of final solutions could be uniform if g( ) and λ are properly chosen.

Neighborhood relationship in MOEA/D

- These sub-problems are related to each other.

  If λ^i and λ^j are close, $g(x,\lambda^i)$ and $g(x,\lambda^j)$ are neighbors. Neighboring problems should have similar solutions.

- N agents are used for solving these N sub-problems.



- During the search, neighboring agents can help each other.

MOEA/D framework

- Agent *i* records x^i, the best solution found so far for this particular sub-problem.
- At each generation, each agent *i* does the following:
- ✓ Select several neighbors and obtain their best solutions.
- ✓ Apply genetic operators (mutation & crossover in MOEA/D-DE) on these selected solutions and generate a new solution y'.
- ✓ Apply single objective local search on y' to optimize its objective $g(x,\lambda^i)$ and obtain y.
- ✓ Replace x^i by y if $g(y,\lambda^i)<g(x^i,\lambda^i)$.
- ✓ If not replaced, let one of its neighbors replace their best solutions by y if y is better than their current best solutions (measured by their individual objectives).

ENS-MOEA/D

- For effective performance of MOEA/D, neighborhood size (NS) parameter has to be tuned.

- A large NS promotes collaboration among dissimilar sub-problems, which advances the information exchange among the diverse sub-problems, and thus it speeds up the convergence of the whole population; while a small NS encourages combination of similar solutions and is good for local search in a local neighborhood, which maintains the diversity for the whole population.

- However, in some cases, a large NS can also benefit on diversity recovery; while a small NS is also able to facilitate the convergence.

- For instance, during the evolution, some sub-problems may get trapped in a locally optimal regions. In order to force those sub-problems escape from premature convergence, a large NS is required for the exploration. On the other hand, if the global optima area is already found, a small NS will be favorable for local exploitation.

ENS-MOEA/D

Neighborhood sizes (NS) are a crucial control parameter in MOEA/D.

An ensemble of different neighborhood sizes (NSs) with online self-adaptation is proposed (ENS-MOEA/D) to overcome the difficulties such as

1) **tuning the numerical values of NS for different problems;**

2) **specifications of appropriate NS over different evolution stages when solving a single problem.**

S. Z. Zhao, P. N. Suganthan, Q. Zhang, "Decomposition Based Multiobjective Evolutionary Algorithm with an Ensemble of Neighborhood Sizes", *IEEE Trans. on Evolutionary Computation,* **DOI:** 10.1109/TEVC.2011.2166159, June 2012.

ENS-MOEA/D

In ENS-MOEA/D, *K* fixed neighborhood sizes (NSs) are used as a pool of candidates. During the evolution, a neighborhood size will be chosen for each subproblem from the pool based on the candidates' previous performances of generating improved solutions. In ENS-MOEA/D, the certain fixed number of previous generations used to store the success probability is defined as the Learning Period (LP). At the generation G>LP-1 , the probability of choosing the kth (*k = 1, 2, …, K*) NS is updated by:

$$p_{k,G} = \frac{R_{k,G}}{\sum_{k=1}^{K} R_{k,G}}$$

$$R_{k,G} = \frac{\sum_{g=G-LP}^{G-1} FEs\_success_{k,g}}{\sum_{g=G-LP}^{G-1} FEs_{k,g}} + \varepsilon, \quad (k=1,2,..,K;\ G > LP)$$

ENS-MOEA/D Experimental results

- **ENS-MOEA/D is tested on the 10 unconstrained test instances in CEC 2009 MOEA Competition which includes two and three objective problems (Latest benchmark on MO problems).**

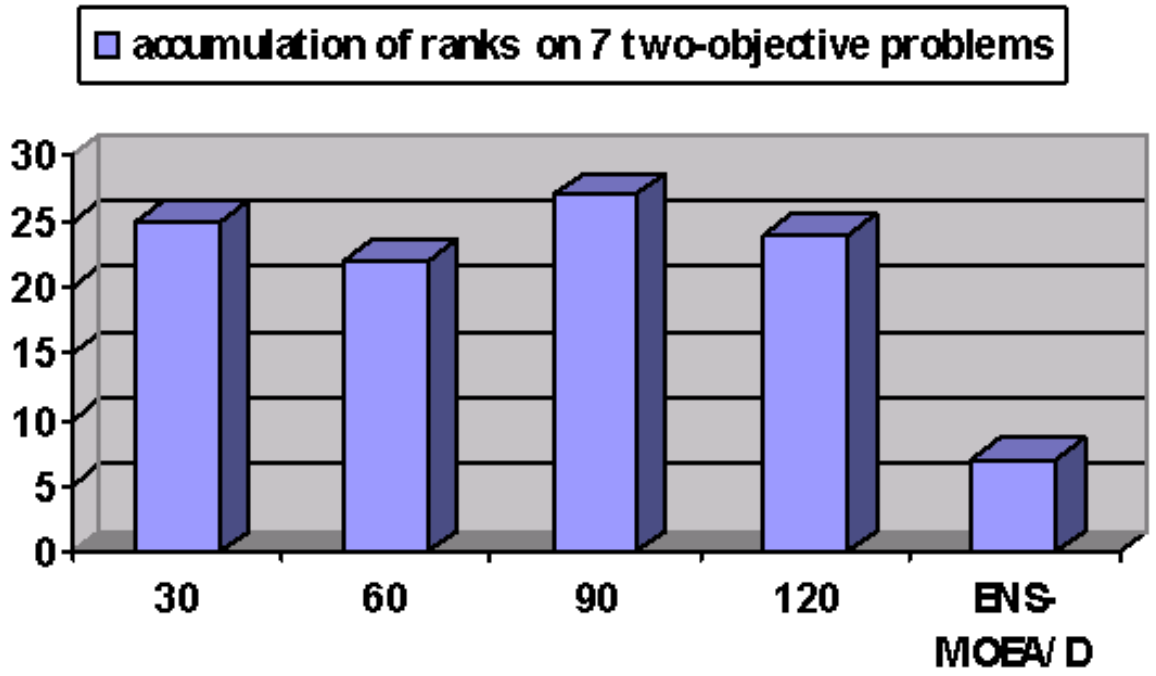
- **The IGD performance measure is used as in the CEC 2009 MOEA competition.**

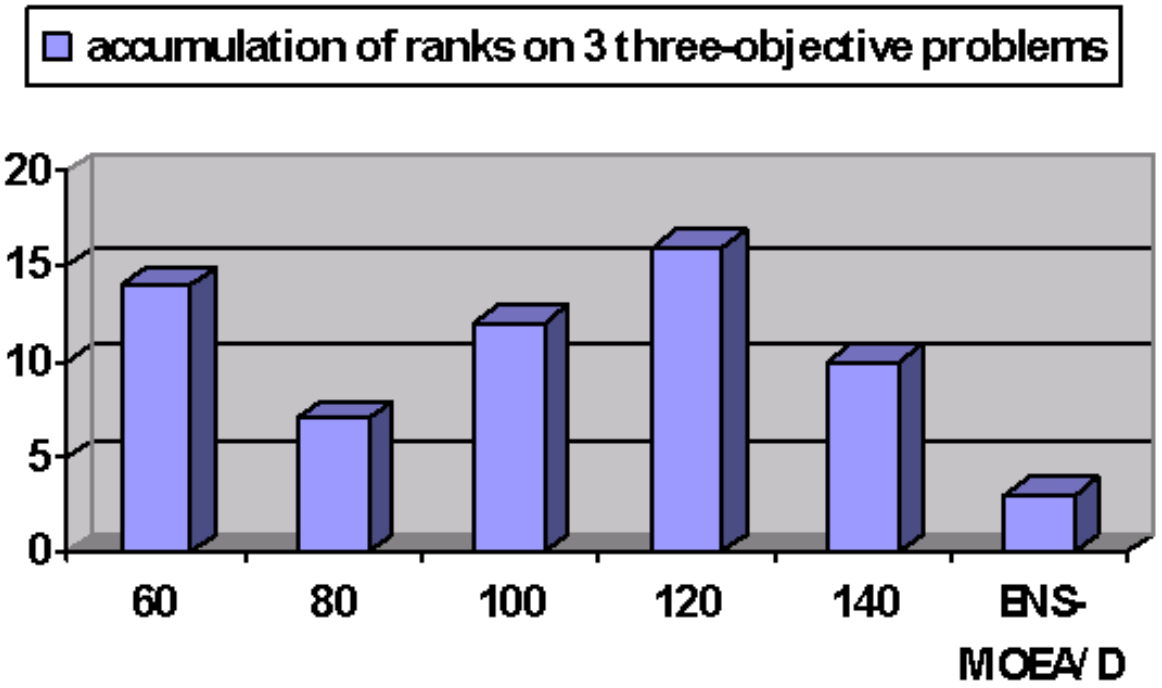
- **The four different NSs for the two-objective problems are 30, 60, 90 and 120, where NS=60 is the original parameter setting in the MOEA/D in the NSs for the three-objective problems are 60, 80, 100, 120 and 140, where 100 is the original parameter setting for NS in the MOEA/D.**

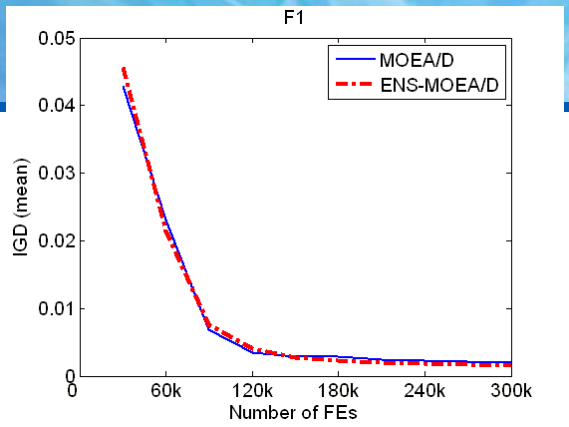
ENS-MOEA/D Experimental results

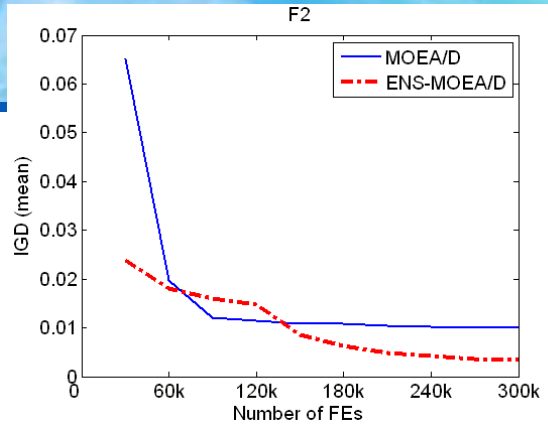
- **We conducted a parameter sensitivity investigation of *LP* for ENS-MOEA/D using four different values (10, 25, 50 and 75) on the 10 benchmark instances. By observing the mean of IGD values over 25 runs we can conclude that the *LP* is not so sensitive to most of the benchmark functions, and it is set as *LP*=50.**

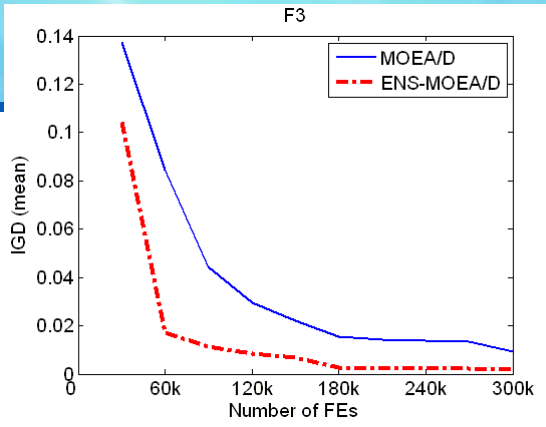
- **The mean of IGD values over 25 runs among all the variants of MOEA/D with different fixed *NS* and ENS-MOEA/D are ranked. Smaller the ranks, better the performance.**

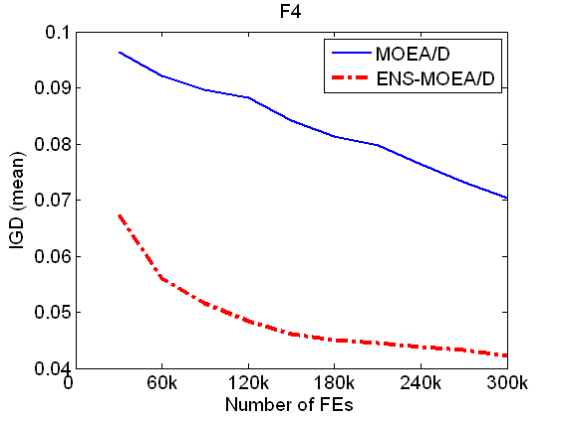


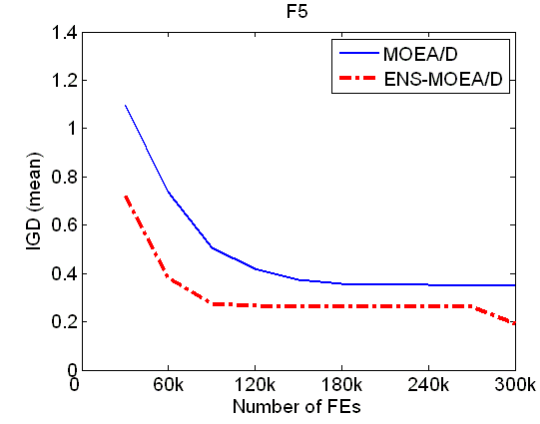


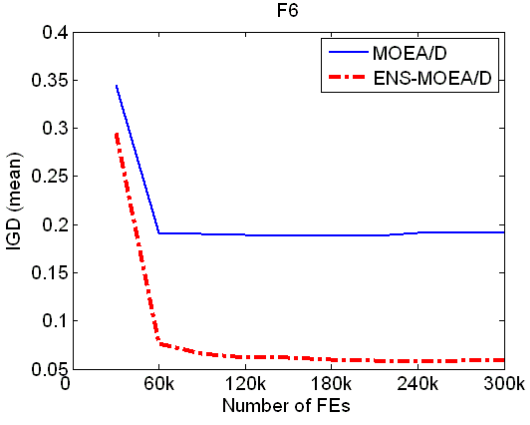

F1
MOEA/D
ENS-MOEA/D
IGD (mean)
Number of FEs
0.05
0.04
0.03
0.02
0.01
0
60k
120k
180k
240k
300k

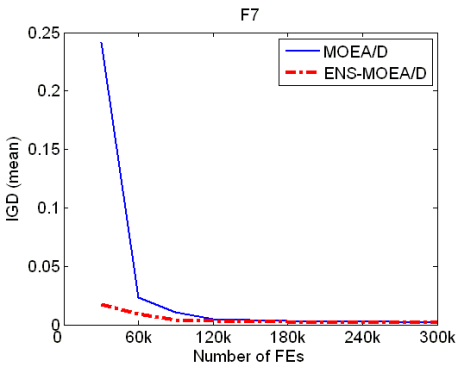

F2
MOEA/D
ENS-MOEA/D
IGD (mean)
Number of FEs
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
60k
120k
180k
240k
300k

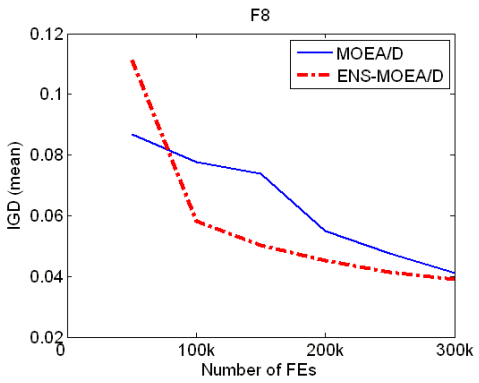

F3
MOEA/D
ENS-MOEA/D
IGD (mean)
Number of FEs
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
60k
120k
180k
240k
300k

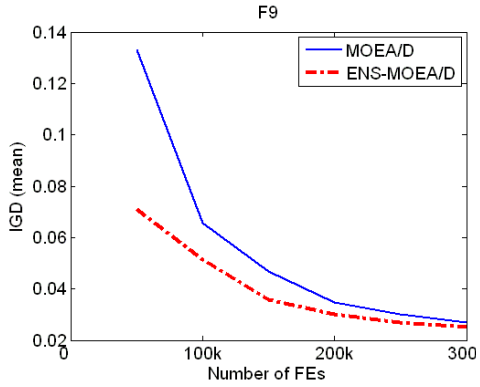

F4
MOEA/D
ENS-MOEA/D
IGD (mean)
Number of FEs
0.1
0.09
0.08
0.07
0.06
0.05
0.04
0
60k
120k
180k
240k
300k

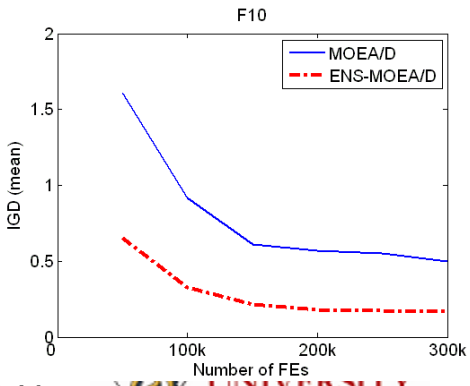

F5
MOEA/D
ENS-MOEA/D
IGD (mean)
Number of FEs
1.4
1.2
1
0.8
0.6
0.4
0.2
0
60k
120k
180k
240k
300k


F6
MOEA/D
ENS-MOEA/D
IGD (mean)
Number of FEs
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
60k
120k
180k
240k
300k


F7
MOEA/D
ENS-MOEA/D
IGD (mean)
Number of FEs
0.25
0.2
0.15
0.1
0.05
0
60k
120k
180k
240k
300k


F8
MOEA/D
ENS-MOEA/D
IGD (mean)
Number of FEs
0.12
0.1
0.08
0.06
0.04
0.02
0
100k
200k
300k


F9
MOEA/D
ENS-MOEA/D
IGD (mean)
Number of FEs
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
100k
200k
300


F10
MOEA/D
ENS-MOEA/D
IGD (mean)
Number of FEs
2
1.5
1
0.5
0
100k
200k
300k

Large Scale Global Optimization

Currently DE Variants with local Search are competitive.

CEC'08 Special Session / Competition on large scale single objective global optimization with bound constraints

CEC10 Special Session / Competition on large-scale single objective global optimization with bound constraints

Soft Computing J's Special Issue:
http://sci2s.ugr.es/eamhco/CFP.php

Large Scale Optimization

- Optimization algorithms perform differently when solving different optimization problems due to their distinct characteristics. But, some of them often lose their efficacy when solving complex problem instances with high dimensions.
- There are two main difficulties for those optimization methods whose performance deteriorates quickly as the dimensionality of the search space increases.
- The high demand on exploration capabilities of the optimization methods. When the solution space of a problem increases exponentially with the number of problem dimension, more efficient search strategies are required to explore all promising regions within a given time budget.

Large Scale Optimization

- **The complexity of a problem characteristics may increase with increasing dimensionality, e.g. uni-modality in lower dimensions may become multi-modality in higher dimensions for some problems.**
- **Due to these reasons, a successful search strategy in lower dimensions may no longer be capable of finding the optimal solution in higher dimension.**
- **In this work, three LSO algorithms with the best performance are presented –** MOS, jDElscop and SaDe-MMTS **(In the special issue of the Soft Computing Journal on Scalability of Evolutionary Algorithms and other Metaheuristics for Large Scale Continuous Optimization Problems).**

Large Scale Optimization

- We consider problems for which these 3 algorithms performed the same based on the final function value.
- We attempt to differentiate these algorithms based on their performance over evolution process, i.e. convergence curve.

Statistical Tests

- **In recent years, the use of statistical tests to improve the evaluation process of the performance of methods in computational intelligence**
- **Usually, statistical tests are employed into any experimental analysis to decide whether a method offers a significant improvement, or not, over the other methods on a given problem.**
- **Statistical tests can be categorized into two classes: parametric and nonparametric, depending on the characteristics of data.**

J. Derrac, S. García, D. Molina, F. Herrera, "**A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms**", *Swarm and Evolutionary Computation*, Vol. 1, No. 1, pp. 3-18, Mar 2011.

NANYANG TECHNOLOGICAL UNIVERSITY

Parametric vs. nonparametric

- **Although parametric tests could be used in the analysis of experiments in evolutionary computation, the lack of the required properties such as independence, normality, and homoscedasticity, normally yields to the nonparametric tests, especially in multi-problem analysis.**
- **Even though, value-to-reach criterion was used, it suffers from several shortcomings:**

(1) complexity of each problem can be different in a benchmark with several problems thereby requiring different value-to-reach targets;

(2) if we wish to compare convergence performance by using value-to-reach, we require several value-to-reach targets thereby further complicating the issue pointed out in (1);

(3) when several algorithms fail to reach a particular value-to-reach target within the maximum permitted function evaluations, comparing these algorithms will be impossible by using value-to-reach method.

Nonparametric test

- In this paper, one of the most commonly used nonparametric procedures, Friedman ranks test is illustrated and applied for the comparison multiple DE algorithms over multiple test problems and multiple evolution stages.
- The first step in calculating the test statistic is to convert the original results to ranks. They are computed using the following procedure:

1. Gather observed results for each algorithm/problem pair.
2. For each problem *i*, rank values from 1 (best result) to *k* (worst result). Denote these ranks as r_i^j ($1 \le j \le k$).
3. For each algorithm *j*, average the ranks obtained in all problems to obtain the final rank, $R_j = \frac{1}{n}\sum_i r_i^j$.

It ranks the algorithms for each problem separately; the best performing algorithm should have the rank of 1, the second best rank 2, etc. Again, in case of ties, we recommend computing average ranks. Under the null hypothesis, which states that all the algorithms behave similarly (therefore their ranks *Rj* should be equal)

NANYANG TECHNOLOGICAL UNIVERSITY

Friedman ranks test

- **However, a drawback of the above ranking scheme employed by the Friedman test is that it allows for comparison of one level (i.e. just at one point along the convergence plot) at a time only. When the number of algorithms for comparison is small, this may pose a disadvantage, since one level comparison may not be meaningful.**

- **Instead of that, a multi-point comparison over convergence plot is more reliable for comparing multiple algorithms. In order to overcome this shortage, in the work, an improved version of basic Friedman test, namely Friedman aligned ranks is used in our experimental analysis.**

S-Z. Zhao, P. N. Suganthan, "Comprehensive Comparison of Convergence Performance of Optimization Algorithms based on Nonparametric Statistical Tests", *IEEE Congress on Evolutionary Computation*, Brisbane, Australia, June 2012.

Friedman aligned ranks test

- A value of location is computed as the average performance achieved by all algorithms in each problem. Then, the difference between the performance obtained by an algorithm and the value of location is obtained. This step is repeated for each combination of algorithms and problems. The resulting differences (aligned observations), which keep their identities with respect to the problem and the combination of algorithms to which they belong, are then ranked from 1 to $k \cdot n$ relative to each other.
- This ranking scheme is the same as that employed by a multiple comparison procedure which employs independent samples, such as the Kruskal–Wallis test. The ranks assigned to the aligned observations are called aligned ranks.

Ensemble of Parameters & Strategies for DE

- As the performance of DE is sensitive to the choice of the mutation strategies and crossover operators as well as their associated control parameters, in order to obtain optimal performance, time consuming parameter tuning is commonly conducted to the conventional DE. Different mutation and crossover strategies with different parameter settings can be appropriate during different stages of the evolution. Furthermore, different optimization problems require different mutation strategies with different parameter values depending on the nature of problem and available computation resources. Motivated by these observations, a DE with an ensemble of mutation and crossover strategies and their associated control parameters was proposed as EPSDE.
- A pool of distinct mutation and crossover strategies along with a pool of values for each control parameter coexists throughout the evolution process and competes to produce offspring. The mutation strategies or the parameters present in a pool have diverse characteristics, so that they can exhibit distinct performance characteristics during different instances / points of the evolution, when dealing with a particular problem.

Sa-EPSDE-MMTS

- However, the candidate pool of mutation and crossover strategies and parameters in EPSDE is only restrictive to avoid the unfavorable influences of less effective mutation strategies and parameters. In other word, if the target vector is better than the trial vector, then the target vector is randomly reinitialized with a new mutation strategy and associated parameter values from the respective pools or from the successful combinations stored with equal probability. Instead of reinitialization of new mutation strategy and associated parameter values, the allocation of available computation resources of them is adapted based on their previous recorded performance proportionally (as in the Self-adaptive framework in SaDE-MMTS) to yield a new algorithm, Sa-EPSDE-MMTS. (MMTS- Modified multi-trajectory search)
- The original MTS employs search agents, which search for better solutions by moving with different step sizes in the parameter space from the original positions in each dimension. The step size was defined within a search domain to fit the requirement of global and local search.

NANYANG TECHNOLOGICAL UNIVERSITY

Low Level Self-adaptation in Sa-EPSDE

- **Sa-EPSDE benefits from the self-adaptation schemes of the mutation and crossover strategies and parameters in the candidate pool by learning from their previous experiences to fit different characteristic of the problems and different search requirements of evolution phases.**

- **Consequently, suitable offspring generation strategy along with associated parameter settings will be determined adaptively to match different phases of the search process more efficiently than in EPSDE.**

S. Z. Zhao and P. N. Suganthan, S. Das, "Self-adaptive Differential Evolution with Multi-trajectory Search for Large Scale Optimization," *Soft Computing,* **DOI:** 10.1007/s00500-010-0645-4., Nov 2011.

NANYANG TECHNOLOGICAL UNIVERSITY

Low Level Self-adaptation in MMTS

- **An adaptation approach is proposed to adaptively determine the initial step size parameter used in the MMTS. In each MMTS phase, the average of all mutual dimension-wise distances between current population members (AveDis) is calculated, one of the five linearly reducing factors (LRF) from 1 to 0.1, 5 to 0.1, 10 to 0.1, 20 to 0.1 and 40 to 0.1 is selected base on the previous experiences , and this LRF is applied to scale AveDis over the evolution.**

- **After that, the step size will be further reduced when a better solution is found along a particular dimension.**

High Level Self-adaptation between Sa-EPSDE & MMTS

- **The MMTS is used periodically for a certain number of function evaluations during conducting Sa-EPSDE, which is determined by an adaptation way.**

- **At the beginning of optimization procedure, the Sa-EPSDE and the MMTS are firstly conducted sequentially within one search cycle by using same number of function evaluations. Then the success rates of both Sa-EPSDE and MMTS are calculated. Subsequently, function evaluations are assigned to Sa-EPSDE and MMTS in each search cycle proportional to the success rates of both search methods.**

Experimental setting

- **Some of the test functions used in the Special Issue of Soft Computing on "Scalability of Evolutionary Algorithms and other Metaheuristics for Large Scale Continuous Optimization Problems" are considered in our simulation. This is a summary of the benchmark problems:**
- *• Shifted Unimodal Functions:*
- *– F1: Shifted Sphere Function*
- *– F2: Shifted Schwefel's Problem 2.21*
- *• Shifted Multimodal Functions:*
- *– F3: Shifted Rosenbrock's Function*
- *– F4: Shifted Rastrigin's Function*
- *– F5: Shifted Griewank's Function*
- *– F6: Shifted Ackley's Function*
- *• Shifted Unimodal Functions*
- *– F7: Shifted Schwefel's Problem 2.22*
- *– F8: Shifted Schwefel's Problem 1.2*
- *– F9: Shifted Extended f10*
- *– F10: Shifted Bohachevsky*
- *– F11: Shifted Schaffer*
- **The hybrid composition functions F12-F19 are obtained combining two functions from the set F1-F11.**

NANYANG TECHNOLOGICAL UNIVERSITY

Experimental setting

- **Four algorithms are used in the experiments.**
- ***A hybrid memetic algorithm based on the Multiple Offspring (MOS) framework***
- ***Self-adaptive DE is enhanced by incorporating the JADE mutation strategy and modified multi-trajectory search (MMTS) algorithm (SaDE-MMTS)***
- ***Self-adaptive DE algorithm, called jDElscop, for solving large-scale optimization problems with continuous variables***
- **All parameter settings are kept as the same in their original papers. For algorithms MOS, SaDE-MMTS and jDElscop, the programs used are downloaded from the open source given in. The parameter settings in proposed Sa-EPSDE-MMTS are referred to the two original works, SaDE-MMTS and EPSDE.**
- **Experiments were conducted on the minimization problems with 50 and 100 dimensions. To solve these problems, the maximum number of fitness evaluations is 5,000·D for all the methods. Each run stops when the maximal number of evaluations is exhausted.**

Experimental setting

- In our statistical analysis, the results of all algorithms are recorded 10 times along the evolution process / convergence plot, i.e. objective values at 10%, 20%,…, 100% of maximum number of fitness evaluations are stored and these values of all algorithms over all 25 runs are used in the multiple comparisons by Friedman Aligned Ranks test with ranks 1 to 1000 ($10 \times 4 \times 25$).

- In this configuration, the comprehensive convergence performance with the solution quality is considered in the comparison during the evolution for all algorithms, without any specific defined comparison standards (such as scaled multiple Value-to-Reach numerical values for multiple problems with diverse complexity). Smaller accumulated ranks in this analysis, means better performance.

NANYANG TECHNOLOGICAL UNIVERSITY

Experimental Results

- **In this work, our objective is to analyze the actual performance differences among all algorithms yielding statistically similar final solutions. Hence, our experiment only focuses on the problems yielding the same performance for these four algorithms. Therefore, the nine problems considered here are F1-F7, F12 and F16.**

- **In following Tables, the rank values for each problems achieved by Friedman Aligned Ranks test on the error values obtained by all algorithms over 25 runs are presented, best rank values are shown in bold.**

50D test problems

	MOS	jDElscop	SaDE-MMTS	Sa-EPSDE-MMTS
F1	**103250**	167561.5	126438.5	**103250**
F2	98625.5	199580.5	105956.5	**96337.5**
F3	130514.5	180582	119563.5	**69840**
F4	**95525**	167824	128004	109147
F5	**110610**	145514	119254	125122
F6	**94086**	192866.5	114606	98941.5
F7	**70894.5**	144019.5	135895.5	149690.5
F12	119845	157183	**110258**	113214
F16	130035	193173	**86510.5**	90781.5
Average	**105931.7**	172033.7	116276.3	106258.2
Times of best rank	5	0	2	3

100D test problems

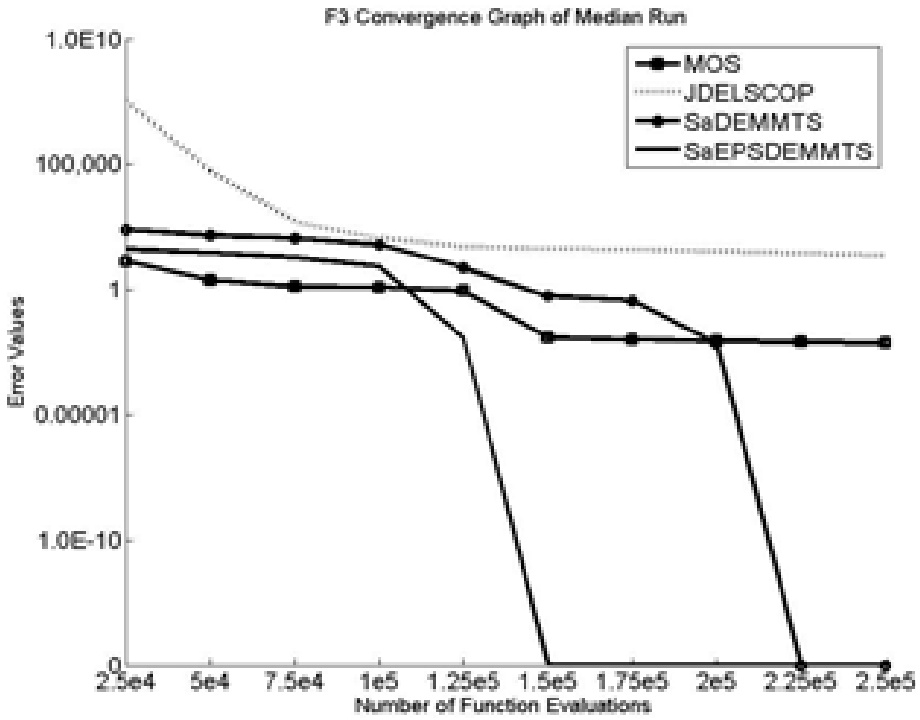
	MOS	jDElscop	SaDE-MMTS	Sa-EPSDE-MMTS
F1	**106250**	174602.5	113397.5	**106250**
F2	110028.5	213441	98314	**78716.5**
F3	129472	191372	114485	**65171**
F4	**109846**	146407.5	118461	125785.5
F5	**109784**	146428	123847	120441
F6	**87500**	217336	108164	**87500**
F7	**70054.5**	156082.5	135846.5	138516.5
F12	121116.5	146927.5	**113814.5**	118641.5
F16	130546	161682	**79812**	128460
Average	108288.6	172697.6	111793.5	**107720.2**
Times of best rank	5	0	2	4

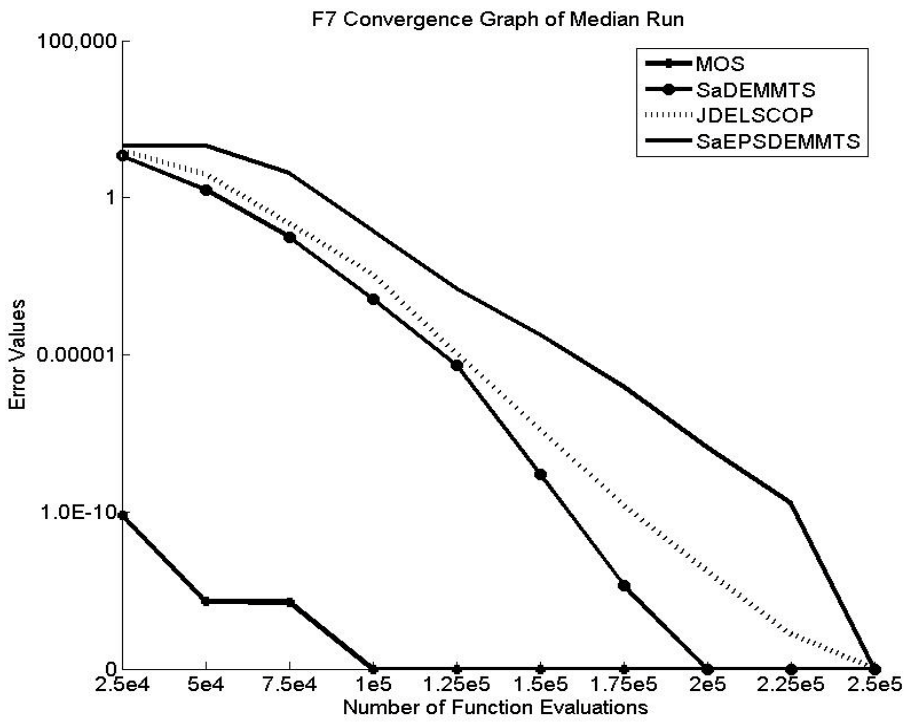
Analysis on Experimental Results

- Although all algorithms have the same final solutions, based on the nonparametric statistical test results in Tables, the rank values for each problem achieved by Friedman Aligned Ranks test on the objective values demonstrate the statistical difference among all algorithms.
- Therefore, the insight of actual performance differences among algorithms can be exposed by performing statistical comparisons over the convergence plot.
- Moreover, the average ranks of all problems also show the overall performance of the test problems among different algorithms. Hence, from Tables, it can be observed that MOS and Sa-EPSDE-MMTS have similar performance on all 50D and 100D test problems, while they are outperforming SaDE-MMTS and jDElscop.

Experimental Results

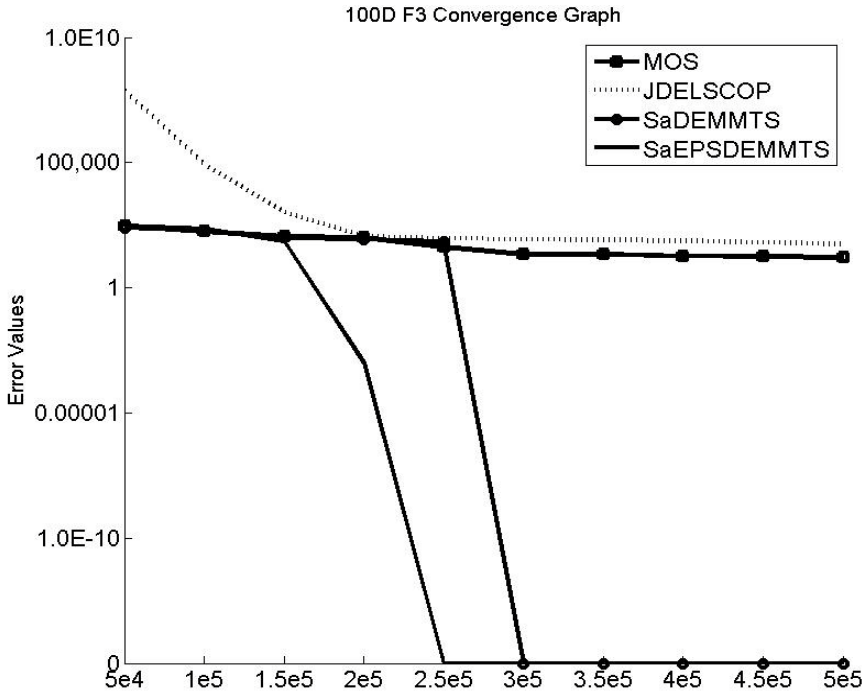
- **Figs. illustrate the convergence graph obtained by all four algorithms on 50D test problems F3 and F7.**

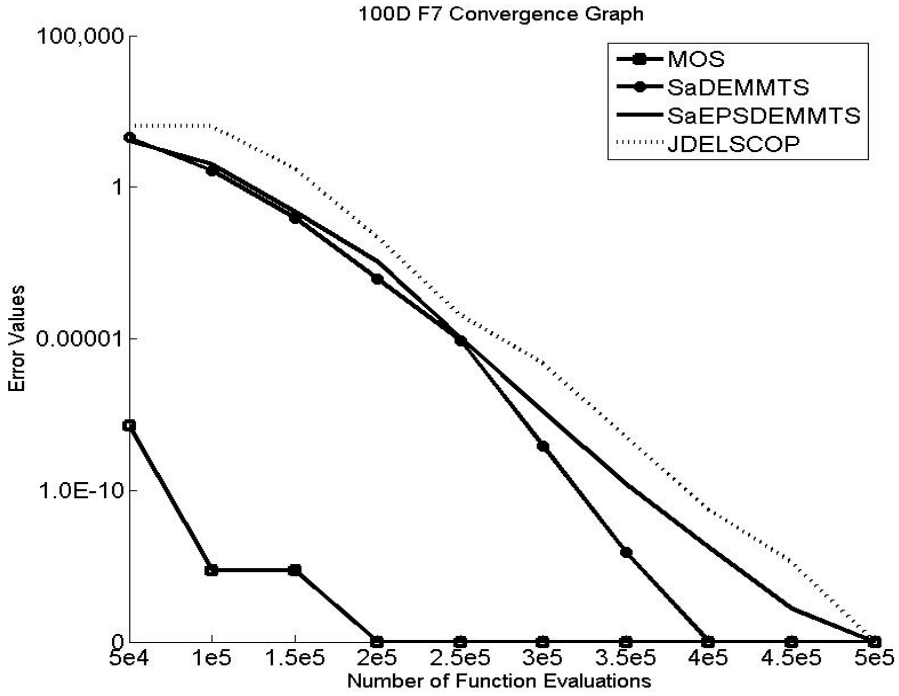




Experimental Results

- **Figs. illustrate the convergence graph obtained by all four algorithms on 100D test problems F3 and F7.**





Analysis on Experimental Results

- It can be concluded firstly that there is no statistical difference among the final solutions obtained by all algorithms in Figs. 2 and 4 as well as the small variations of the final solutions over all algorithms in Figs. 1 and 3.
- However, by looking at the whole convergence graph, all four algorithms indeed have statistically significant difference on convergence performance, which indicates the consistent observation with the results of Nonparametric Statistical Test in Tables.

Conclusions

- **Four state-of-the-art Differential Evolution (DE) based algorithms, MOS, SaDE-MMTS, jDElscop and Sa-EPSDE-MMTS, are tested on the most recent LSO benchmark problems and comparatively evaluated using nonparametric statistical analysis. Instead of using the "Value-to-Reach" as the comparison criterion, in our experimental analysis, multiple comparisons over multiple evolution points by using the Friedman Aligned Ranks test are investigated on those test problems in order to quantitatively compare convergence performance of different algorithms.**

- **Based the statistical results, among all the algorithms having the same final solutions, the different convergence characteristics of different algorithms are revealed quantitatively. It is recommended that the nonparametric statistical analysis by using Friedman Aligned Ranks test could be widely used in the performance evaluation of evolutionary computation algorithms to characterize convergence performance variations quantitatively.**

NANYANG TECHNOLOGICAL UNIVERSITY

Single Objective Multimodal Optimization

Currently DE & PSO with Neighborhood Search are Competitive.

S. Das, S. Maity, B-Y Qu, P. N. Suganthan, "**Real-parameter evolutionary multimodal optimization — A survey of the state-of-the-art**", Vol. 1, No. 2, pp. 71-88, *Swarm and Evolutionary Computation*, June 2011.

CEC10 Special Session on Niching Introduces novel scalable test problems:

B. Y. Qu and P. N. Suganthan, "Novel Multimodal Problems and Differential Evolution with Ensemble of Restricted Tournament Selection", *IEEE CEC*, Barcelona, Spain, July 2010

Multi-modal Optimization

- Aim: To find multiple global and local optima
- Evolutionary Algorithms vs. Classical Optimization Methods
- EAs converge to the global or a sub-optimal point
- Prevent convergence to a single solution and maintain multiple solutions – Niching (each desired solution is called a niche)

Multi-modal Optimization Methods

- Some Niching Techniques
  - Sharing
  - Clearing
  - Crowding
  - Restricted Tournament Selection
  - Clustering
  - Species Based
  - Neighborhood based DE

Multi-modal Optimization - Sharing

- Sharing
  - First among Niching Techniques
  - Proposed by Holland – Improved by Goldberg & Richardson
  - Population divided into subgroups based on similarity of individuals (σ - threshold of dissimilarity or niche radius)
  - Information sharing with other individuals in the same niche (Fitness sharing)

$$f_i' = \frac{f_i}{m}$$

m is niche count

  - Complexity – $O(NP)$

NP – population size

Multi-modal Optimization - Clearing

- Clearing
  - Retain the best members while eliminating the worst individuals of each niche
  - Complexity – $O(cNP)$

    NP – population size, c – number of subpopulations

  - Advantages
    - Lower Complexity
    - Significant reduction in genetic drift due to selection noise
    - Population can be much smaller

Multi-modal Optimization - Crowding

- Crowding
  - Proposed by De Jong
  - Newly generated individuals replace similar individuals
  - Similarity determined by distance metric
  - 2 parents randomly selected and produce 2 offspring by Mutation and crossover
  - Offspring replace nearest or similar parent if they are of greater fitness
  - Complexity $O(NP)$

Multi-modal Optimization - RTS

- Restricted Tournament Selection (RTS)
  - Proposed by Harick
  - Similar to Crowding
  - Corresponding to each offspring randomly choose *w* individuals from the population

  w – window size
  - From *w*, pick the nearest or similar individual to the offspring
  - Restricts competition with dissimilar individuals
  - Complexity – $O(NP * w)$
  - DE with ensemble of Restricted Tournament Selection

B. Y. Qu and P. N. Suganthan, "Novel multimodal Problems and Differential Evolution with Ensemble of Restricted Tournament Selection", IEEE Congress on Evolutionary Computation, pp. 1-7, Barcelona, Spain, July 2010.

Neighborhood Mutation Based DE

STEPS OF GENERATING OFFSPRING USING NEIGHBORHOOD MUTATION

Input	A population of solutions of current generation (current parents)
Step 1	For $i = 1$ to NP (population size)
	1.1 Calculate the Euclidean distances between individual i and other members in the population.
	1.2 Select m smallest Euclidean distance members to individual i and form a subpopulation ($subpop$) using these m members.
	1.3 Produce an offspring u_i using DE equations within $subpop_i$, i.e., pick r_1, r_2, r_3 from the subpopulation.
	2.3 Reset offspring u_i within the bounds if any of the dimensions exceed the bounds.
	2.4 Evaluate offspring u_i using the fitness function.
	Endfor
Step 2	Selection NP fitter solutions for next generation according to the strategies of different niching algorithm.
Output	A population of solutions for next generation

Compared with about 15 other algorithms on about 27 benchmark problems including recent IEEE TEC articles.

B-Y Qu, P N Suganthan, J J Liang, "Differential Evolution with Neighborhood Mutation for Multimodal Optimization," *IEEE Trans on Evolutionary Computation*, Doi: 10.1109/TEVC.2011.2161873, 2012.

Multi-modal Optimization

- Ensemble of Niching Algorithms (ENA)
  - 4 parallel Population s are used in this implementation
  - 2 populations use clearing method with 2 different clearing radii
  - 2 populations use RTS with 2 different window sizes.
  - Each popln generates its own offspring popln
  - All offspring are considered by all 4 poplns for inclusion.

E. L. Yu, P. N. Suganthan, "Ensemble of niching algorithms", *Information Sciences*, Vol. 180, No. 15, pp. 2815-2833, Aug. 2010.

Ensemble of niching algorithms

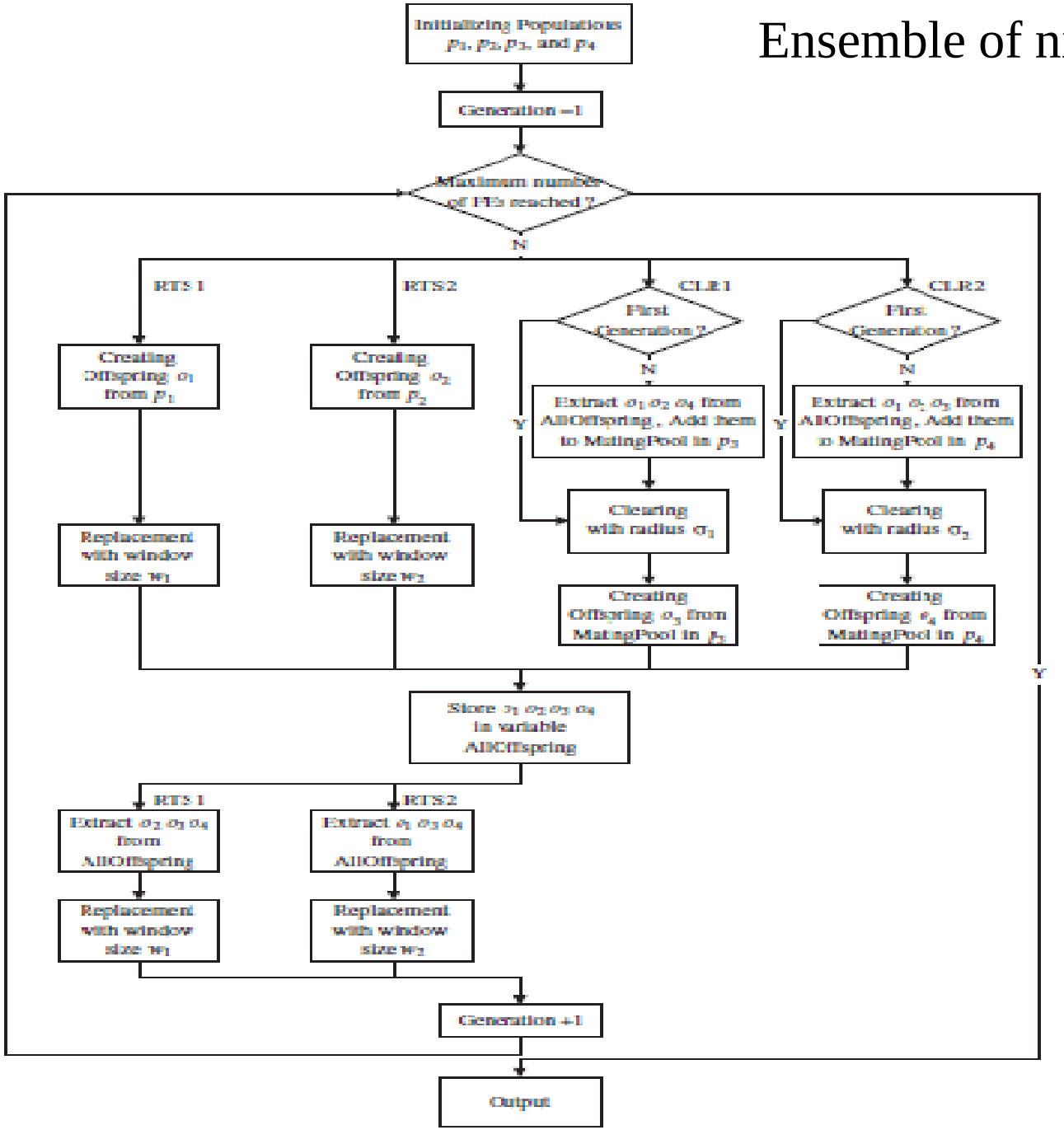


Fig. 3. Flowchart of the ensemble of niching algorithm.

NANYANG TECHNOLOGICAL UNIVERSITY

Multimodal Optimization by PSO

Disadvantages of classical niching PSO

1. **Niching parameters are difficult to specify**
2. **Topological based**
3. **Slow convergence**
4. **Low accuracy**

NANYANG TECHNOLOGICAL UNIVERSITY

Advantages of LIPS (Locally Informed Particle Swarm)

1. Benefit of FIPS velocity update equation to ensure good usage of all neighborhood information especially at the late stage of the search process which lead to fast convergence and high accuracy Local information from the nearest neighborhood are used to lead the particles

2. Euclidean distance based neighborhood selection to ensure the neighborhoods are from the same niches which increase the algorithm's local search and fine tuning ability

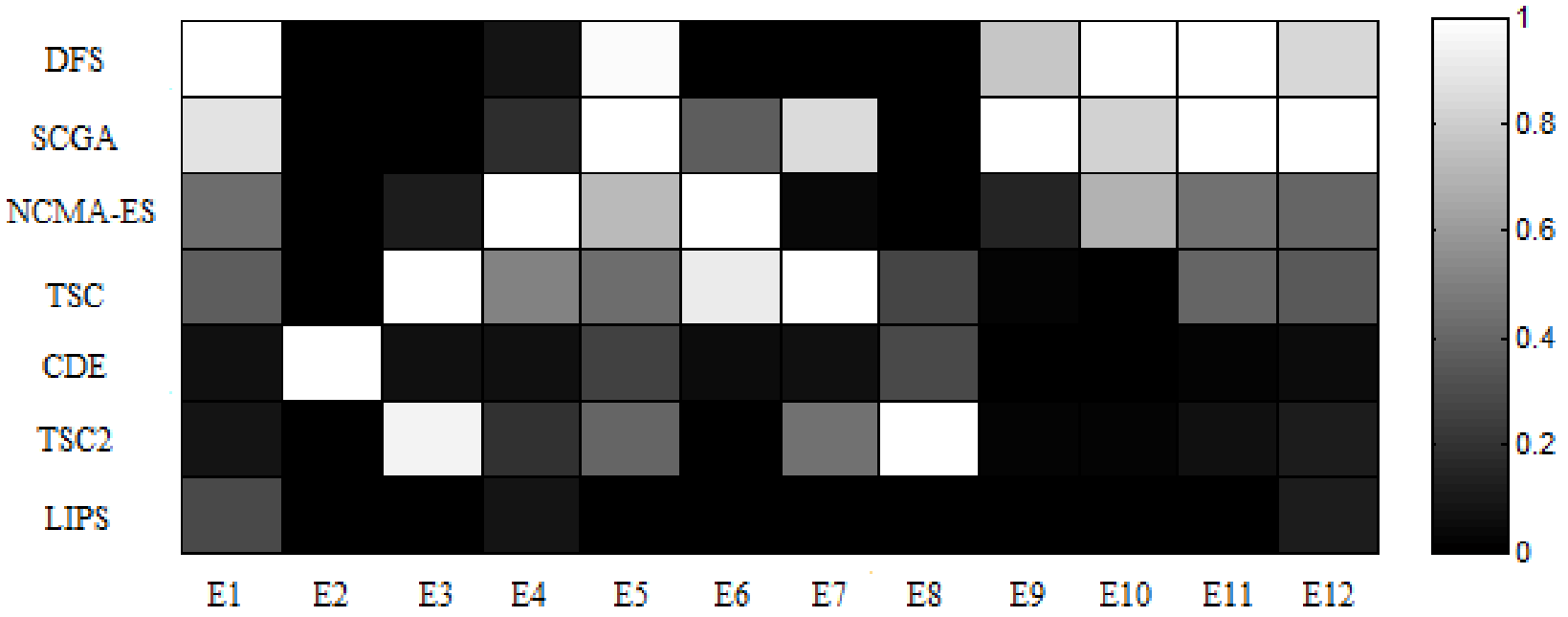
B-Y Qu, P. N. Suganthan, S. Das, "A Distance-Based Locally Informed Particle Swarm Model for Multi-modal Optimization,", accepted by *IEEE Trans on Evolutionary Computation,* **DOI:** 10.1109/TEVC.2012.2203138

NANYANG TECHNOLOGICAL UNIVERSITY

Steps of LIPS

Step 1	Randomly generate the initial solutions.
Step 2	Evaluate the initial solutions and initialize the *pbest*
	For *i=1* to *NP* (population size)
Step 2	Indentify the nearest *nsize* number neighborhood best to *ith* particle's *pbest*.
Step 3	Update the particles velocity using FIPS equation.
Step 4	Update the particles position.
Step 5	Evaluate the newly generated particle.
Step 6	Update the *pbest* for the *ith* particle.
	Endfor
Step 7	Stop if a termination criterion is satisfied. Otherwise go to Step 2.

Results



Overview of peak accuracies of each algorithm. Results are normalized for each problem, so that 0 refers to the best and 1 to the worst algorithm

Single Objective Dynamic Optimization

Currently DE & PSO with External Archives are Competitive.

Trung Thanh Nguyen, Shengxiang Yang, Juergen Branke,

Evolutionarydynamicoptimization: A survey of the state of the art,
Swarm and Evolutionary Computation, Volume 6, October 2012, Pages 1–24

CEC'09 Special Session / Competition on Dynamic Optimization

NANYANG TECHNOLOGICAL UNIVERSITY

Dynamic Optimization Problems (DOPs)

- Many real-life applications are transient in time. E.g. stock market, satellite array beam-forming, adaptive filter coefficients.
- Fitness evaluation in Dynamic Optimization Problems (DOPs) is subject to time

$$F = f(\bar{x}, t), \bar{x} = [x_1, x_2, \dots x_D]$$

- Population-based optimization looses population diversity as popln converges to global solution(s), but in DOPs, the true optima may shift after convergence.

Dynamic Optimization Problems (DOPs)

Techniques for Handling Dynamic objectives in Evolutionary Algorithms (EAs)

1. Generate Diversity after change – Re-evaluate some members for change after every generation, and reinitialize whole population upon change detection.
2. Maintain Diversity throughout search – Apply different updating strategies for individuals in population, e.g. quantum/Brownian individuals
3. Memory-based approaches – Archive past population information for future re-insertion (Useful for oscillating optima)
4. Multiple-population approaches – Conduct multiple concurrent searches using multiple sub-populations thereby maintaining diversity.

Dynamism Handling

1. Multi-population approaches –
   a) Multiple sub-populations
   b) Favored population: more individuals allocated to most promising sub-population (i.e. fittest local optima)
   c) Exclusion principle (exclude overlapping search by more than one sub-population)
2. Maintaining Diversity throughout run –
   a) Quantum Individuals
   b) Brownian Individuals
   c) Entropic Individuals
3. Memory-based approaches –
   a) Aged Individuals or stagnated individuals to be reinitialized
   b) Memory can be used to copy past good solutions in the memory back into the evolving population.

Ensemble Differential Evolution with Dynamic Subpopulations and Adaptive Clearing (EDESAC)

- Motivation
  - Diversity maintenance to prevent stagnation
  - Self-adaptation strategy for unpredictable dynamic change
  - Memory technique – good for recurrent /periodic changes
  - Greedy mutation strategy for exploitation
- Implementation
  - Multi-population approach using Ensemble of mutation, crossover strategies & parameter values (EPSDE)
  - Gbestpop with greedy mutation strategy grows over fixed FEs; shift from exploration to exploitation
  - Adaptive clearing of past good solutions and local optima in archive

S. Hui, P. N. Suganthan, "Ensemble Differential Evolution with Dynamic Subpopulations and Adaptive Clearing for solving Dynamic Optimization Problems", *IEEE Congress on Evolutionary Computation*, Brisbane, Australia, June 2012.

Ensemble Differential Evolution with Dynamic Subpopulations and Adaptive Clearing (EDESAC)

1) Total pop size (N) = 80, sub-pop size (L_size) = 10

2) Sub-pop EPSDE Mutation Strategies:

a) *DE/current-to-pbest/2*

$$\boldsymbol{v_i^G = x_i^G + (1 - F_i)(x_{pbest}^G - x_i^G) + F_i\,(x_{rand1,i}^G - x_{rand2,i}^G)}$$

b) DE/rand/2

$$\boldsymbol{v_i^G = x_{rand1,i}^G + F_i\,(x_{rand2,i}^G - x_{rand3,i}^G + x_{rand4,i}^G - x_{rand5,i}^G)}$$

F_i =[0.3, 0.7, 0.9], C_i =[0.1, 0.5, 0.9], CR methods = [*Bin*, *Exp*]

3) *Gbestpop* chosen at 30000 FEs (max FEs before change = 100000)

- Greedy tournament mutation strategy:

  $$\boldsymbol{v_i^G = x_{tourn1,i}^G + F_i\,(x_{tourn2,i}^G - x_{tourn3,i}^G)}$$

  where $f(\boldsymbol{x_{tourn1}}) > f(\boldsymbol{x_{tourn2}}) > f(\boldsymbol{x_{tourn3}})$ in maximization problem

- ***for k=1 to 3,*** (***r1, r2*** = random index without replacement)

  $$\boldsymbol{x_{tourn\ k}} \begin{cases} = \boldsymbol{x_{r1\ k}}, & \textbf{\textit{if}}\ f(\boldsymbol{x_{r1\ k}}) > f(\boldsymbol{x_{r2\ k}}) \\ = \boldsymbol{x_{r2\ k}}, & \textbf{\textit{otherwise}} \end{cases}$$

- *Gbestpop* size incremented every 3000 Fes

- Worst performing sub-pop eliminated when total pop size $\geq N + L\_size$

Ensemble Differential Evolution with Dynamic Subpopulations and Adaptive Clearing (EDESAC)

- Aging applied on local best in sub-pop to identify local optima

if $|f(u_i^G) - f(x_i^G)| < 0.1$ or $\| u_i^G - x_i^G \| < 0.01$

    $age_i = max(age_i, 20)$

else $age_i = min(age_i, 5)$ *end*

if $i = lbest$ & $age_i > 30$ & $i \neq gbest$

    $x_i^G \rightarrow archive$

    $f(x_i^G) \rightarrow archive\_fit$

end

- Adaptive clearing radius (CL_R) performed on archive

Rank archive $= [a_1, a_2 \ldots, a_P]$, *such that* $f(a_i) < f(a_{i+1})$, *for* $i = 1,2 \ldots P-1$

if $P < 5$ & *rand()*< 0.1, $CL\_R = 0.5$

else if $5 < P < 20$ & *rand*$() < 0.1$, $CL\_R = 0.1$

else $CL\_R = 1$

for $i = 1$ *to* $P-1$

    for $k = 2$ *to* P

    if $\| a_i - a_k \| < CL\_R$, *remove* a_k *from archive*

    end

end

- Archive used for reinitialization when change detected

Self-Adaptive DE (jDE)

- Multi-population approach with self-adapting parameter control
  - Aging of all individuals except global optima to escape local optima
  - Archiving of past global best individuals for future re-initialization
- Self-adaptive control parameters F and CR

$$F_i^{G+1} = \begin{cases} F_i^G, U(0,1) \geq \tau_1 \\ F_L + U(0,1).F_H, otherwise \end{cases}, F_L \leq F_i \leq F_H$$

$$CR_i^{G+1} = \begin{cases} CR_i^G, U(0,1) \geq \tau_2 \\ U(0,1), otherwise \end{cases}$$

where τ_1, τ_2 are probabilities for controlling F and CR

Also to maintain a degree of diversity, the following must be satisfied

$$2F^2 - \frac{2}{NP} + \frac{CR}{NP} = 0$$

[1] J. Brest, A. Zamuda, B. Boskovic, M. S. Maucec and V. Zumer, "Dynamic Optimization using Self-Adaptive Differential Evolution", *Proc. of the 2009 Congress on Evolutionary Computation*, pp. 415 – 422, 2009.

[2] D. Zaharie, "Critical Values for the Control Parameters of Differential Evolution Algorithms", Proc of Mendel 2002, 8th Int. Conf. on Soft Computing, 2002, pp. 62- 67.

Self-Adaptive DE (jDE)

- Aging strategy of individuals

$$\begin{gathered}\boldsymbol{if}\ x_i = g_{best},\\ age_i = 0\\ \boldsymbol{else\ if}\ (x_i = l_{best})\&(age_i > 30)\&(rand < 0.1),\\ reinitialize\ subpop\ that\ contains\ x_i\end{gathered}$$

- Overlapping search (Exclusion) – reinitialize sub-pop whose local best is too close to another local best
- Maintain local diversity – reinitialize individuals that are too close to loca best.
- Age of individual is incremented every generation. Fitness and diversity control:

$$\begin{gathered}\boldsymbol{if}\ \|x_i - x_j\| < 0.01\ \boldsymbol{OR}\ \|f(x_i) - f(x_j)\| < 0.1,\\ age_i = min\{age_i, 20\}\\ \boldsymbol{else}\ age_i = min\{age_i, 5\}\end{gathered}$$

- Archiving of past global best after every change detection. Whenever re-initialization occurs, individuals have chances to be either taken from archive or be randomly generated

Single Objective Constrained Optimization

Currently DE with local Search and Ensemble of Constraint handling are competitive.

CEC'06 Special Session / Competition on Evolutionary Constrained Real Parameter single objective optimization

CEC10 Special Session / Competition on Evolutionary Constrained Real Parameter single objective optimization

E Mezura-Montes, C. A. Coello Coello, "**Constraint-handling in nature-inspired numerical optimization: Past, present and future**", Vol. 1, No. 4, pp. 173-194, *Swarm and Evolutionary Computation*, Dec 2011.

Constraint Handling Methods [MS08b]

- Many optimization problems in science and engineering involve constraints. The presence of constraints reduces the feasible region and complicates the search process.
- Evolutionary algorithms (EAs) always perform unconstrained search.
- When solving constrained optimization problems, they require additional mechanisms to handle constraints [Coello02]

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, …)
  - …
- **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.

Superiority of Feasible [Deb00]

- In SF when two solutions *X*i and Xj are compared, *X*i is regarded superior to *X*j under the following conditions:
  1. *X*i is feasible and *X*j is not.
  2. *X*i and *X*j are both feasible and *X*i has smaller objective value (for minimization problems) than *X*j.
  3. *X*i and *X*j are both infeasible, but *X*i has a smaller overall constraint violation.

Superiority of Feasible (SF)

- Therefore, in SF feasible ones are always considered better than the infeasible ones.
- Two infeasible solutions are compared based on their overall constraint violations only, while two feasible solutions are compared based on their objective function values only.
- Comparison of infeasible solutions on the constraint violation aims to push the infeasible solutions to feasible region, while comparison of two feasible solutions on the objective value improves the overall solution.

Self-Adaptive Penalty (SP) [TY06]

- In self adaptive penalty function method two types of penalties are added to each infeasible individual to identify the best infeasible individuals in the current population.
- The amount of the added penalties is controlled by the number of feasible individuals currently present in the combined population.
- If there are a few feasible individuals, a higher amount of penalty is added to infeasible individuals with a higher amount of constraint violation.
- On the other hand, if there are several feasible individuals, then infeasible individuals with high fitness values will have small penalties added to their fitness values.
- These two penalties will allow the algorithm to switch between finding more feasible solutions and searching for the optimum solution at anytime during the search process.

Self-Adaptive Penalty (SP)

- The final fitness value based on which the population members are ranked is given as, $F(X)$ where $d(X)$ is the distance value and $p(X)$ is the penalty value

$$d(X) = \begin{cases} \upsilon(X), & \text{if } r_f = 0 \\ \sqrt{f''(X)^2 + \upsilon(X)^2}, & \text{otherwise} \end{cases}$$

$$r_f = \frac{\text{Number of feasible individuals}}{\text{population size}},$$

$\upsilon(X)$ is overall constraint violation

$$f''(X) = \frac{f(X) - f_{\min}}{f_{\max} - f_{\min}}$$

$f_{\min}$, $f_{\max}$ are the minimum and maximum of $f(X)$ irrespective of feasibility.

Self-Adaptive Penalty (SP)

- The penalty $p(X) = (1 - r_f)M(X) + r_f N(X)$

where

$$M(X) = \begin{cases} 0, & \text{if } r_f = 0 \\ \upsilon(X), & \text{otherwise} \end{cases}$$

$$N(X) = \begin{cases} 0, & \text{if } X \text{ is a feasible individual} \\ f''(X), & \text{if } X \text{ is an infeasible individual} \end{cases}$$

The solutions are ranked based on the following fitness function:

$F(X)=d(X) + p(X)$

Epsilon Constraint handling (EC) [TS06]

- The relaxation of the constraints is controlled by using the ε parameter
- High quality solutions for problems with equality constraints can be obtained by proper control of the ε parameter [TS06].
- The ε parameter is initialized using:

$$\varepsilon(0) = \upsilon(X_\theta)$$

where X_θ is the top θ th individual in the initialized population after sorting w. r. t constraint violation.

Epsilon Constraint Handling (EC)

$$\varepsilon(k) = \begin{cases} \varepsilon(0)\left(1 - \dfrac{k}{T_c}\right)^{cp}, & 0 < k < T_c \\ 0, & k \geq T_c \end{cases}$$

- The ε is updated until the generation counter *k* reaches the control generation T_c, after which ε is set to zero to obtain solutions with no constraint violation.
- The recommended parameter settings are

$$T_c \in [0.1T_{\max}, 0.8T_{\max}] \qquad cp \in [2, 10]$$

Stochastic Ranking (SR) [RY00]

- SR achieves a balance between objective and the overall constraint violation stochastically at a low computational cost [RY00]
- A probability factor p_f is used to determine whether the objective function value or the constraint violation value determines the rank of each individual

Stochastic Ranking (SR)

- Basic form of the SR can be represented as

If (no constraint violation or rand < p_f)

Rank based on the objective value only

else

Rank based on the constraint violation only

End

Ensemble of Constraint Handling Methods

- According to the no free lunch theorem (NFL) [WM97], no single state-of-the-art constraint handling technique can outperform all others [MS96] [WCGZ07] on every problem.

- Each constrained problem would be unique in terms of the ratio between feasible search space and the whole search space, multi-modality and the nature of constraint functions.

- Evolutionary algorithms are stochastic in nature. Hence the evolution paths can be different in every run even when the same problem is solved by using the same algorithm implying different CH can be efficient in different runs.

R. Mallipeddi, P. N. Suganthan, "Ensemble of Constraint Handling Techniques", *IEEE Trans. on Evolutionary Computation,* Vol. 14, No. 4, pp. 561 - 579 , Aug. 2010

NANYANG TECHNOLOGICAL UNIVERSITY

Ensemble of Constraint Handling Methods: MOTIVATION

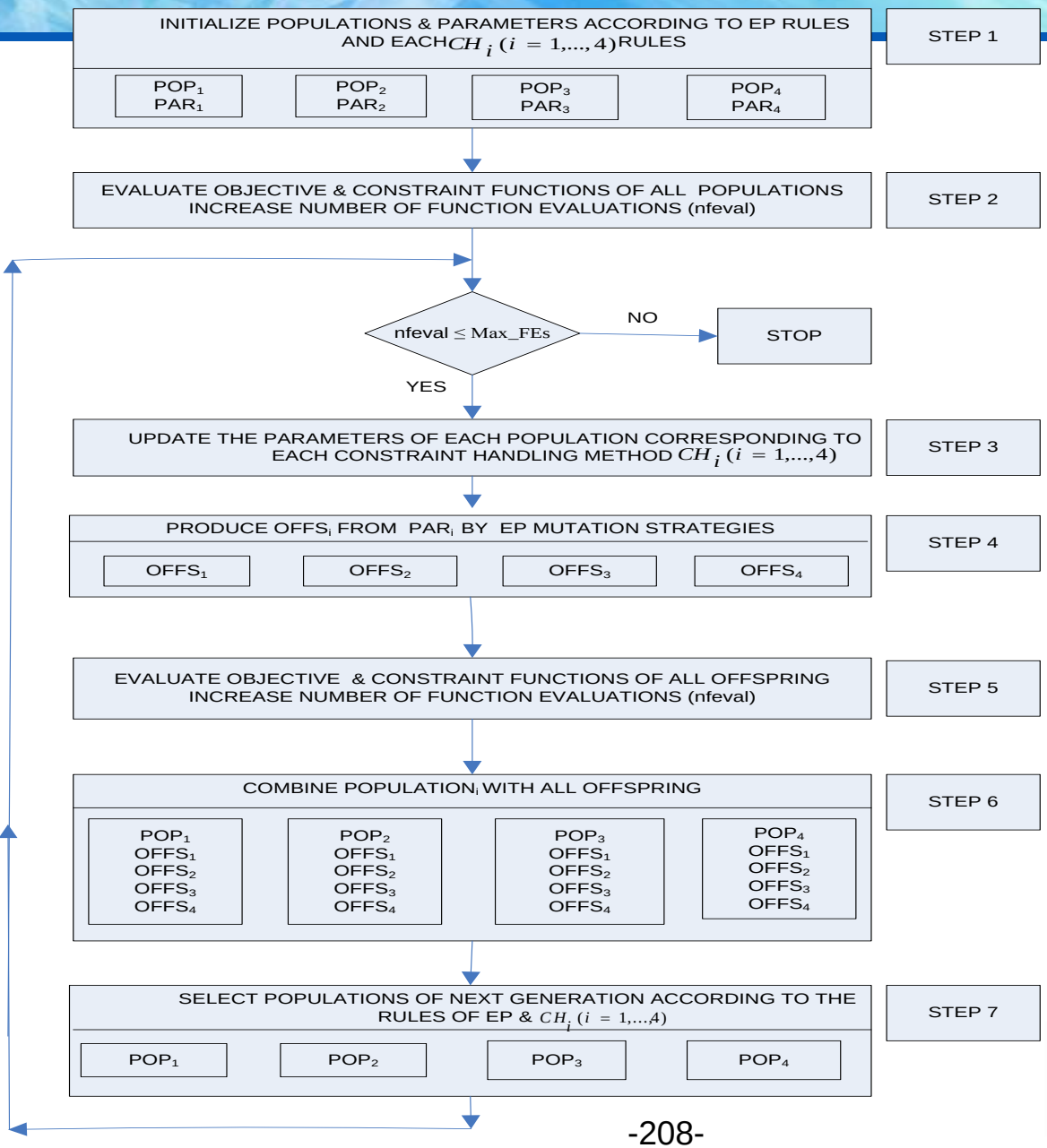
- Therefore, depending on several factors such as the ratio between feasible search space and the whole search space, multi-modality of the problem, nature of equality / inequality constraints, the chosen EA, global exploration/local exploitation stages of the search process, different constraint handling methods can be effective during different stages of the search process.

- Hence, solving a particular constrained problem requires numerous trial-and-error runs to choose a suitable constraint handling technique and to fine tune the associated parameters. Even after this, the NFL theorem says that one well tuned method may not be able to solve all problems instances satisfactorily.

ECHT [MS08b]

- Each constraint handling technique has its own population and parameters.
- Each population corresponding to a constraint handling method produces its offspring.
- The parent population corresponding to a particular constraint handling method not only competes with its own offspring population but also with offspring population of the other constraint handling methods.
- Due to this, an offspring produced by a particular constraint handling method may be rejected by its own population, but could be accepted by the populations of other constraint handling methods.

Ensemble of constraints Handling Techniques (ECHT)



NANYANG TECHNOLOGICAL UNIVERSITY

ECHT

- Hence, in ECHT every function call is utilized effectively. If the evaluation of objective / constraint functions is computationally expensive, more constraint handling methods can be included in the ensemble to benefit more from each function call.
- And if a particular constraint handling technique is best suited for the search method and the problem during a point in the search process, the offspring population produced by the population of that constraint handling method dominates the other and enters other populations too.
- In the subsequent generations, these superior offspring will become parents in other populations too.

ECHT

- Therefore, ECHT transforms the burden of choosing a particular constraint handling technique and tuning the associated parameter values for a particular problem into an advantage.
- If the constraint handling methods selected to form an ensemble are similar in nature then the populations associated with each of them may lose diversity and the search ability of ECHT may be deteriorated.
- Thus the performance of ECHT can be improved by selecting diverse and competitive constraint handling methods.

ECHT

- The constraint handling methods used in the ensemble are

1. Superiority of Feasible (SF) [Deb00]
2. Self-Adaptive penalty (SP) [TY06]
3. Stochastic Ranking (SR) [RY00]
4. Epsilon Constraint handling (EC) [TS06]

Test Functions and Parameter Settings

- 24 well known benchmark functions of CEC2006 are used [LRMCSCD06].
- $T_c = 0.5T_{\max}$ and $c_p = 5$.
- Population size of single constraint handling method is 200.
- In ECHT the population size of each constraint handling method is 50.
- Each algorithm is run 30 times.
- Total function evaluations given are 240,000.
- ECHT1 uses the same parameters as in single constraint handling method.
- ECHT2 uses tuned parameters $T_c = 0.8T_{\max}$ and $c_p = 2$

NANYANG TECHNOLOGICAL UNIVERSITY

RESULTS

Comparison of single constraint handling methods and ECHT where ( ) contains ranking

Fcn & Optimal Value		SF	SP	EC	SR	ECHT1	ECHT2
G05 5126.4967	best	5126.4969	5126.4967	5126.4967	5126.4969	5126.4967	5126.4967
	Md	5131.7147	5126.5211	5126.4968	5130.0131	5126.4967	5126.4967
	Mn	5161.5388(5)	5127.7182(3)	5126.5058(2)	5158.3317(4)	5126.4967(1)	5126.4967(1)
	worst	5485.1800	5134.6751	5126.6048	5329.3866	5126.4972	5126.4967
	sd	7.8E+01	2.3E+00	2.2E-02	6.0E+01	0.0E+00	0.0E+00
	h	1	1	1	1	-	-
G10 7049.2480	b	7054.1548	7049.2566	7051.2552	7055.6236	7049.2487	7049.2483
	Md	7118.6628	7050.4251	7096.0003	7138.4684	7049.3456	7049.2488
	Mn	7123.6326(5)	7051.6584(3)	7094.3033(4)	7155.6031(6)	7049.4342(2)	7049.2490(1)
	w	7242.9482	7080.8116	7213.3871	7428.5722	7050.3902	7049.2501
	sd	5.3E+01	5.7E+00	3.75E+01	80.E+01	2.00E-01	6.60E-04
	h	1	1	1	1	-	-
G21 193.7245	b	193.7491	195.3702	193.7290	193.7300	193.7245	193.7246
	Md	198.7857	232.4343	240.1518	193.7486	196.3287	193.7399
	Mn	225.4789(4)	241.7966(6)	234.7433(5)	206.1255(2)	215.4050(3)	193.7438(1)
	w	294.1277	330.4669	275.8960	261.9906	273.1474	193.7741
	sd	4.10E+01	3.9E+01	2.66E+01	2.63E+01	3.02E+01	1.65E-02
	h	0	0	0	0	-	-
RANK		57	60	54	68	36	22

Conclusions

- The overall total ranks for SF, SP, EC, SR, ECHT1 and ECHT2 are 57, 60, 54, 60, 36 and 22 respectively.
- The minimum rank that any algorithm could achieve by becoming the best in all problems is 22.
- The average ranks for SF, SP, EC, SR, ECHT1 and ECHT2 are 2.59, 2.73, 2.45, 3.09, 1.64 and 1.00 respectively.
- From the ranking results, it is obvious that the performance of single constraint handling methods vary substantially and no single constraint handling method is superior to other single constraint handling methods in every problem as their ranks are low and approximately similar.

Conclusions

- To statistically compare the performance of ensemble with the four single constraint handling methods, *t*-test results (*h* values) are presented, where each row represents the *t*-test results for a particular problem for single constraint handling methods against the ensemble (ECHT1).
- To allow fair comparison, we compare the single cases with only ECHT1 which uses the same parameters as that of the single cases.
- Numerical values -1, 0, 1 represent that the ECHT1 is inferior to, equal to and superior to the SF, SP, EC and SR constraint handling method, respectively.
- From the *t*-test results, we can observe that single constraint handling methods are superior to, equal to and worse than the ECHT1 in 0, 63 and 25 cases respectively out of the total 88 cases.
- Thus, the ECHT1 is always either better or equal compared to the EP with a single CH method.

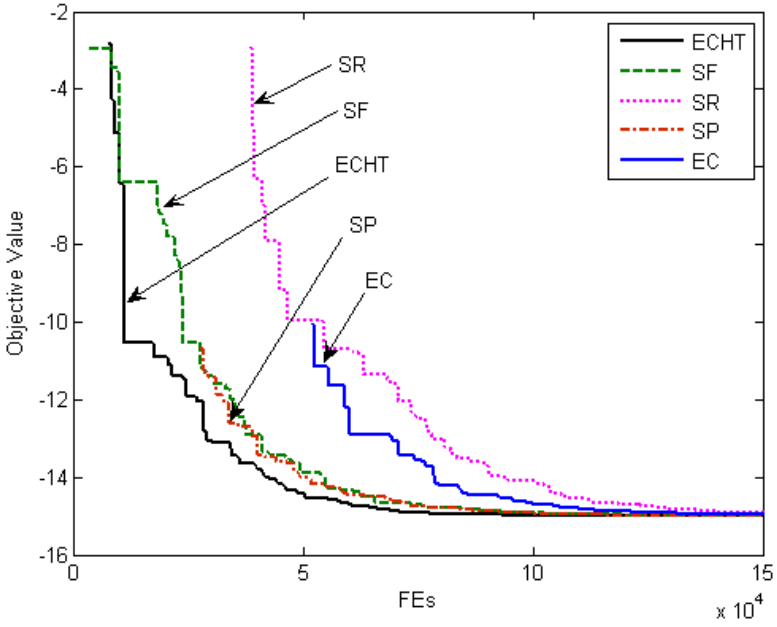
NANYANG TECHNOLOGICAL UNIVERSITY

RESULTS

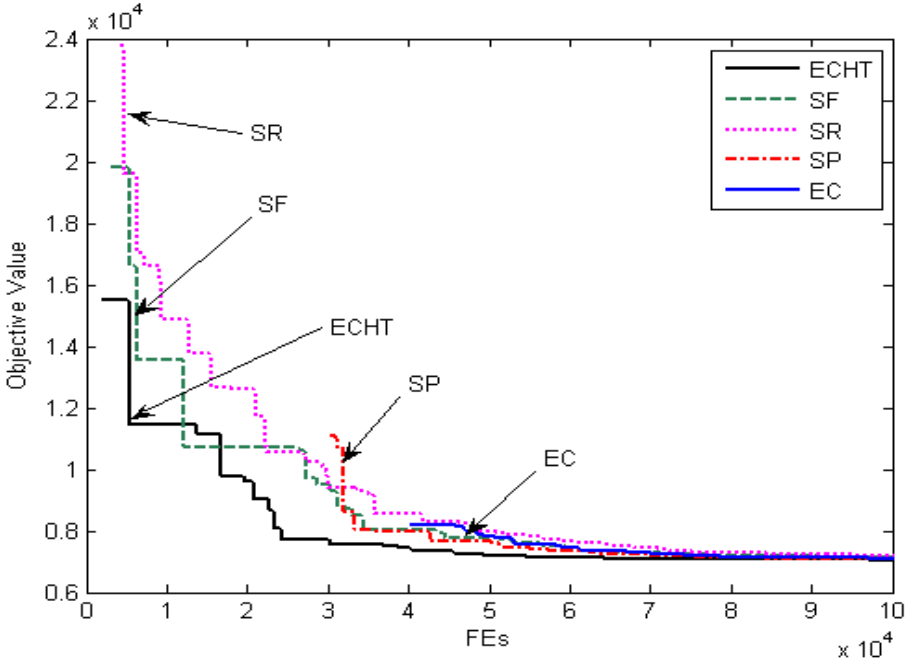
Comparison of single constraint handling methods and ECHT.

FUNCTION	SF	SP	EC	SR	ECHT1	ECHT2
G01	1	0	1	1	-	-
G02	0	0	0	0	-	-
G03	0	0	0	0	-	-
G04	0	0	0	0	-	-
G05	1	1	1	1	-	-
G06	1	1	1	1	-	-
G07	0	0	0	0	-	-
G08	0	0	0	0	-	-
G09	0	0	0	0	-	-
G10	1	1	1	1	-	-
G11	0	0	0	0	-	-
G12	0	0	0	0	-	-
G13	0	0	0	0	-	-
G14	1	0	1	1	-	-
G15	0	0	0	0	-	-
G16	0	0	0	0	-	-
G17	1	1	0	1	-	-
G18	0	0	0	0	-	-
G19	1	1	1	1	-	-
G21	0	0	0	0	-	-
G23	0	0	0	0	-	-
G24	0	0	0	0	-	-
RANK	57	60	54	68	36	22

RESULTS



Problem G01



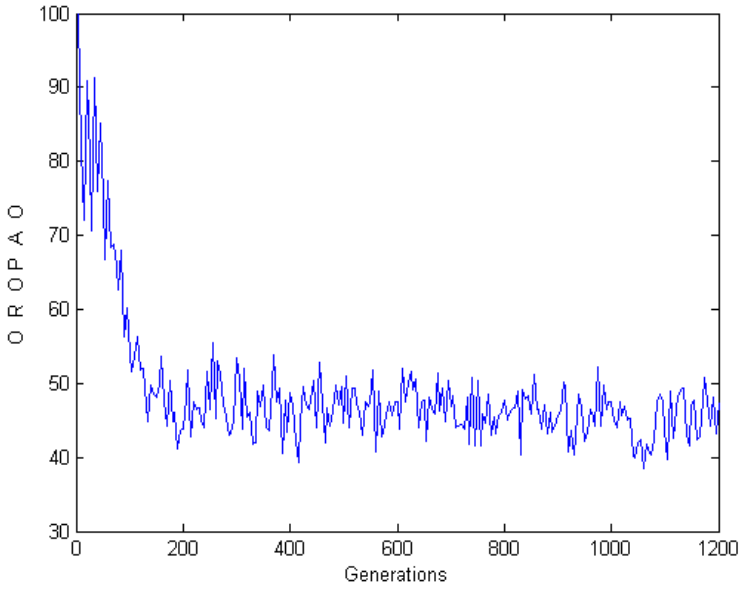
Problem G10

Convergence Plots

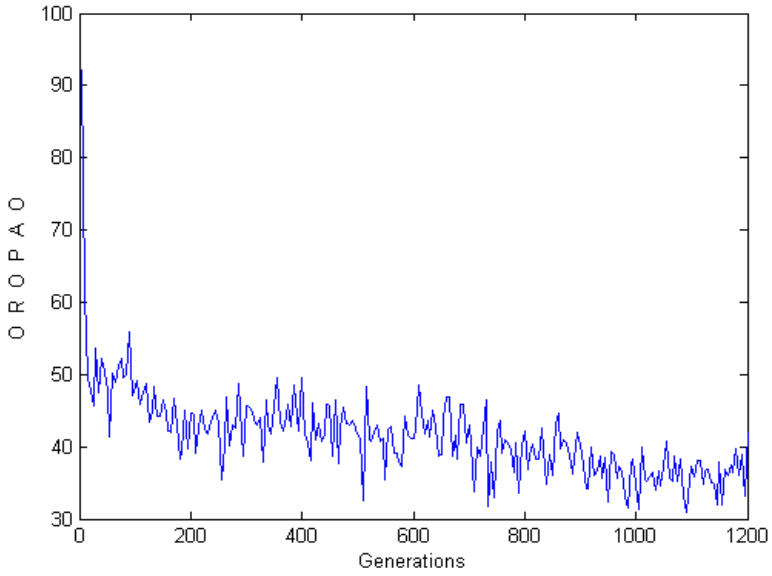
Conclusions

- The superior performance of ensemble over the single constraint handling methods within the given maximum function evaluations is due to:
  - the efficient use of every function call by all four populations and the search process in the ensemble benefiting from the better performance of different constraint handling methods during different stages of the search process.
  - The evidence for the efficient use of every function call in the ensemble is presented in the following figures, in which the horizontal axis represent the generation, while the vertical axis represents the number of offspring population members rejected by their own populations, but are accepted by other populations.
  - Due to the diverse nature of the constraint handling methods present in ECHT premature convergence can be avoided and consistency/convergence speed can be improved.

RESULTS



Problem G01



Problem G10

Plot showing no. of Offspring Rejected by Own Population and Accepted by Others

Comparison Against Other Methods

Characteristics of the state-of-the-art methods

Algorithm	Search Method	Runs	Gen	Max_FEs	Prob.	δ
(ES+SR) [RY00]	(30,200)-ES	30	1750	350 000	13	0.0001
SAFF [FW03]	GA with Gray coding	20	20000	1400 000	11	0.0001
AGFCOGA [VY05]	Real Coded GA	50	5000	50 000	11	0.001
(RCGA+SP) [TY06]	Real Coded GA	50	5000	500 000	13	0.0001
SMES [MC05]	(100,300)-ES	30	800	240 000	13	Decreases from 1.0E-03 to 4E-04
ATMES [WCZZ08]	(50,300)-ES	30	800	240 000	13	Decreases from 3.0 to 5E-06
Multi-objective [WCGZ07]	Hybrid EA, popln-250	30	-	240 000	13	1E-07
EP with ECHT, SF, SP, EC, SR	EP, population-200	30	1200	240 000	24	Decreases from 3.0 to 1.0E-04

RESULTS

Comparison with state-of the-art-methods

Fcn & Optimal value		(ES+SR) [RY00]	SMES [MC05]	ATMES [WCZZ08]	Multi-Objective [WCGZ07]	ISR [RY05]	ECHT2
G02 -0.803619	b	-0.803515(5)	-0.803601(3)	-0.803339(6)	-0.803550(4)	-0.803619(1)	-0.8036191(1)
	Md	-0.785800(6)	-0.792549(4)	-0.792420(5)	-0.794591(2)	-0.793082(3)	-0.8033239(1)
	Mn	-0.781975(6)	-0.785238(4)	-0.790148(3)	-0.792610(2)	-0.782715(5)	-0.7998220(1)
	w	-0.726288(5)	-0.751322(4)	-0.756986(3)	-0.756938(2)	-0.723591(6)	-0.7851820(1)
	sd	2.0E-02	1.67E-02	1.3E-02	1.0E-02	2.20E-02	6.29E-03
G05 5126.4967	b	5126.497(1)	5126.599(6)	5126.4989(4)	5126.4981(4)	5126.497(1)	5126.4967(1)
	Md	5127.372(5)	5160.198(6)	5126.776(4)	5126.4981(3)	5126.497(1)	5126.4967(1)
	Mn	5128.881(5)	5174.492(6)	5127.648(4)	5126.4981(3)	5126.497(1)	5126.4967(1)
	w	5142.472(5)	5160.198(6)	5135.256(4)	5126.4984(3)	5126.497(1)	5126.4967(1)
	sd	3.5E+00	5.006E+01	1.8E+00	1.727E-07	7.20E-13	0.0E+00
G10 7049.248	b	7054.316(6)	7051.903(4)	7052.253(5)	7049.286598(3)	7049.248(1)	7049.2483(1)
	Md	7372.613(6)	7253.603(5)	7215.357(4)	7049.486145(3)	7049.248(1)	7049.2488(2)
	Mn	7559.192(6)	7253.047(5)	7250.437(4)	7049.525438(3)	7049.250(2)	7049.2490(1)
	w	8835.655(6)	7638.366(5)	7560.224(4)	7049.984208(3)	7049.270(2)	7049.2501(1)
	sd	5.3E+02 {6}	1.36E+02	1.2E+02	1.502E-01	3.20E-03	6.60E-04
RANK		133	152	94	66	48	29

Conclusions on Comparison with state-of the-art-methods

- The algorithms are ranked based on the best (b), median (m), mean (M) and worst (w) values
- The minimum rank that any algorithm could achieve is 28 (7 problems and 4 rankings each). The overall rank for each algorithm is presented in the last row of Table.
- Thus the average rank for (ES+SR), SMES, ATMES, multi-objective, ISR and ECHT2 are 4.75, 5.43, 3.36, 2.36, 1.71 and 1.06 respectively.
- From the ranking, it is obvious that the performance of ECHT2 is superior to the five state-of-the-art algorithms.

NANYANG TECHNOLOGICAL UNIVERSITY

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

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

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

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)* ≠ *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*≤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*≥*Max_FEs*

Benchmarking Evolutionary Algorithms

- CEC05 comparison results (Single obj. + bound const.)
- CEC06 comparison results (Single obj + general const.)

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)

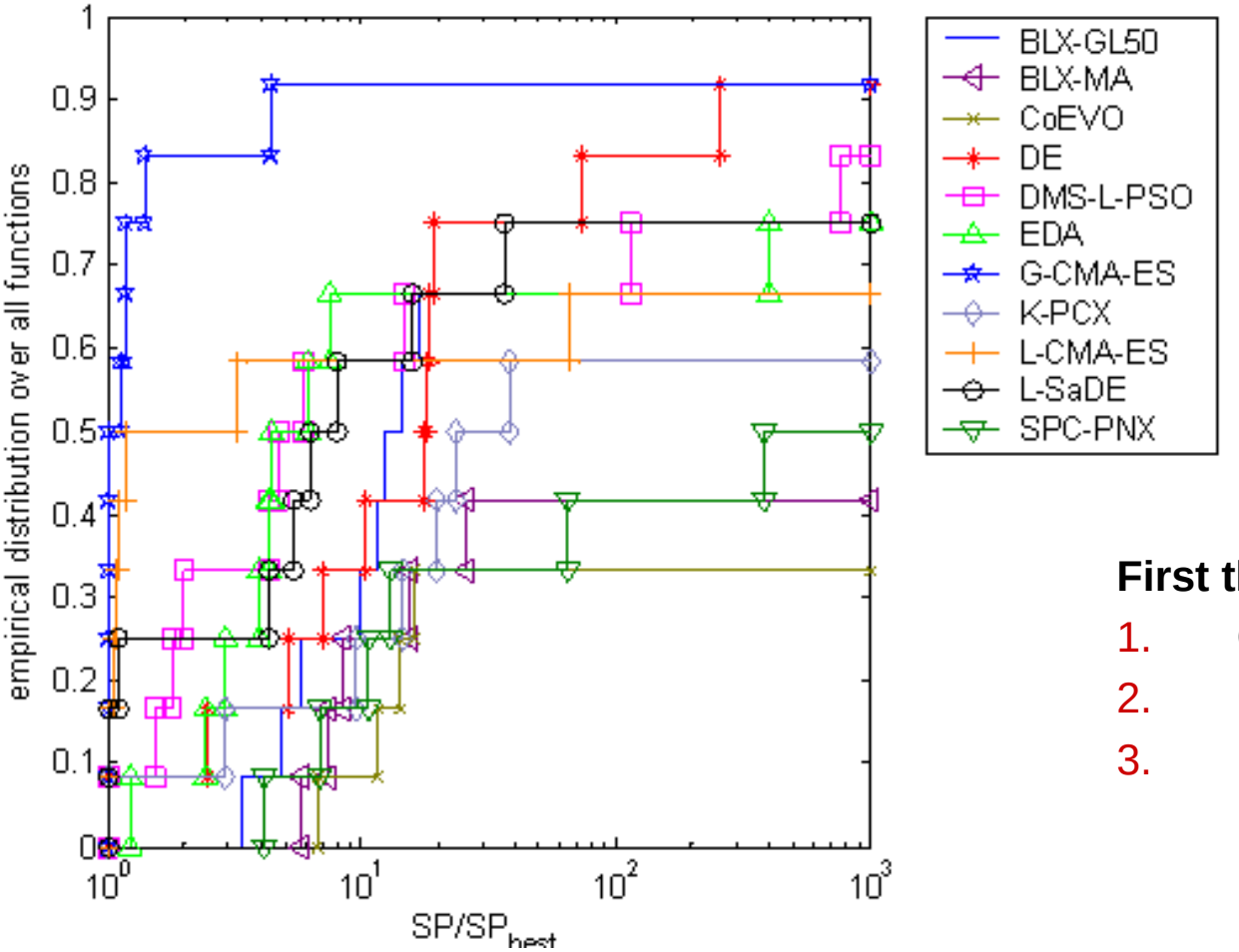
CEC'05 Comparison Results

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

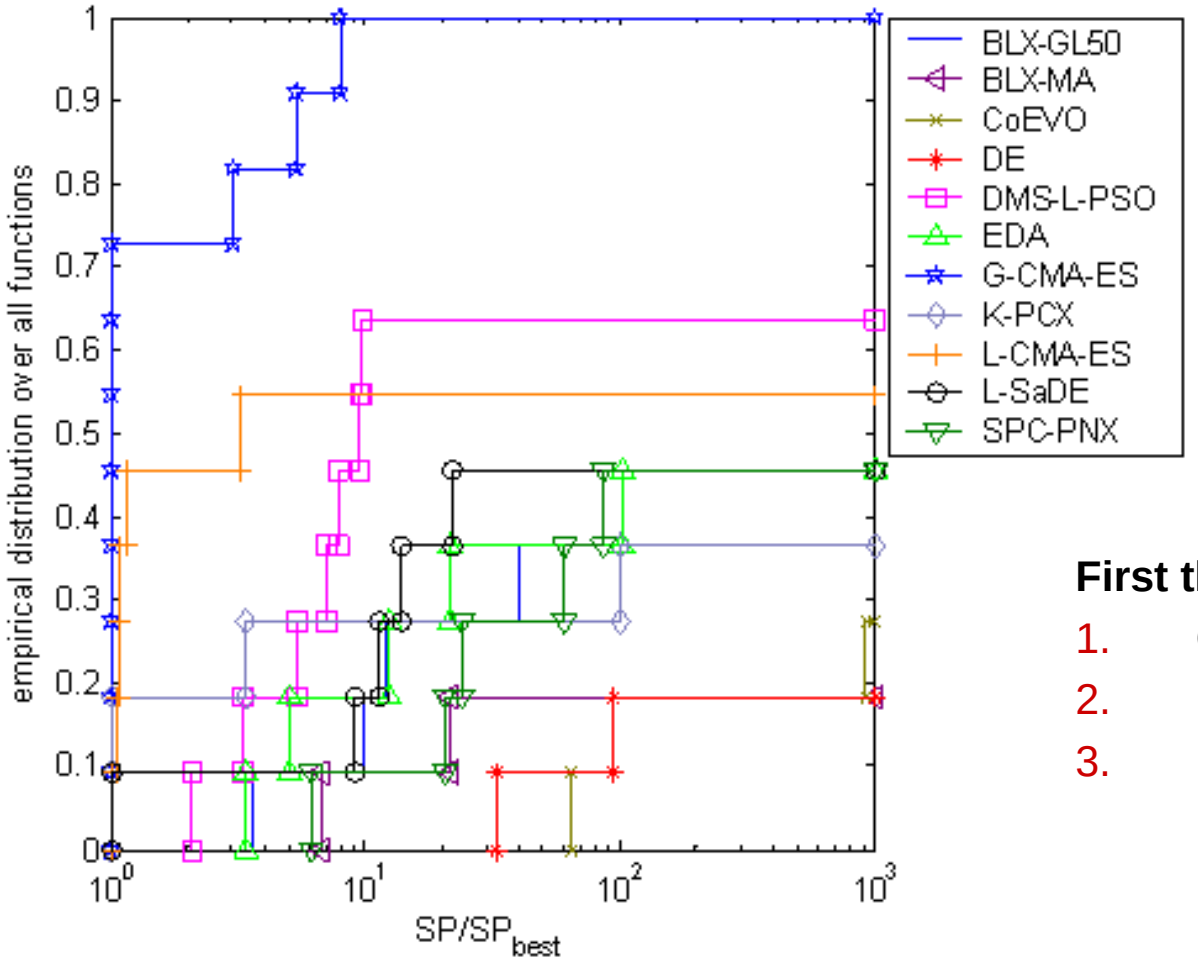
CEC'05 Comparison Results

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

Involved 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 **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

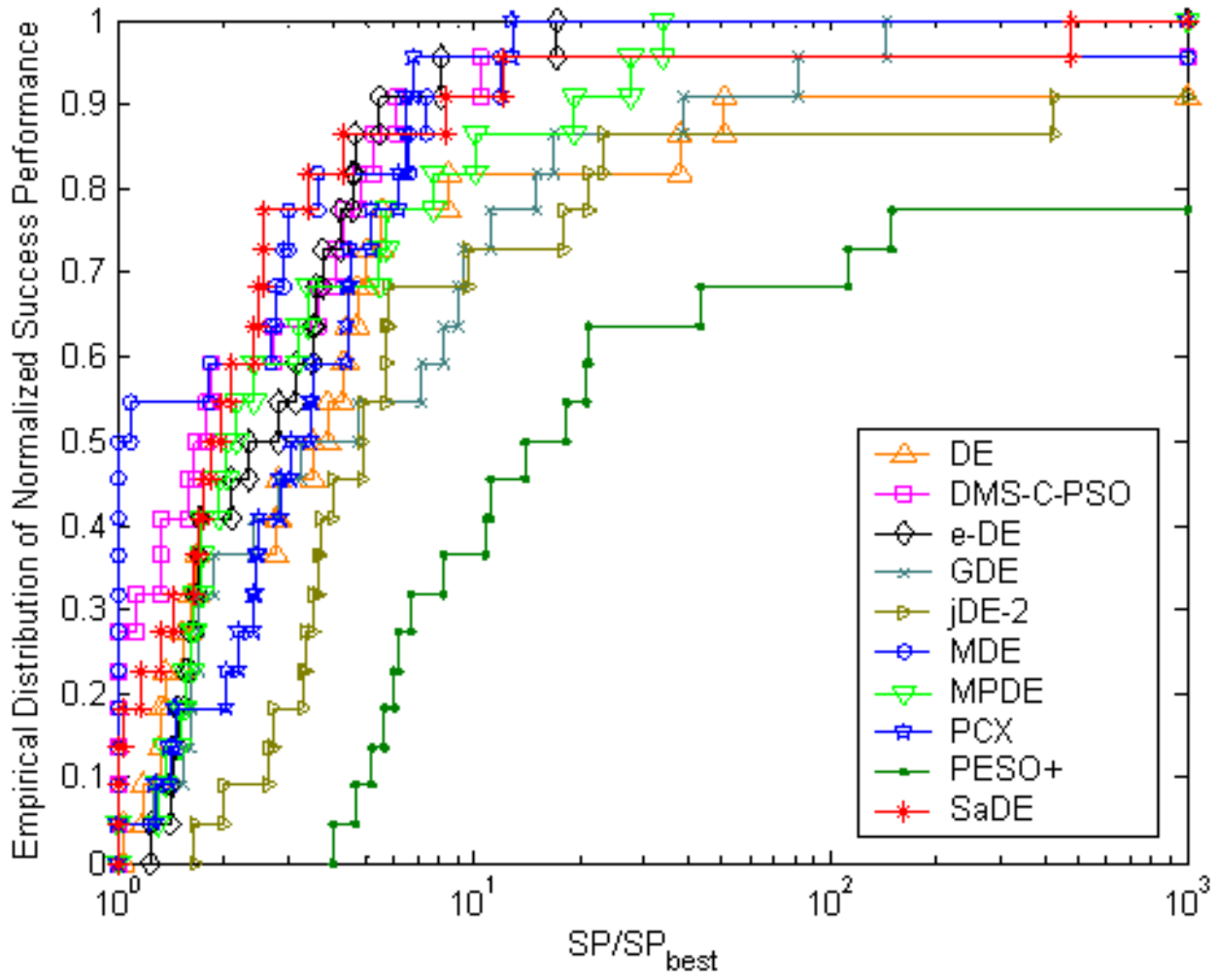
Func / Algorithms		1	2	3	4	5	6	7	8	9	10	11
DE	81.64%	100%	84%	0%	100%	100%	100%	100%	100%	100%	100%	100%
DMS-C-PSO	95.27%	100%	96%	100%	100%	100%	100%	100%	100%	100%	100%	100%
e-DE	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
GDE	80.91%	100%	72%	4%	100%	92%	100%	100%	100%	100%	100%	100%
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

Algorithms \ Func	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

Acknowledgement

- Thanks to the CEC 2013 organizers for giving us this opportunity

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

- Thanks to my former and current research students and staff for contributing to this presentation: Ms Nandar Lynn, Mr. Sheldon Hui, Dr. Qu Boyang, Dr. Qin Kai, Dr Jane Jing Liang, Dr Vicky Ling Huang, Ms Ling Yu, Dr Mallipeddi Rammohan, Dr. Zhao Shizheng,

NANYANG TECHNOLOGICAL UNIVERSITY