Differential Evolution: Population Topologies, Ensemble Strategies and Adaptation

Dr. P. N. Suganthan, EEE, NTU, Singapore

Some Software Resources Available from:

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

IEEE CEC 2017, Donostia - San Sebastián, Spain, 5th June 2017

Overview

I. Introduction to Real Variable Optimization & DE

II. Single Objective Optimization

III. Constrained Optimization

IV. Dynamic Optimization

V. Multi-objective Optimization

VI. Large Scale Optimization

VII. Multimodal Optimization

But, first a little publicity .

S. Das, S. S. Mullick, P. N. Suganthan, "Recent Advances in Differential Evolution - An Updated Survey," Swarm and Evolutionary Computation, Vol. 27, pp. 1-30, 2016.

S. Das and P. N. Suganthan, "Differential Evolution: A Survey of the State-of-the-Art", *IEEE Trans. on Evolutionary Computation*, 15(1):4 – 31, Feb. 2011.

Optimization Benchmark Test Problems, Surveys,

Codes of several of our research publications

Are available from

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

(limited to our own publications & CEC Competitions)

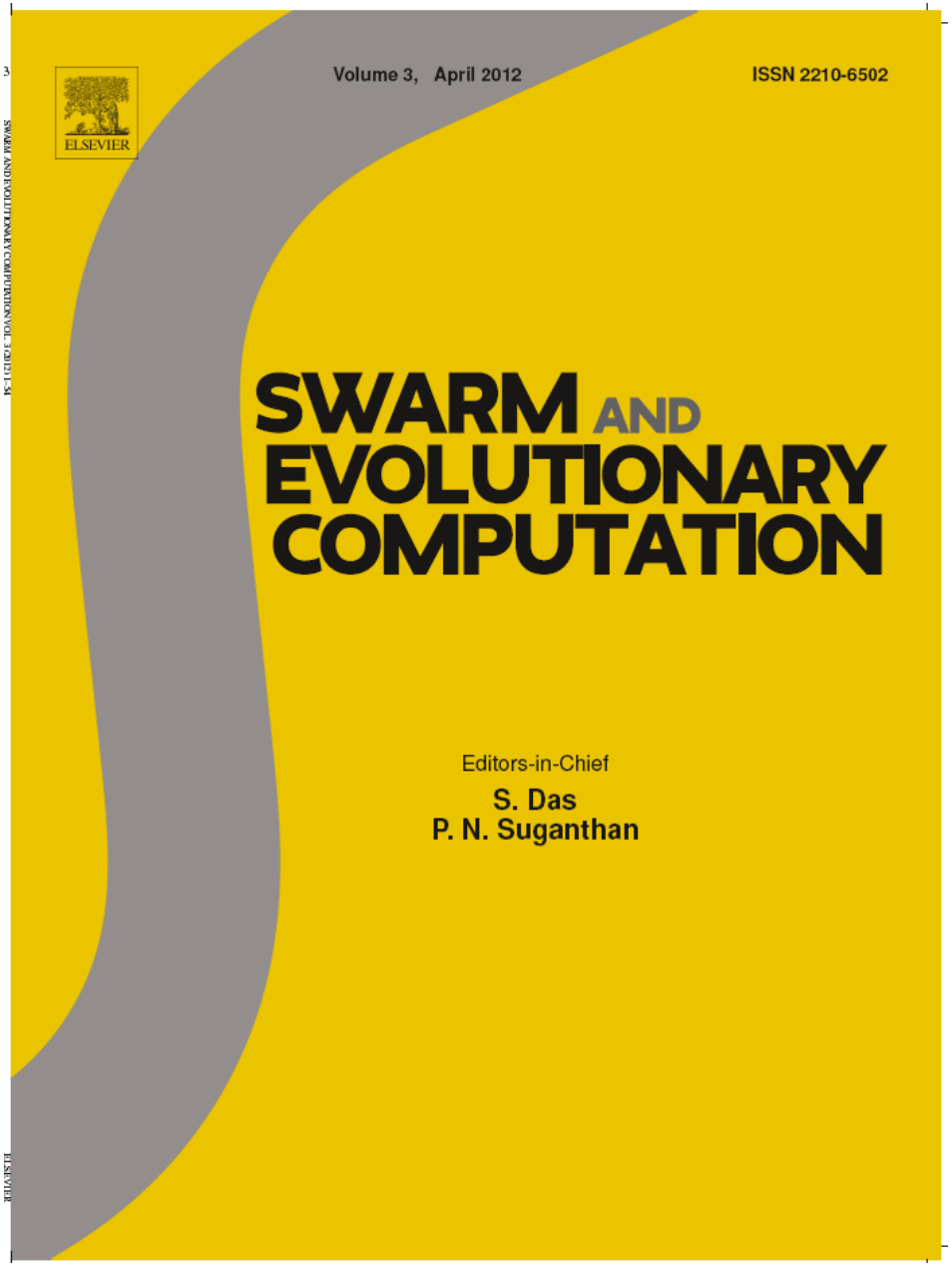
Ensemble / Adaptive Methods for Evolutionary Algorithms:

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

Consider submitting to SWEVO journal dedicated to the EC-SI fields

SCI Indexed from Vol. 1, Issue 1.

2016 IFs:
2 years = 2.9
5 years= 5.7



Overview

I. Introduction to Real Variable Optimization & DE

II. Single Objective Optimization

III. Multimodal Optimization

IV. Dynamic Optimization

V. Multi-objective Optimization

VI. Large Scale Optimization

VII. Constrained Optimization

General Thoughts: NFL (No Free Lunch Theorem)

- Glamorous Name for Commonsense?
  - Over a large set of problems, it is impossible to find a single best algorithm
  - DE with Cr=0.90 & Cr=0.91 are two different algorithms → Infinite algos.
  - **Practical Relevance:** Is it common for a practicing engineer to solve several practical problems at the same time? (NO)
  - **Academic Relevance:** Very High

Other NFL Like Commonsense Scenarios

Panacea: A medicine to cure all diseases (No need for doctors), *Amrita* the nectar of immortal perfect life …

Silver bullet: in politics … (**you can search these on internet**)

Jack of all trades, but master of none

If you have a hammer all problems look like nails

General Thoughts: Convergence

- What is exactly convergence in the context of EAs & SAs ?
  - The whole population reaching a single point (within a tolerance)
  - Single point based search methods & convergence …
- In the context of real world problem solving, are we going to reject a good solution because the population hasn't converged ?

- Good to have all population members converging to the global solution **OR** good to have high diversity even after finding the global optimum ? **(Fixed Computational budget Scenario)**

- **What we do not want to have:**

For example, in the context of PSO, we do not want to have chaotic oscillations

$$c_1 + c_2 > 4.1+$$

General Thoughts: Algorithmic Parameters

- Good to have many algorithmic parameters / operators ?
- Possible to be robust against parameter / operator variations ?
- What are Reviewers' preferences ?
- **Or good to have several parameters that can be adaptively tuned on the fly to achieve top performance on diverse problems?**
- **If NFL says that a single algorithm is not the best for each problem in a large set of problems, then good to have many algorithmic parameters & operators to be adapted for different problems !!**

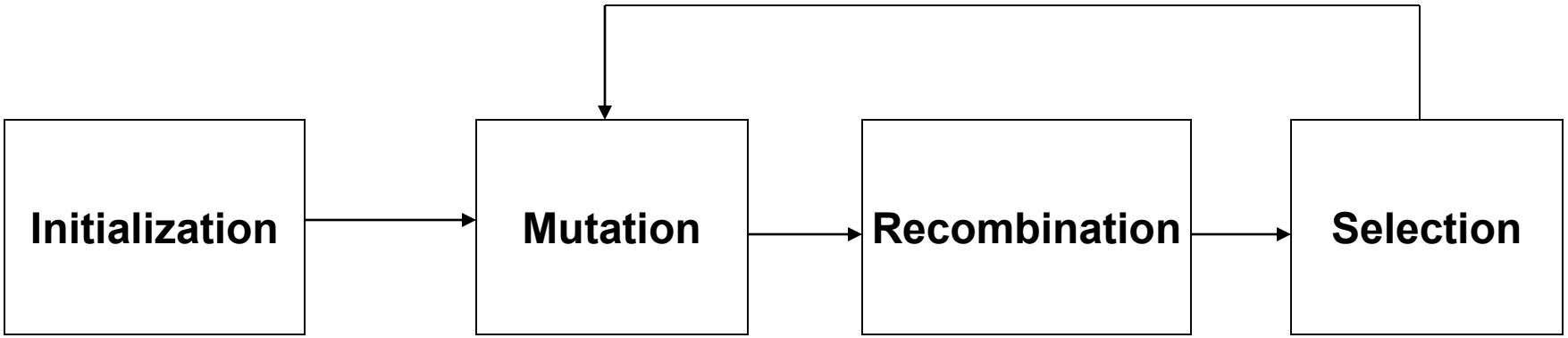
CEC 2015 Competitions: "Learning-Based Optimization"

Similar Literature: Thomas Stützle, Holger Hoos, …

Differential Evolution

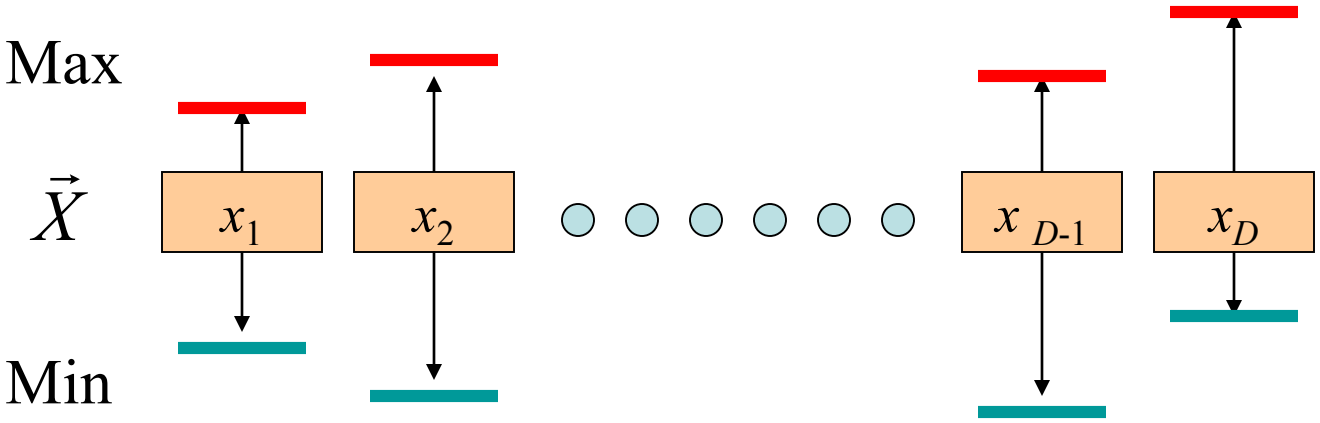
- **A stochastic population-based algorithm for continuous function optimization (Storn and Price, 1995)**
- **Finished 3rd at the First International Contest on Evolutionary Computation, Nagoya, 1996 (*icsi.berkley.edu/~storn*)**
- **Outperformed several variants of GA and PSO over a wide variety of numerical benchmarks over past several years.**
- **Continually exhibited remarkable performance in competitions on different kinds of optimization problems like dynamic, multi-objective, constrained, and multi-modal problems held under IEEE congress on Evolutionary Computation (CEC) conference series.**
- **Very easy to implement in any standard programming language.**
- **Very few control parameters (typically three for a standard DE) and their effects on the performance have been well studied.**
- **Spatial complexity is very low as compared to some of the most competitive continuous optimizers like CMA-ES.**

- DE is an Evolutionary Algorithm
- This Class also includes GA, Evolutionary Programming and Evolutionary Strategies



Basic steps of an Evolutionary Algorithm

Representation

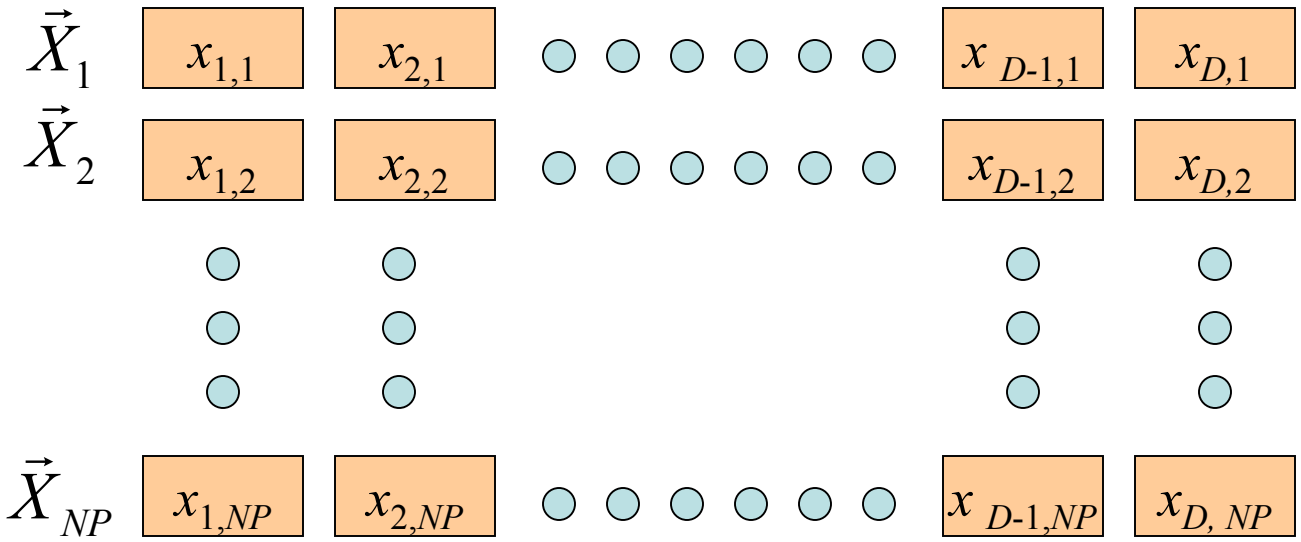


Solutions are represented as vectors of size *D* with each value taken from some domain.

May wish to constrain the values taken in each domain above and below.

Maintain Population - *NP*

We will maintain a population of size *NP*



The population size *NP*

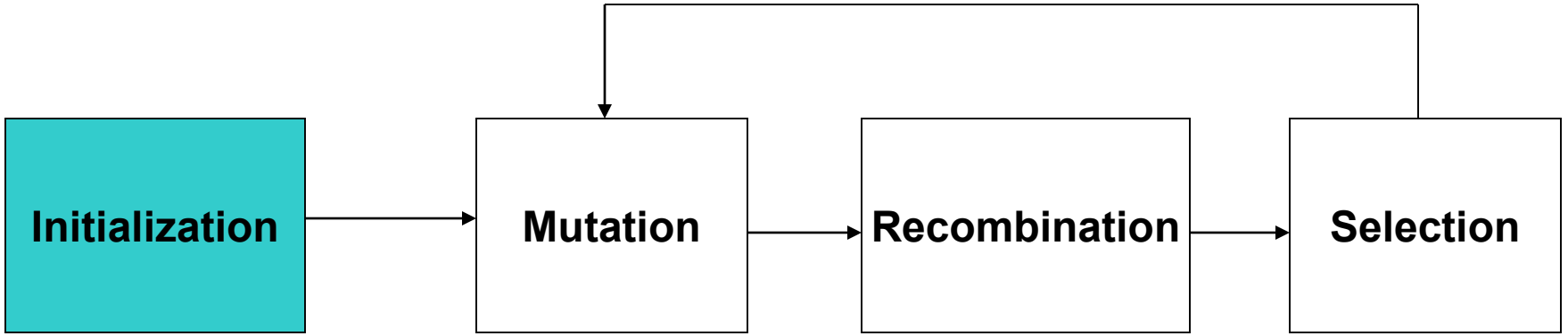
1) The influence of *NP* on the performance of DE is yet to be extensively studied and fully understood.

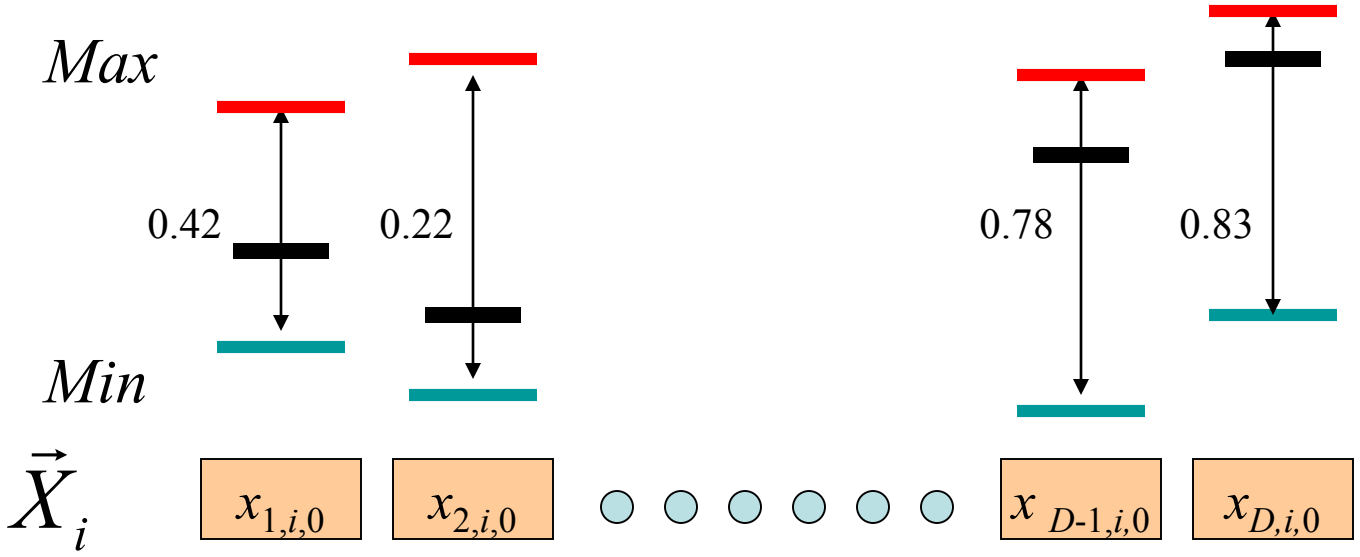
2) Storn and Price have indicated that a reasonable value for *NP* could be chosen between 5*D* and 10*D* (*D* being the dimensionality of the problem).

3) Brest and Maučec presented a method for gradually reducing population size of DE. The method improves the efficiency and robustness of the algorithm and can be applied to any variant of DE.

4) But, recently, all best performing DE variants used populations ~50-100 for dimensions from 50D to 1000D for the following scalability Special Issue:

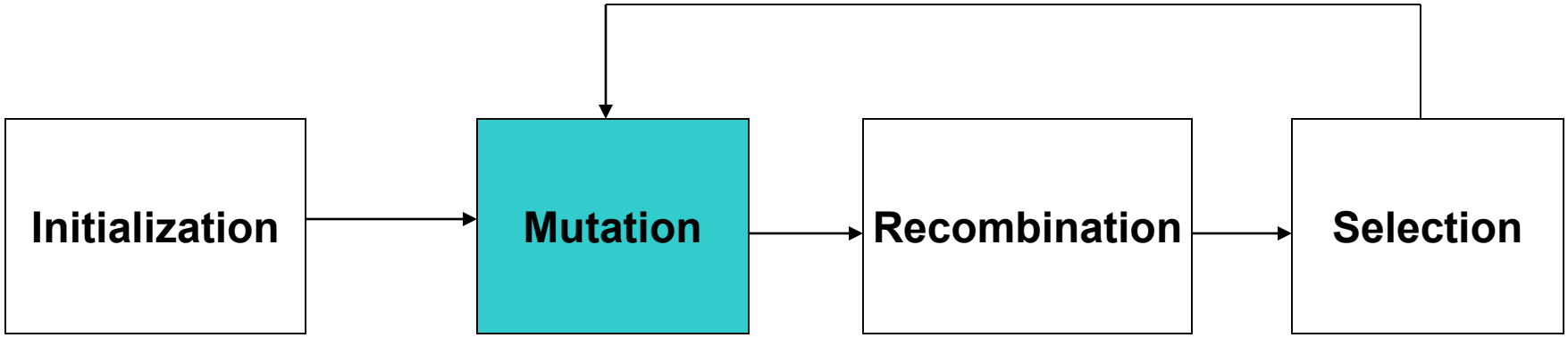
F. Herrera M. Lozano D. Molina, "Test Suite for the Special Issue of Soft Computing on Scalability of Evolutionary Algorithms and other Metaheuristics for Large Scale Continuous Optimization Problems". Available: http://sci2s.ugr.es/eamhco/CFP.php.





$$x_{j,i,0} = x_{j,\min} + rand_{i,j}[0,1] \cdot (x_{j,\max} - x_{j,\min})$$

Different $rand_{i,j}[0,1]$ values are instantiated for each i and j.

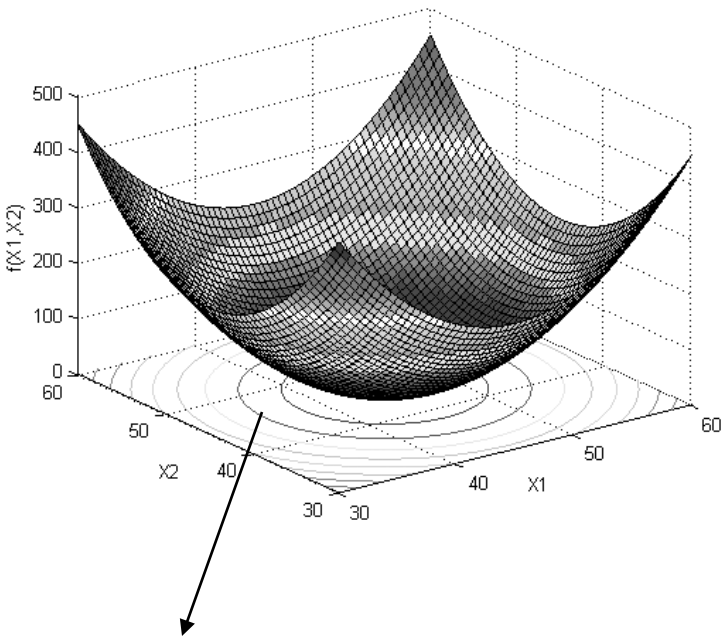


- For each vector select three other parameter vectors randomly.
- Add the weighted difference of two of the parameter vectors to the third to form a donor vector (most commonly seen form of DE-mutation):

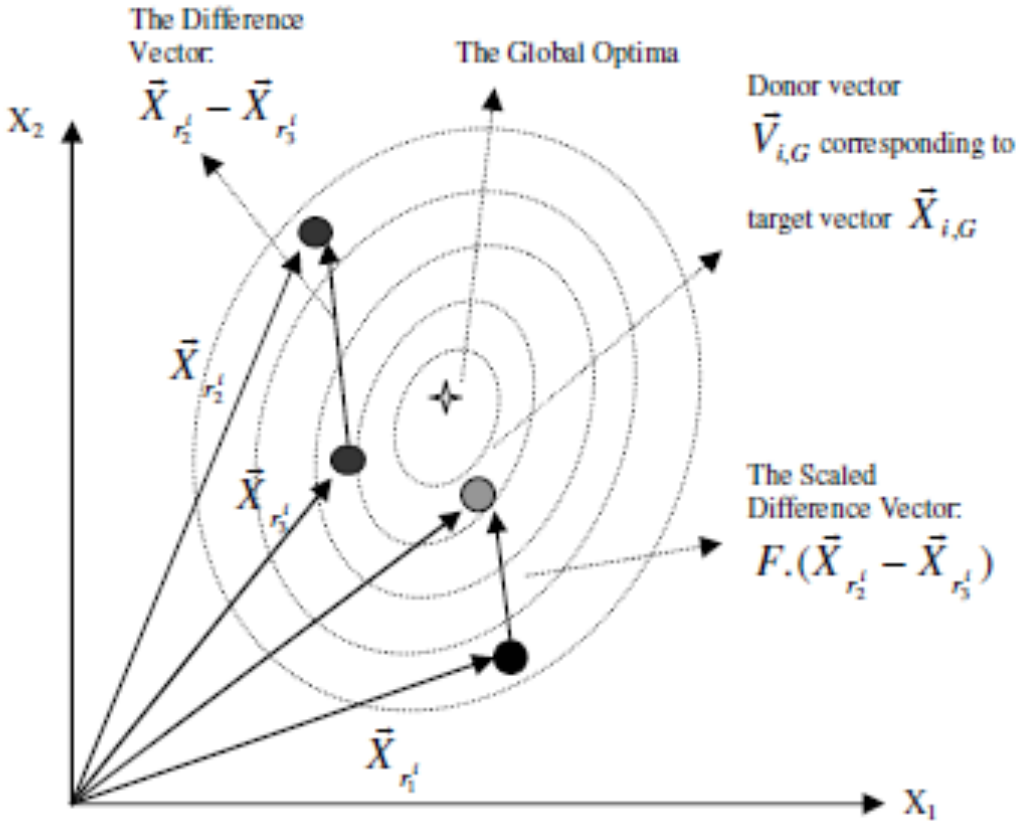
$$\vec{V}_{i,G} = \vec{X}_{r_1^i,G} + F \cdot (\vec{X}_{r_2^i,G} - \vec{X}_{r_3^i,G}).$$

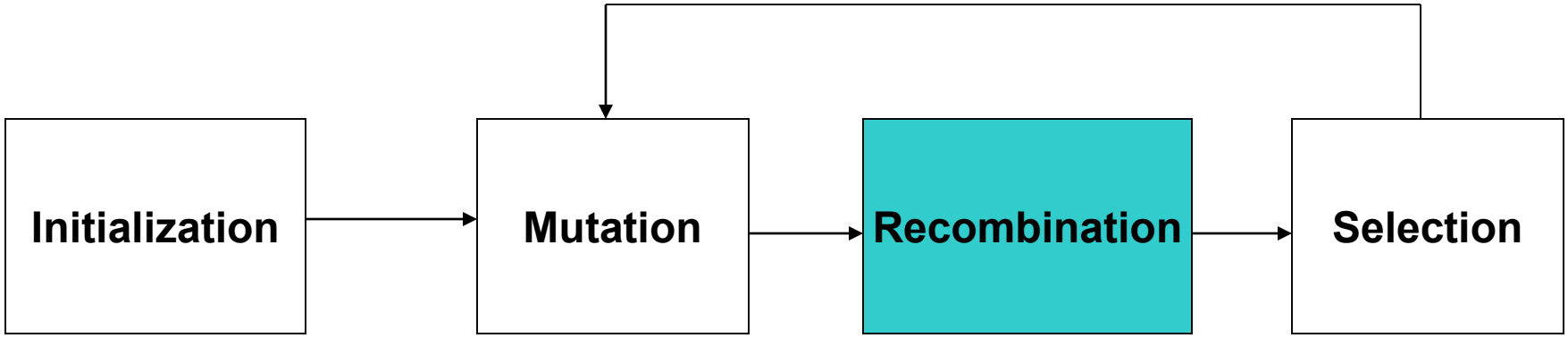
- The scaling factor *F* is a constant from (0, 2)
- Self-referential Mutation

Example of formation of donor vector over two-dimensional constant cost contours



Constant cost contours of Sphere function





Binomial (Uniform) Crossover:

Components of the donor vector enter into the trial offspring vector in the following way:

Let j_{rand} be a randomly chosen integer between 1,...,D.

$$u_{j,i,G} = \begin{cases} v_{j,i,G}, & \text{if } (rand_{i,j}[0,1) \le Cr \text{ or } j = j_{rand}) \\ x_{j,i,G}, & \text{otherwise,} \end{cases}$$

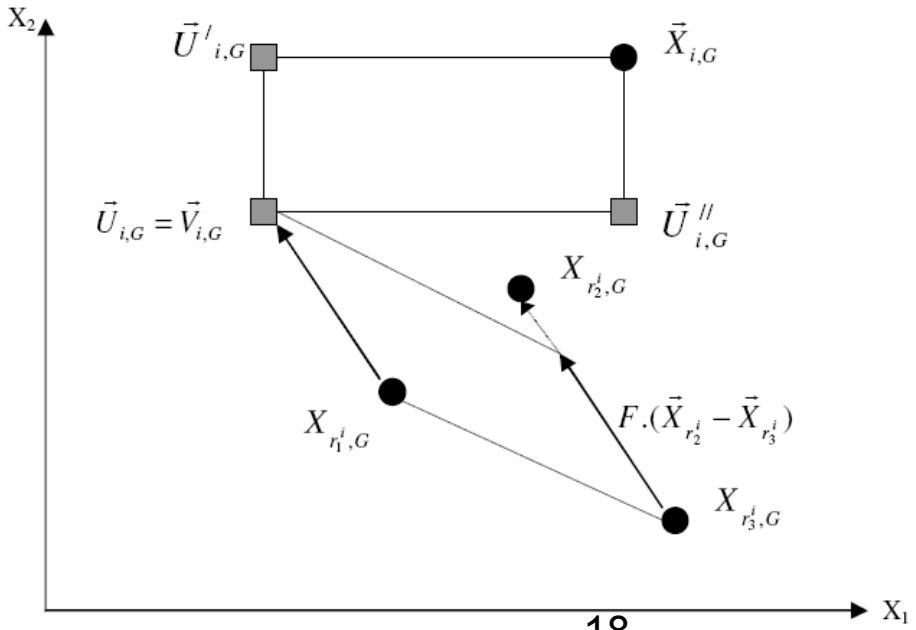
An Illustration of Binomial Crossover in 2-D Parametric Space:

Three possible trial vectors:

i) $\vec{U}_{i,G} = \vec{V}_{i,G}$ such that both the components of $\vec{U}_{i,G}$ are inherited from $\vec{V}_{i,G}$.

ii) $\vec{U}'_{i,G}$, in which the first component ($j = 1$) comes from $\vec{V}_{i,G}$ and the second one ($j = 2$) from $\vec{X}_{i,G}$.

iii) $\vec{U}''_{i,G}$, in which the first component ($j = 1$) comes from $\vec{X}_{i,G}$ and the second one ($j = 2$) from $\vec{V}_{i,G}$.



Exponential (two-point modulo) Crossover:

First choose integers *n* (as starting point) and *L* (number of components the donor actually contributes to the offspring) from the interval [1,*D*]

$$u_{j,i,G} = \begin{cases} v_{j,i,G}, & \text{for } j = \langle n \rangle_D, \langle n+1 \rangle_D, \ldots, \langle n+L-1 \rangle_D \\ x_{j,i,G}, & \text{for all other } j \in [1, D], \end{cases}$$

where the angular brackets $\langle \ \rangle_D$ denote a modulo function with modulus *D*.

Pseudo-code for choosing *L:*

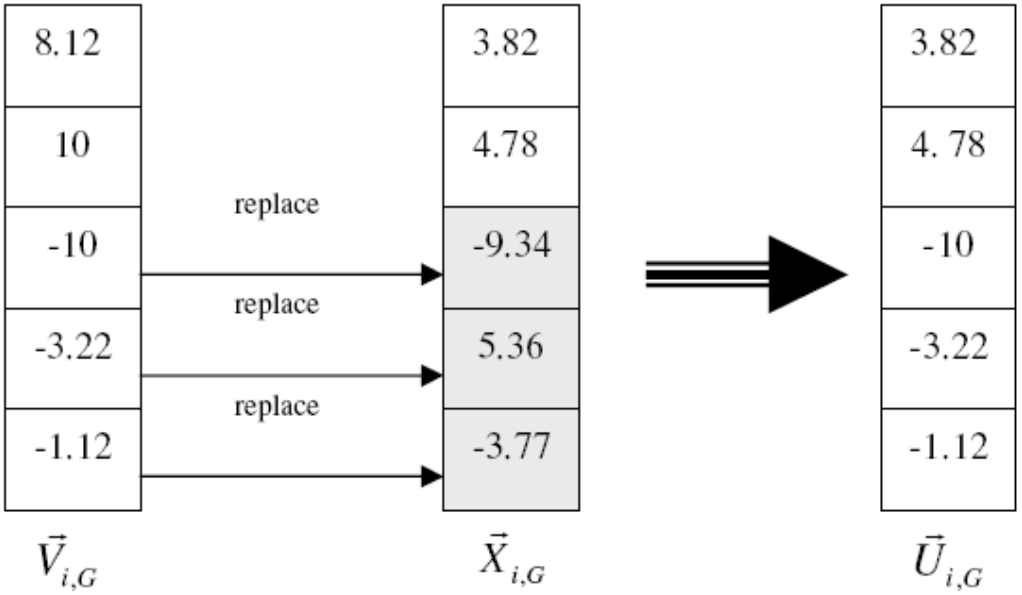
```
L = 0;
DO
{
    L = L+1;
} WHILE (((rand[0,1) ≤ Cr) AND (L < D)) ;
```

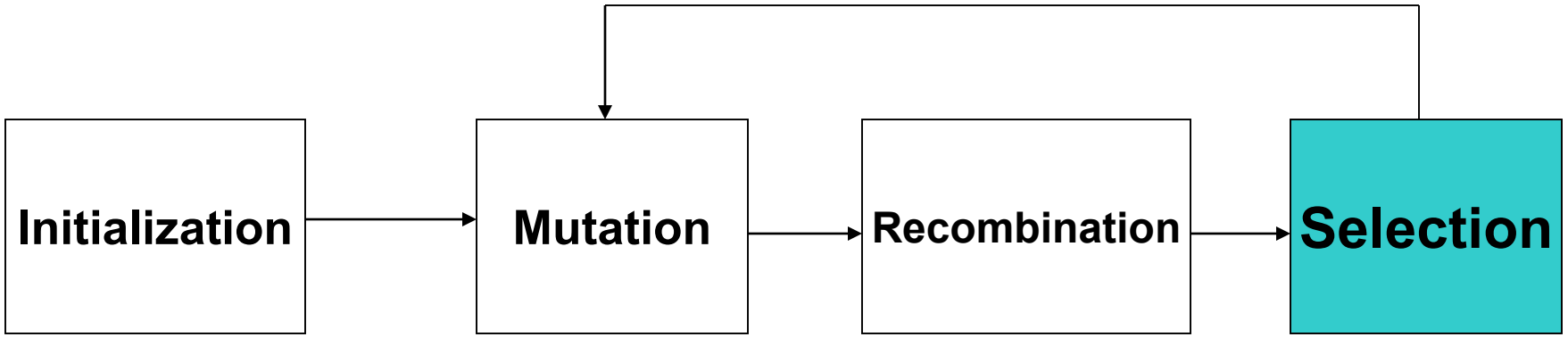
Exploits linkages among neighboring decision variables. If benchmarks have this feature, it performs well. Similarly, for real-world problems with neighboring linkages.

Example: Let us consider the following pair of donor and target vectors

$$\vec{X}_{i,G} = \begin{bmatrix} 3.82 \\ 4.78 \\ -9.34 \\ 5.36 \\ -3.77 \end{bmatrix} \qquad \vec{V}_{i,G} = \begin{bmatrix} 8.12 \\ 10 \\ -10 \\ -3.22 \\ -1.12 \end{bmatrix}$$

Suppose *n* = 3 and *L*= 3 for this specific example. Then the exponential crossover process can be shown as:





- **"Survival of the fitter" principle in selection: The trial offspring vector is compared with the target (parent) vector and the one with a better fitness is admitted to the next generation population.**

$$\vec{X}_{i,G+1} = \vec{U}_{i,G}, \text{ if } f(\vec{U}_{i,G}) \le f(\vec{X}_{i,G})$$
$$= \vec{X}_{i,G}, \text{ if } f(\vec{U}_{i,G}) > f(\vec{X}_{i,G})$$

- **Importance of parent-mutant crossover & parent-offspring competition-based selection**

An Example of Optimization by DE

Consider the following two-dimensional function

$$f(x, y) = x^2 + y^2$$ ***The minima is at (0, 0)***

Let's start with a population of 5 candidate solutions randomly initiated in the range (-10, 10)

$X_{1,0}$ = [2, -1] $X_{2,0}$ = [6, 1] $X_{3,0}$ = [-3, 5] $X_{4,0}$ = [-2, 6]
$X_{5,0}$= [6,-7]

For the first vector X_1, randomly select three other vectors say X_2, X_4 and X_5

Now form the donor vector as, $V_{1,0} = X_{2,0} + F.\ (X_{4,0} - X_{5,0})$

$$V_{1,0} = \begin{bmatrix} 6 \\ 1 \end{bmatrix} + 0.8 \times \left\{ \begin{bmatrix} -2 \\ 6 \end{bmatrix} - \begin{bmatrix} 6 \\ -7 \end{bmatrix} \right\} = \begin{bmatrix} -0.4 \\ 10.4 \end{bmatrix}$$

Now we form the trial offspring vector by exchanging components of $V_{1,0}$ with the target vector $X_{1,0}$

Let *rand*[0, 1) = 0.6. If we set *Cr* = 0.9, since 0.6 < 0.9, $u_{1,1,0} = V_{1,1,0} = -0.4$

Again next time let *rand*[0, 1) = 0.95 > *Cr*
Hence $u_{1,2,0} = x_{1,2,0} = -1$

So, finally the offspring is $U_{1,0} = \begin{bmatrix} -0.4 \\ -1 \end{bmatrix}$

Fitness of parent:

$f(2, -1) = 2^2 + (-1)^2 = 5$

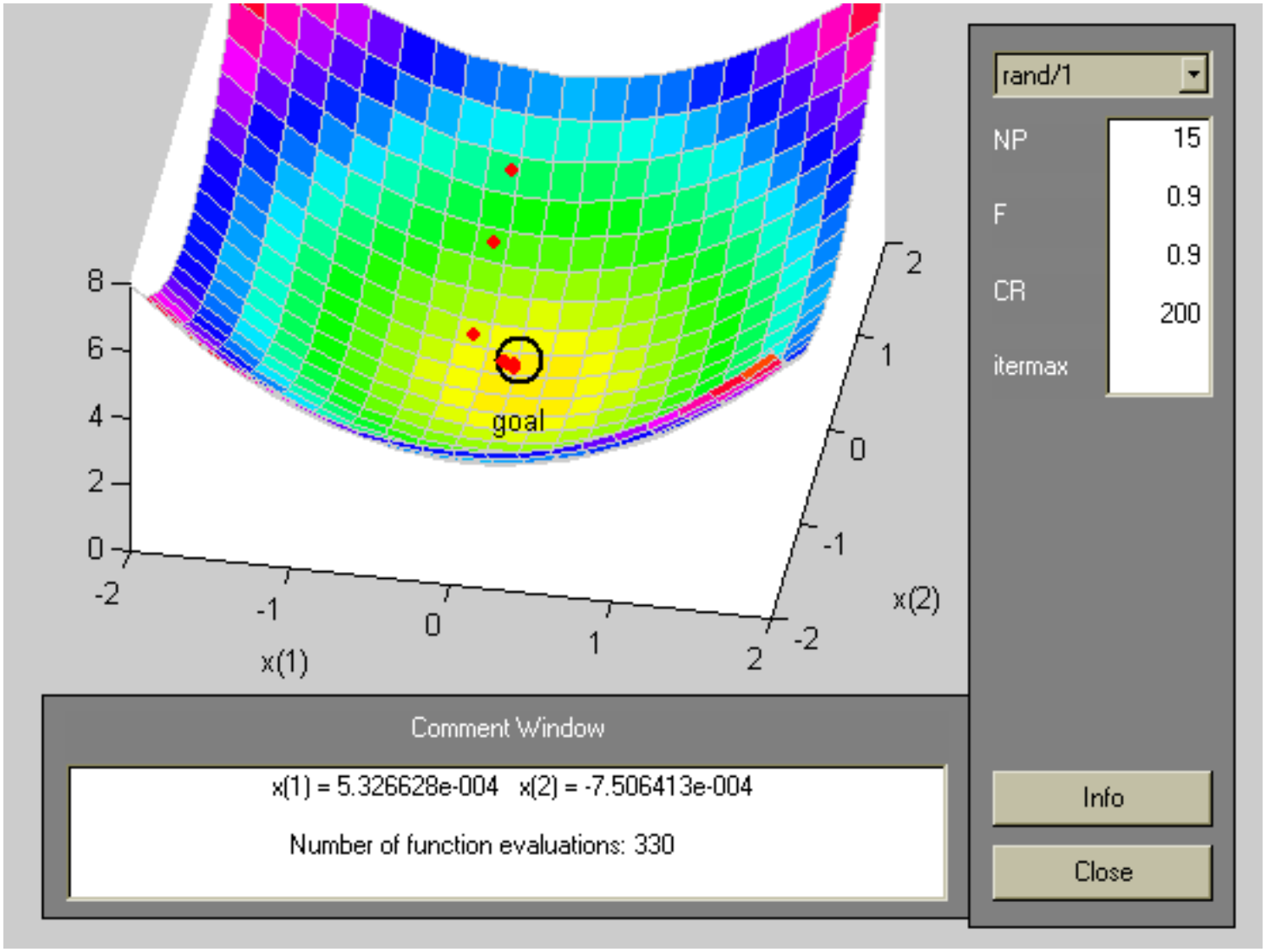
Fitness of offspring

$f(-0.4, -1) = (-0.4)^2 + (-1)^2 = 1.16$

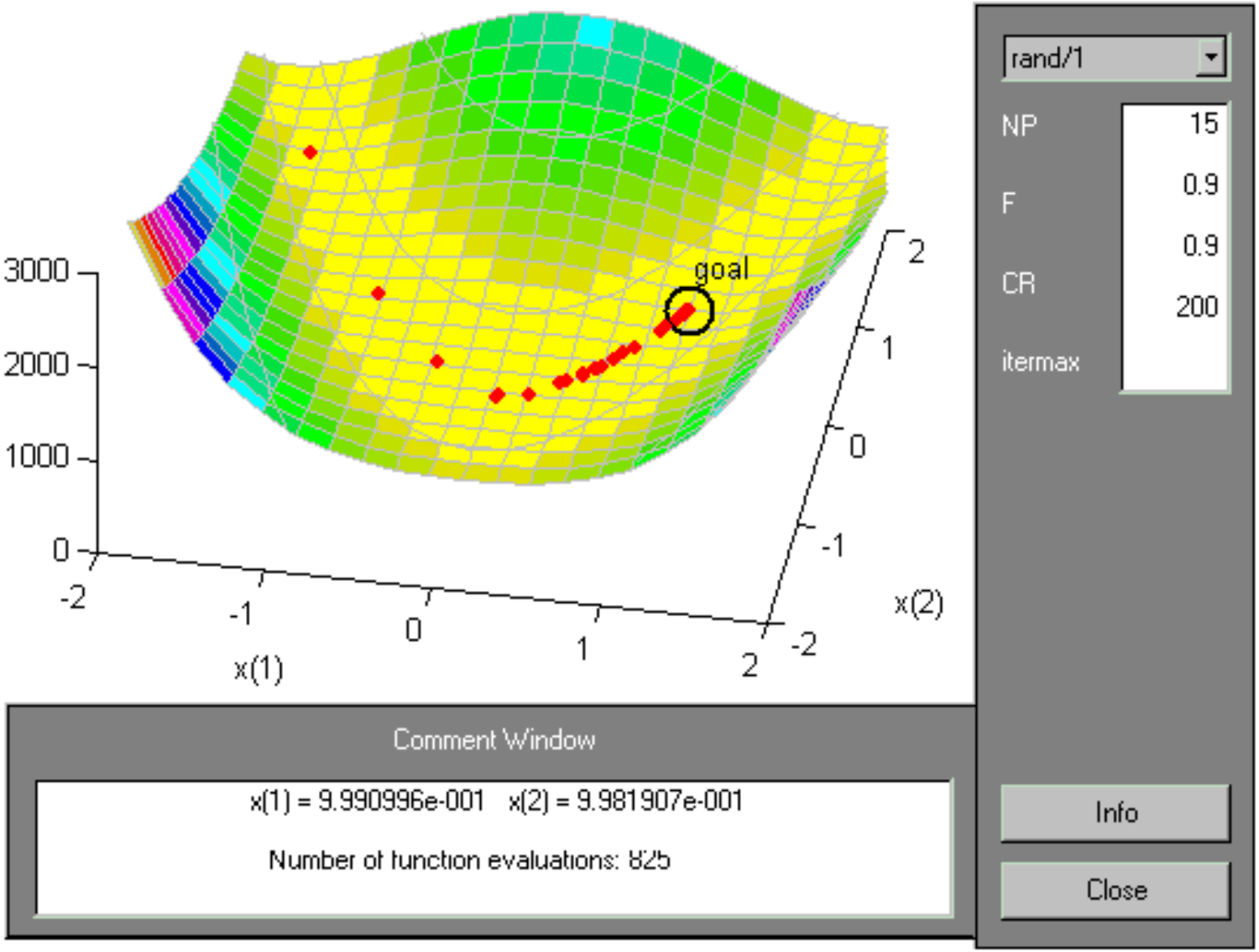
Hence the parent is replaced by offspring at *G* = 1

Population at $G = 0$	Fitness at $G = 0$	Donor vector at $G = 0$	Offspring Vector at $G = 0$	Fitness of offspring at $G = 1$	Evolved population at $G = 1$
$X_{1,0}$ = [2,-1]	5	$V_{1,0}$ =[-0.4,10.4]	$U_{1,0}$ =[-0.4,-1]	1.16	$X_{1,1}$ =[-0.4,-1]
$X_{2,0}$= [6, 1]	37	$V_{2,0}$ =[1.2, -0.2]	$U_{2,0}$ =[1.2, 1]	2.44	$X_{2,1}$ =[1.2, 1]
$X_{3,0}$= [-3, 5]	34	$V_{3,0}$ =[-4.4, -0.2]	$U_{3,0}$ =[-4.4, -0.2]	19.4	$X_{3,1}$ =[-4.4, -0.2]
$X_{4,0}$= [-2, 6]	40	$V_{4,0}$ =[9.2, -4.2]	$U_{4,0}$ =[9.2, 6]	120.64	$X_{4,1}$ =[-2, 6]
$X_{5,0}$= [6, 7]	85	$V_{5,0}$ =[5.2, 0.2]	$U_{5,0}$ =[6, 0.2]	36.04	$X_{5,1}$ =[6, 0.2]

Locus of the fittest solution: DE working on 2D Sphere Function



Locus of the fittest solution: DE working on 2D Rosenbrock Function



Five most frequently used DE mutation schemes

"DE/rand/1": $\vec{V}_i(t) = \vec{X}_{r_1^i}(t) + F \cdot (\vec{X}_{r_2^i}(t) - \vec{X}_{r_3^i}(t)).$

"DE/best/1": $\vec{V}_i(t) = \vec{X}_{best}(t) + F.(\vec{X}_{r_1^i}(t) - \vec{X}_{r_2^i}(t)).$

"DE/target-to-best/1": $\vec{V}_i(t) = \vec{X}_i(t) + F.(\vec{X}_{best}(t) - \vec{X}_i(t)) + F.(\vec{X}_{r_1^i}(t) - \vec{X}_{r_2^i}(t)),$

"DE/best/2": $\vec{V}_i(t) = \vec{X}_{best}(t) + F.(\vec{X}_{r_1^i}(t) - \vec{X}_{r_2^i}(t)) + F.(\vec{X}_{r_3^i}(t) - \vec{X}_{r_4^i}(t)).$

"DE/rand/2": $\vec{V}_i(t) = \vec{X}_{r_1^i}(t) + F_1.(\vec{X}_{r_2^i}(t) - \vec{X}_{r_3^i}(t)) + F_2.(\vec{X}_{r_4^i}(t) - \vec{X}_{r_5^i}(t)).$

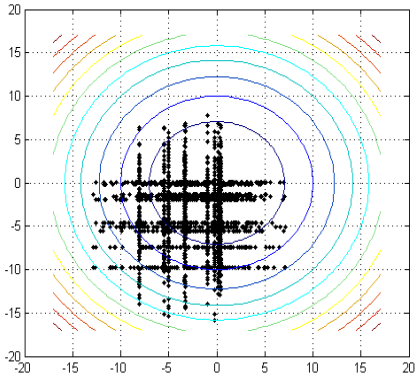
The general convention used for naming the various mutation strategies is DE/x/y/z, where DE stands for Differential Evolution, x represents a string denoting the vector to be perturbed, y is the number of difference vectors considered for perturbation of x, and z stands for the type of crossover being used (exp: exponential; bin: binomial)

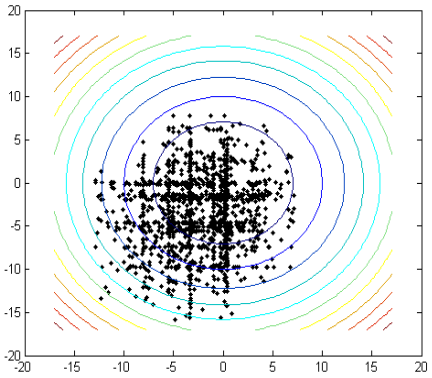
The Crossover Rate *Cr*:

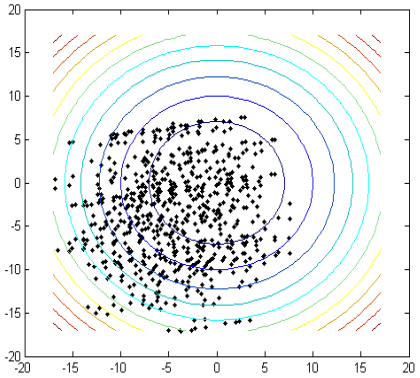
1) The parameter *Cr* controls how many parameters in expectation, are changed in a population member.
2) Low value of *Cr*, a small number of parameters are changed in each generation and the stepwise movement tends to be orthogonal to the current coordinate axes → Good for separable problems

3) High values of *Cr* (near 1) cause most of the directions of the mutant vector to be inherited prohibiting the generation of axis orthogonal steps → Good for non-separable problems

Empirical distribution of trial vectors for three different values of *Cr* has been shown. The plots were obtained by obtained by running DE on a single starting population of 10 vectors for 200 generations with selection disabled.


(a) *Cr* = 0


(b) *Cr* = 0.5


(c) *Cr* = 1.0

For schemes like DE/rand/1/bin the performance is rotationally invariant only when *Cr* = 1.

At that setting, crossover is a vector-level operation that makes the trial vector a pure mutant i.e.

$$\vec{U}_{i,G} = \vec{X}_{r_1^i,G} + F \cdot (\vec{X}_{r_2^i,G} - \vec{X}_{r_3^i,G}).$$

Overview

I. Introduction to Real Variable Optimization & DE

II. Future of Real Parameter Optimization

III. Single Objective Optimization

IV. Multimodal Optimization

V. Dynamic Optimization

VI. Multi-objective Optimization

VII. Large Scale Optimization

VIII. Constrained Optimization

Importance of Population Topologies

- In population based algorithms, population members exchange information between them.
- Single population topology permits all members to exchange information among themselves – the most commonly used.
- Other population topologies have restrictions on information exchange between members – the oldest is island model
- Restrictions on information exchange can slow down the propagation of information from the best member in the population to other members (i.e. single objective global optimization)
- Hence, this approach
  - slows down movement of other members towards the best member(s)
  - Enhances the exploration of the search space
  - Beneficial when solving multi-modal problems

As global version of the PSO converges fast, many topologies were Introduced to slow down PSO

PSO with Neighborhood Operator

Presumed to be the oldest paper to consider distance based neighborhoods for real-parameter optimization.

Lbest is selected from the members that are closer (w.r.t. Euclidean distance) to the member being updated.

Initially only a few members are within the neighborhood (small distance threshold) and finally all members are in the n'hood.

Island model and other static/dynamic neighborhoods did not make use of Euclidean distances, instead just the indexes of population members.

Our recent works are extensively making use of distance based neighborhoods to solve many classes of problems.

P. N. Suganthan, "Particle swarm optimizer with neighborhood operator," in *Proc. Congr. Evol. Comput.*, Washington, DC, pp.1958–1962, **1999**.

Two Subpopulations with Heterogeneous Ensembles & Topologies

- Proposed for balancing exploration and exploitation capabilities
- Population is divided into exploration / exploitation sub-poplns
  - Exploration Subpopulation group uses exploration oriented ensemble of parameters and operators
  - Exploitation Subpopulation group uses exploitation oriented ensemble of parameters and operators.

- Topology allows information exchange only from explorative subpopulation to exploitation sub-population. Hence, diversity of exploration popln not affected even if exploitation popln converges.

- The need for memetic algorithms in real parameter optimization: Memetic algorithms were developed because we were not able to have an EA or SI to be able to perform both exploitation and exploration simultaneously. This 2-popln topology allows with heterogeneous information exchange.

Two Subpopulations with Heterogeneous Ensembles & Topologies

- Sa.EPSDE realization (for single objective Global):

N. Lynn, R Mallipeddi, P. N. Suganthan, “Differential Evolution with Two Subpopulations," LNCS 8947, SEMCCO 2014.

- 2 Subpopulations CLPSO (for single objective Global)

N. Lynn, P. N. Suganthan, “Comprehensive Learning Particle Swarm Optimization with Heterogeneous Population Topologies for Enhanced Exploration and Exploitation,” *Swarm and Evolutionary Computation*, 2015.

- Neighborhood-Based Niching-DE: Distance based neighborhood forms local topologies while within each n’hood, we employ exploration-exploitation ensemble of parameters and operators.

S. Hui, P N Suganthan, “Ensemble and Arithmetic Recombination-Based Speciation Differential Evolution for Multimodal Optimization,” IEEE T. Cybernetics, Online since Mar 2015. 10.1109/TCYB.2015.2394466

Ensemble Methods

- Ensemble methods are commonly used for pattern recognition (PR), forecasting, and prediction, e.g. multiple predictors.

- Not commonly used in Evolutionary algorithms ...

There are two advantages in EA (compared to PR):

1. In PR, we have no idea if a predicted value is correct or not. In EA, we can look at the objective values and make some conclusions.

2. Sharing of function evaluation among ensembles possible.

Adaptations

- Self-adaptation: parameters and operators are evolved by coding them together with solution vector
- Separate adaptation based on performance: operators and parameter values yielding improved solutions are rewarded.
- 2nd approach is more successful and frequently used in DE.

Overview

I. Introduction to Real Variable Optimization & DE

II. Future of Real Parameter Optimization

III. Single Objective Optimization

IV. Multimodal Optimization

V. Dynamic Optimization

VI. Multi-objective Optimization

VII. Large Scale Optimization

VIII. Constrained Optimization

The 'jDE' Algorithm (Brest et al., 2006)

- **Control parameters *F* and *Cr* of the individuals are adjusted by introducing two new parameters τ_1 and τ_2**
- **The new control parameters for the next generation are computed as follows:**

$$F_{i,G+1} = F_l + rand_1 * F_u \quad \text{if} \quad rand_2 < \tau_1$$
$$= F_{i,G} \quad \textbf{else.}$$
$$Cr_{i,G+1} = rand_3 \quad \text{if} \quad rand_4 < \tau_2$$
$$= Cr_{i,G} \quad \textbf{else,}$$

$$\tau_1 = \tau_2 = 0.1 \quad F_l = 0.1,$$

The new *F* takes a value from [0.1, 0.9] while the new *Cr* takes a value from [0, 1].

J. Brest, S. Greiner, B. Bošković, M. Mernik, and V. Žumer, "Self-adapting Control parameters in differential evolution: a comparative study on numerical benchmark problems," *IEEE Trans. on Evolutionary Computation*, Vol. 10, Issue 6, pp. 646 – 657, 2006

Self-Adaptive DE (SaDE) (Qin *et al*., 2009)

- Includes both **control parameter adaptation** and **strategy adaptation**

Strategy Adaptation:

Four effective trial vector generation strategies: DE/rand/1/bin, DE/rand-to-best/2/bin, DE/rand/2/bin and DE/current-to-rand/1 are chosen to constitute a strategy candidate pool.

For each target vector in the current population, one trial vector generation strategy is selected from the candidate pool according to the probability learned from its success rate in generating improved solutions (that can survive to the next generation) within a certain number of previous generations, called the *Learning Period* (*LP*).

First published in CEC 2005

A. K. Qin, V. L. Huang, and P. N. Suganthan, Differential evolution algorithm with strategy adaptation for global numerical optimization", *IEEE Trans. on Evolutionary Computation,* 13(2):398-417, April, 2009.

SaDE (Contd..)

Control Parameter Adaptation:

1) ***NP*** **is left as a user defined parameter.**
2) **A set of *F* values are randomly sampled from normal distribution *N*(0.5, 0.3) and applied to each target vector in the current population.**
3) ***CR*** **obeys a normal distribution with mean value CR_m and standard deviation *Std* =0.1, denoted by $N(CR_m, Std)$ where CR_m is initialized as 0.5.**
4) **SaDE gradually adjusts the range of *CR* values for a given problem according to previous *CR* values that have generated trial vectors successfully entering the next generation.**

Opposition-based DE (Rahnamayan *et al.*, 2008)

- **Three stage modification to original DE framework based on the concept of *Opposite Numbers* :**

  Let x be a real number defined in the closed interval [a, b]. Then the opposite number of x may be defined as:

$$\breve{x} = a + b - x$$

ODE Steps:

1) Opposition based Population Initialization: Fittest NP individuals are chosen as the starting population from a combination of NP randomly generated population members and their opposite members.

2) **Opposition Based Generation Jumping:** In this stage, after each iteration, instead of generating new population by evolutionary process, the opposite population is calculated with a predetermined probability *Jr* () and the *NP* fittest individuals may be selected from the current population and the corresponding opposite population.

ODE (Contd.)

3) Opposition Based Best Individual Jumping: In this phase, at first a difference-offspring of the best individual in the current population is created as:

$$\vec{X}_{new\_best,G} = \vec{X}_{best,G} + F'.(\vec{X}_{r_1,G} - \vec{X}_{r_2,G})$$

where r_1 and r_2 are mutually different random integer indices selected from {1, 2, ..., *NP*} and *F'* is a real constant. Next the opposite of offspring is generated as $\vec{X}_{opp\_newbestG}$. Finally the current best member is replaced by the fittest member of the set $\{\vec{X}_{best,G}, \vec{X}_{new\_best,G}, \vec{X}_{opp\_newbest,G}\}$

S. Rahnamayan, H. R. Tizhoosh, and M. M. A. Salama, "Opposition-based differential evolution", *IEEE Trans. on Evolutionary Computation*, Vol. 12, Issue 1, pp. 64-79, Feb. 2008.

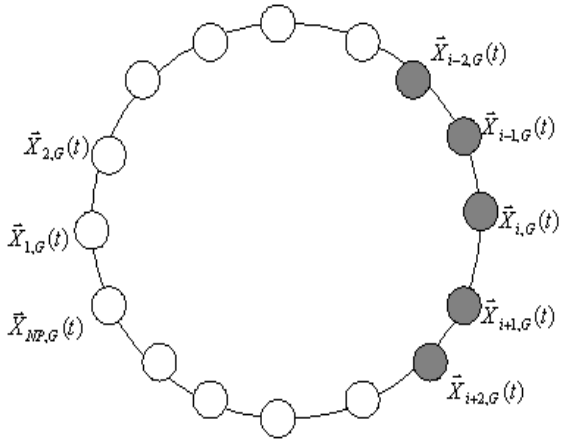
➢ Introduction of Topological Neighborhood-based mutations in Differential Evolution:

Index-based social neighborhoods in DE for global optimization:

- **Removes the exploitative bias of the classical DE mutation scheme:**

$$\vec{V}_i(t) = \vec{X}_i(t) + F.(\vec{X}_{best}(t) - \vec{X}_i(t)) + F.(\vec{X}_{r_1^i}(t) - \vec{X}_{r_2^i}(t)),$$

Initially investigated only ring topologies:



- **Reduces attractions towards specific points**
- **Increases information sharing and promotes explorative search.**

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:

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

Indexed based topologies from PSO domain are faster than Euclidean distance based. But not very effective.

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

JADE (Zhang and Sanderson, 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: $\vec{V}_{i,G} = \vec{X}_{i,G} + F_i \cdot (\vec{X}^{p}_{best,G} - \vec{X}_{i,G}) + F_i \cdot (\vec{X}_{r_1^i,G} - \vec{X}_{r_2^i,G})$

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 $\vec{X}_{r_2^i,G}$ is selected from $P \cup A$

J. Zhang, and A. C. Sanderson, "JADE: Adaptive differential evolution with optional external archive", *IEEE Transactions on Evolutionary Computation*, Vol. 13, Issue 5, Page(s): 945-958, Oct. 2009.

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:

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

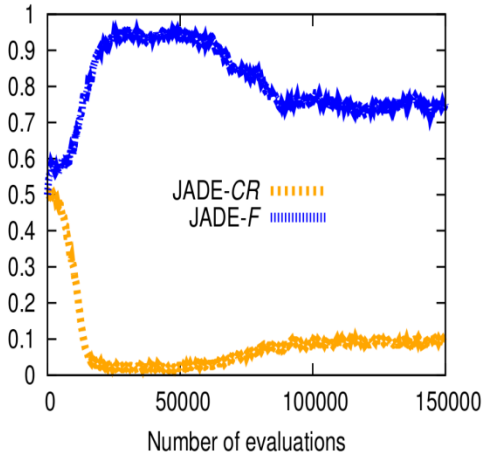
JADE usually performs best with 1/*c* chosen from [5, 20] and *p* from [5%, 20%]

Success-History based Adaptive DE (SHADE)

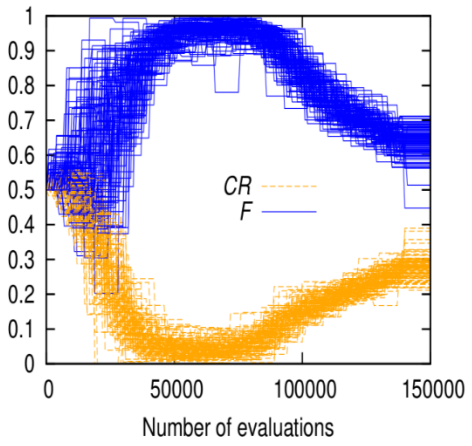
- An improved version of JADE
- Uses a success-history based adaptation
  - Based on a historical memory of successful parameter settings that were previously used found during the run
  - A historical memory M_{CR} , M_F are used, instead of adaptive parameter u_{cr} , u_F
  - This improves the robustness of JADE

Fig. Their adaptation behaviors on Rastrigin (30 dimensions)

JADE uses a single pair u_{cr}, u_F

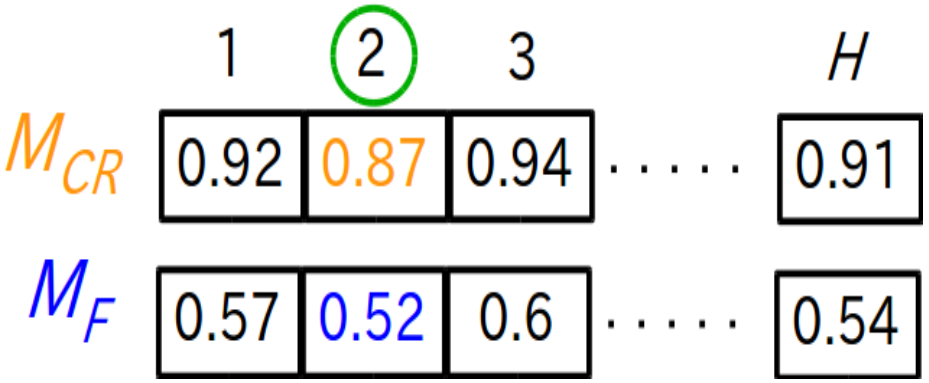


SHADE maintains a diverse set of parameters
in a historical memory M_{CR} , M_F



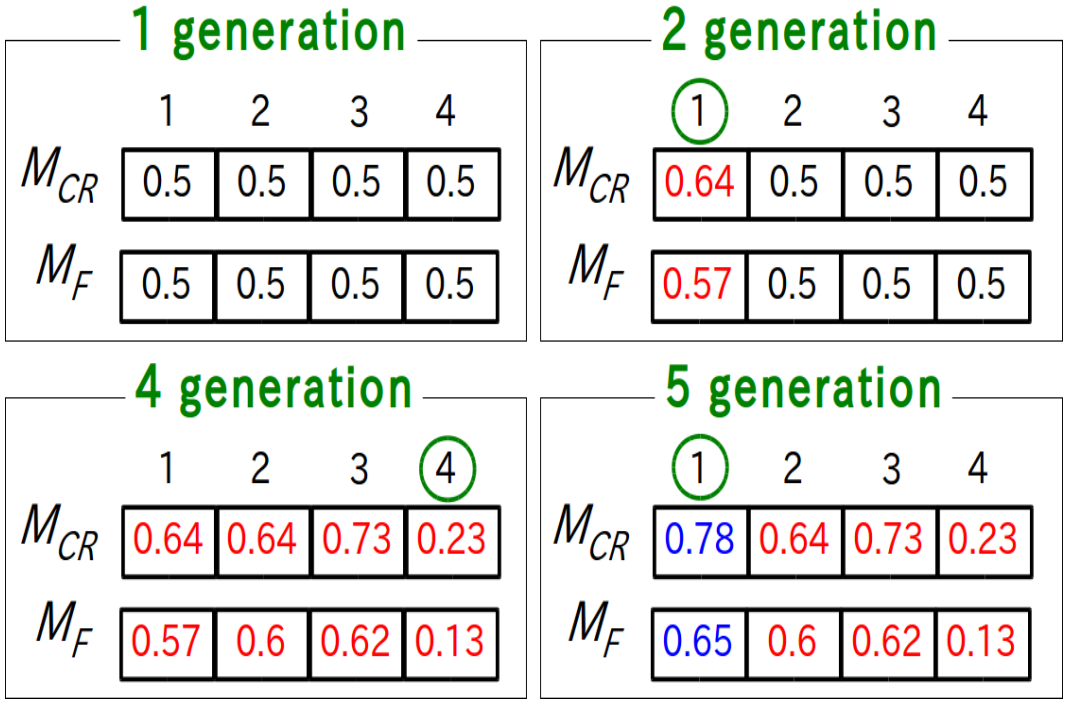
SHADE

- The weighted Lehmer mean (in CEC'14 ver.) values of S_{CR} and S_F , which are successful parameters for each generation, are stored in a historical memory M_{CR} and M_F



- CR_i and F_i are generated by selecting an index r_i randomly from [1, memory size H]
- Example: if selected index $r_i = 2$
  - CR_i = NormalRand(0.87, 0.1)
  - F_i = CauchyRand(0.52, 0.1)

Example of memory update in SHADE



- The contents of both memories are initialized to 0.5 and the index counter is set 1
- The CR_i and F_i values used by successful individuals are recorded in S_{CR} and S_F
- At the end of the generation, the contents of memory are updated by the mean values of S_{CR} and S_F
- The index counter is incremented
- Even if S_{CR} , S_F for some particular generation contains a poor set of values, the parameters stored in memory from previous generations cannot be directly, negatively impacted
- If the index counter exceeds the memory size H, the index counter wraps around to 1 again

Deterministic population reduction methods

- General Policy for Evolutionary Algorithms
  - Explorative search is appropriate for estimating the promising regions
  - Exploitative search is appropriate for finding the higher precision solutions
- Deterministic population reduction methods
  - They use a large population size as initial population and reduce its size
  - This mechanism makes EA more robust and effective.

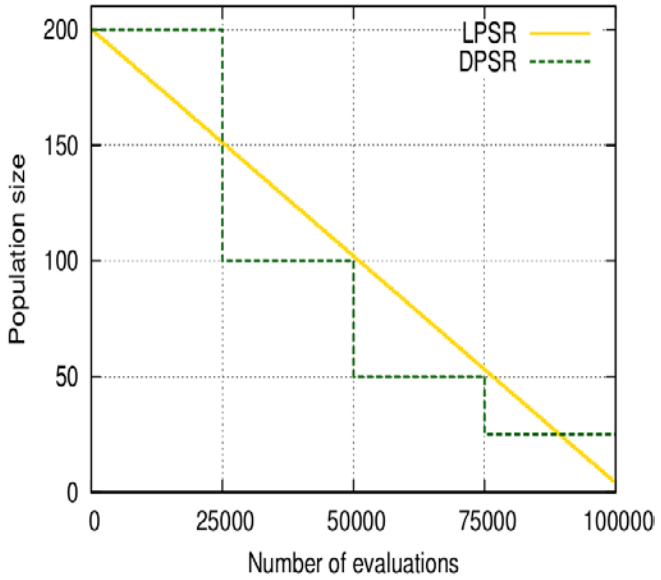
"Evaluating the performance of SHADE on CEC 2013 benchmark problems", Ryoji Tanabe and Alex Fukunaga, The University of Tokyo, Japan (**Codes-Results available, as SHADE_CEC2013)**

L-SHADE in CEC 2014: "Improving the Search Performance of SHADE Using Linear Population Size Reduction," By Ryoji Tanabe and Alex S. Fukunaga

L-SHADE: SHADE with Linear Population Size Reduction

- Deterministic Population Size Reduction (DPSR) [Brest 08]
  - reduces the population by half at predetermined intervals
  - The frequency of the population reduction has to be tuned to match the initial population size as well as the dimensionality of the problem…
- Simple Variable Population Sizing (SVPS) [Laredo 09]
  - is a more general framework in which the shape of the population size reduction schedule is determined according to two control parameters
  - Due to its general versatility, tuning the two control parameters is very hard…
- Linear Population Size Reduction (LPSR) [Tanabe CEC 2014]
  - is a special case of SVPS which reduces the population linearly, and requires only initial population sizes
  - L-SHADE is an extended SHADE with LPSR

Fig. Comparison of population resizing schedule between LPSR and DPSR (# of reduction = 4)



L-SHADE's C++ and Matlab/Octave code can be downloaded from Ryoji Tanabe's site (https://sites.google.com/site/tanaberyoji/)

SPS-L-SHADE-EIG

- A self-optimization approach and a new success history based adaptive differential evolution with linear population size reduction (L-SHADE) which is incorporated with an eigenvector-based (EIG) crossover and a successful-parent selecting (SPS) framework

SPS-L-SHADE-EIG

- The EIG crossover is a rotationally invariant operator
  - Very good performance on numerical optimization problems with highly correlated variables
- The SPS framework provides an alternative of the selection of parents to prevent the situation of stagnation

SPS-L-SHADE-EIG

- The experiment evaluates the performance of the self-optimized SPS-L-SHADEEIG in CEC 2015 real-parameter single objective optimization competition. Winner of CEC 2015

- The parameters of SPS-L-SHADE-EIG are self optimized in terms of each function under IEEE Congress on Evolutionary Computation (CEC) benchmark set in 2015

LSHADE-Epsin

- JADE → SHADE → L-SHADE → LSHADE-EpSin (joint winner in CEC'16)

- An enhanced version of L-SHADE using:
  - An ensemble sinusoidal parameter adaptation and
  - A local search based on a Gaussian walk

- In this ensemble approach, a pool of two different sinusoidal adjustments along with Cauchy distribution are used to adapt the scaling factor for each individual

- The proposed approach falls in the first class which uses an ensemble pool of two sinusoidal waves to adapt *F*, and also proposes an adaptive scheme to control the settings of *freq* parameter in the sinusoidal formula.

Noor H. Awad, Mostafa Z. Ali, Ponnuthurai N. Suganthan, Robert G. Reynolds, An Ensemble Sinusoidal Parameter Adaptation incorporated with L-SHADE for Solving CEC2014 Benchmark Problems, in Proc. IEEE Congr. Evol. Comput., WCCI 2016, Jul. Vancouver, Canada

LSHADE-Epsin- Control Parameter Settings

Ensemble of parameter settings to adapt scaling factor $F_{i,g}$:

Sinusoidal pool

- Activated for the first half of generations $g \in [1, \frac{G_{\max}}{2}]$
- Two different sinusoidal adjustments
- The key advantage of using the sinusoidal formula at first generations is its ability to change the search direction
  - Since the parameter value increases and decrease periodically, thanks to the periodicity of the sine function
  - Hence, the efficiency in exploring the search space is enhanced

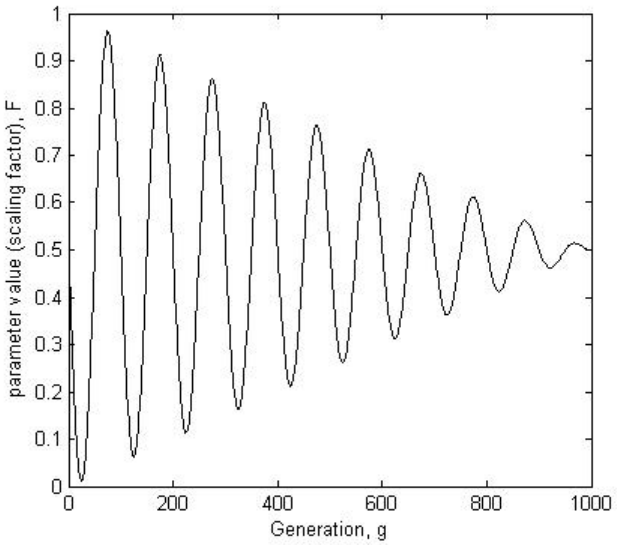
Cauchy distribution

- Activated for the second half of generations $g \in (\frac{G_{\max}}{2}, G_{\max}]$
- As proposed by L-SHADE
- The reason behind using Cauchy at later generations is in its ability to generate values that are best to invoke the exploitation phase
  - Hence, the exploitation phase will be enhanced to search around the best solutions

As a result, our proposed control parameter settings → Provides a good balance between exploration and exploitation

LSHADE-Epsin- Sinusoidal Pool

Non-adaptive Sinusoidal Decreasing Adjustment

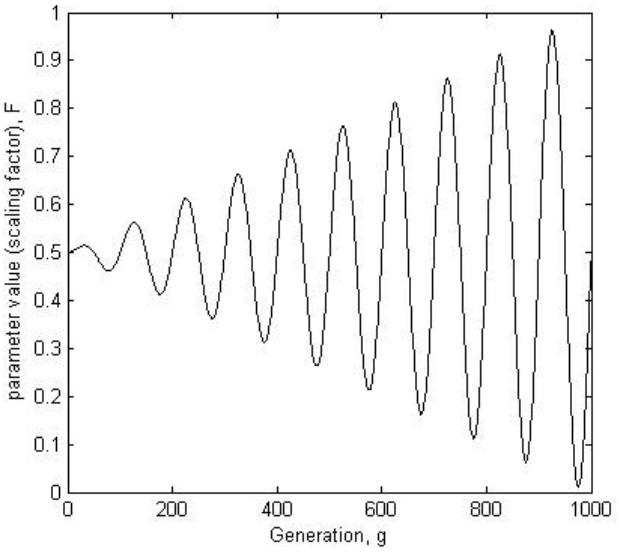


Random Selection

Non-adaptive
Sin Decreasing

Adaptive Sin
Increasing

Adaptive Sinusoidal Increasing Adjustmer



$$F_{i,g} = \frac{1}{2} * \left(\sin(2\pi * freq * g + \pi) * \frac{G_{\max} - g}{G_{\max}} + 1 \right)$$

$$F_{i,g} = \frac{1}{2} * \left(\sin(2\pi * freq_{i,g} * g) * \frac{g}{G_{\max}} + 1 \right)$$

$freq_{i,g}$ is auto-adjusted using a Cauchy distribution as shown in the following equation:

$$freq_{i,g} = Cauchy(\mu_{freq}, 0.1)$$

At each generation, the successful frequencies are stored in S_{freq} and μ_{freq} is updated at the end of each generation using the usual arithmetic mean of S_{freq}

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

- Motivation
  - Empirical guidelines
  - Adaptation/self-adaptation (different variants)
  - Optimization problems (Ex: uni-modal & multimodal)
  - Fixed single mutation strategy & parameters – may not be the best always
- Implementation
  - Contains a pool of mutation strategies & parameter values
  - Compete to produce successful offspring population.
  - Candidate pools must be restrictive to avoid unfavorable influences
  - 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.

Ensemble of Parameters and Mutation and Crossover Strategies in DE (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

$F = \{0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$ $Cr = \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$

ADAPTIVE EPSDE

- Initial population randomly assigned with a mutation & crossover strategies and respective parameters
- Success rate of each parameter or operator is recorded and future usage is proportional to each one's success rate over a few recent past generations.

Differential Evolution With Multi-population Based Ensemble Of Mutation Strategies (MPEDE)

- The most efficient mutation strategy is problem dependent
- Even for one specific problem, the required best mutation strategy may vary during the optimization process
- Different mutation strategy have different exploitation and exploration capabilities and could support each other

Wu, G., Mallipeddi, R., Suganthan, P. N., Wang, R., & Chen, H. (2016). Differential evolution with multi-population based ensemble of mutation strategies. *Information Sciences*, *329*, 329-345.

MPEDE – Motivation

- Constituent mutation strategies should have respective advantages
- Each constituent mutation strategy has its minimum resources
- The constituent mutation strategy that historically performed well should be rewarded with more resources in the closely successive generations
- Resource is represented by the amount of population taken by one mutation strategy

MPEDE – Implementation

- Three well-investigated mutation strategies are included

"current-to-pbest/1"

$$\mathbf{V}_{i,G} = \mathbf{X}_{i,G} + F \cdot (\mathbf{X}_{pbest,G} - \mathbf{X}_{i,G} + \mathbf{X}_{r_1^i,G} - \tilde{\mathbf{X}}_{r_2^i,G})$$

"current-to-rand/1"

$$\mathbf{U}_{i,G} = \mathbf{X}_{i,G} + K \cdot (\mathbf{X}_{r_1^i,G} - \mathbf{X}_{i,G}) + F \cdot (\mathbf{X}_{r_2^i,G} - \mathbf{X}_{r_3^i,G})$$

"rand/1"

$$\mathbf{V}_{i,G} = \mathbf{X}_{r_1^i,G} + F \cdot (\mathbf{X}_{r_2^i,G} - \mathbf{X}_{r_3^i,G})$$

MPEDE – Implementation

- Each constituent mutation strategy has a minimum population resources named indicator sub-population denoted by $\boldsymbol{pop}_1$, $\boldsymbol{pop}_2$ and $\boldsymbol{pop}_3$, respectively.
- Recently best performed mutation strategy is rewarded by an extra population resources named reward sub-population denoted by $\boldsymbol{pop}_4$

Remarks:

- Three indicator sub-populations have relatively small size
- The reward sub-population has larger size.

MPEDE – Implementation

- At the beginning, $\boldsymbol{pop}_1$, $\boldsymbol{pop}_2$ and $\boldsymbol{pop}_3$, are assigned to three constituent mutation strategies, respectively.
- $\boldsymbol{pop}_4$ is randomly assigned to one constituent mutation strategy.
- After every ***ng*** generations, the performance of each mutation strategy is evaluated by the metric

$$\frac{\Delta f_j}{\Delta FES_j}$$

- The determined best-performed mutation strategy will occupy the reward population resource in the following ***ng*** generations.

MPEDE – Implementation

- The control parameters of each mutation strategy are adapted independently, which is similar to that used in JADE.

We eventually realize that better mutation strategies obtained more computational resources in an adaptive manner during the evolution.

Experiments on CEC 2005 benchmark suit show that MPEDE outperforms several other peer DE variants including JADE, jDE, SaDE, EPSDE, CoDE and SHADE.

Overview

I. Introduction to Real Variable Optimization & DE

II. Single Objective Optimization

III. Constrained Optimization

IV. Dynamic Optimization

V. Multiobjective Optimization

VI. Large Scale Optimization

VII. Multimodal Optimization

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

- 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

Constrained Optimization

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

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

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.

Constraint Handling in DE

- Superiority of Feasible (SF)

  Among X_i and X_j, X_i is regarded superior to X_j if :

  - Both infeasible & $\upsilon(X_i) < \upsilon(X_j)$

    push infeasible solutions to feasible region

  - Both feasible & $f(X_i) < f(X_j)$ (minimization problems)

    improves overall solution

  - X_i - feasible & X_j – infeasible

Constraint Handling in DE

- Self-adaptive Penalty

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

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

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

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

$f_{\min}, f_{\max}$ - min. & max. of $f(X)$ irrespective of feasibility

$$r_f=\frac{\#\text{ feasible individuals}}{\text{population size}}$$

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

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

- Amount of penalties - controlled by # of feasible individuals present
- Few feasible – high penalty added to infeasible individuals with high constraint violation.
- More feasible – high penalty added to feasible individuals with high fitness values
- Switch - more feasible - optimal solution

Constraint Handling in DE

- Epsilon Constraint (EC)
  - Relaxation of constraints is controlled by ε parameter
  - High quality solutions for problems with equality constraints

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

X_θ - top θ th individual in initial population (sorted w. r. t. υ)

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

  - The recommended parameter settings are

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

Constraint Handling in DE

- Stochastic Ranking (SR)
  - Balances between objective and υ stochastically (low computational cost)

Basic form of SR

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 Techniques (ECHT)

- No free lunch theorem (NFL)
- Each constrained problem is unique

  (feasible /search space, multi-modality and nature of constraint functions)
- Evolutionary algorithms are stochastic in nature.

  (same problem & algorithm – diff. constraint handling methods - evolution paths can be diff.)
- Diff. stages– Diff. constraint handling methods effective

  (feasible/ search space, multi-modality, nature of constraints, chosen EA)
- To solve a particular problem - numerous trial-and-error runs

  (suitable constraint handling technique and to fine tune associated parameters)

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

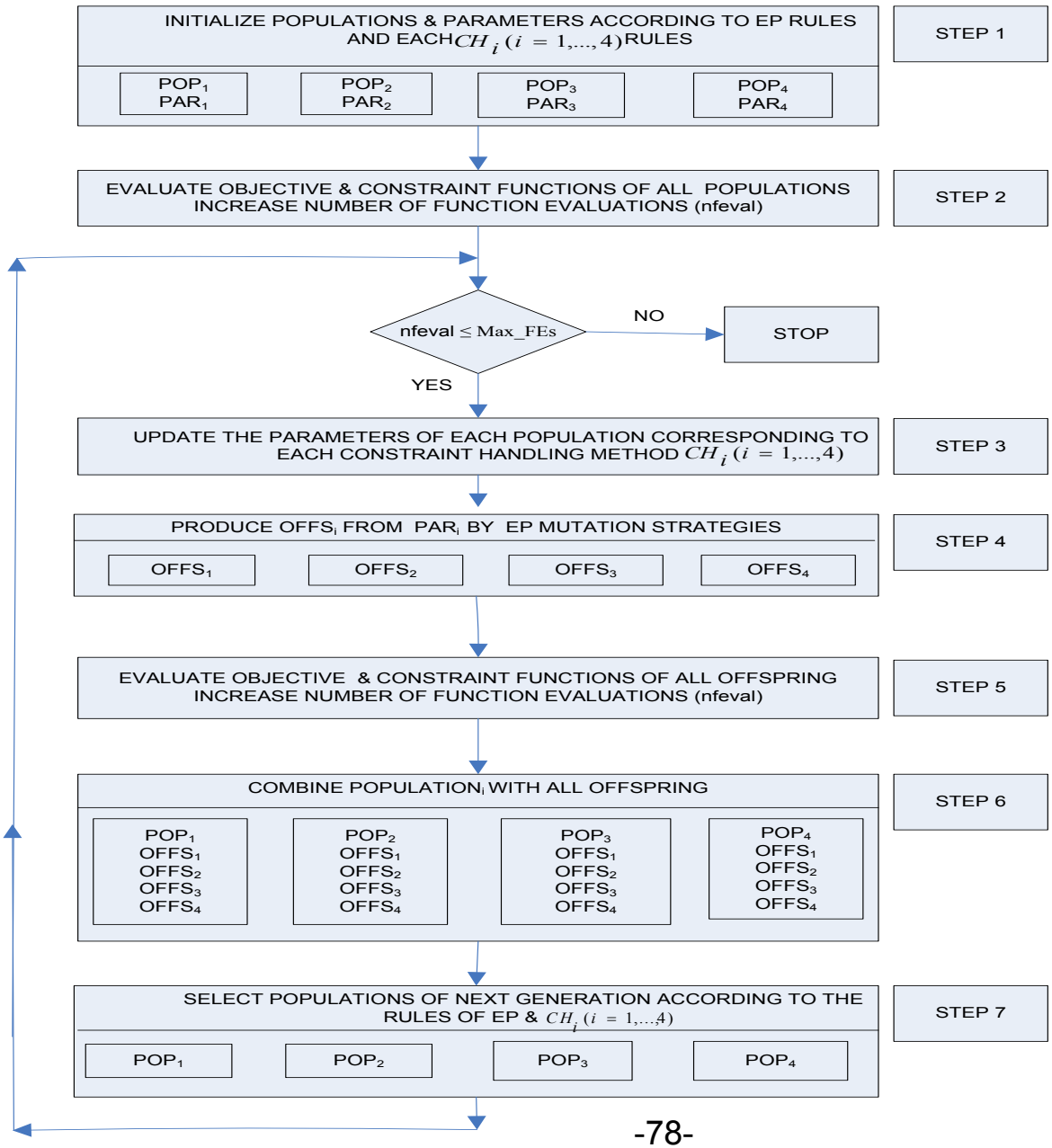
ECHT: MOTIVATION

- 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: MOTIVATION

- 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.

ECHT: Flowchart



ECHT

- Efficient usage of function calls

  (evaluation of objective / constraint functions is computationally expensive)

- Offspring of best suited constraint handling technique survive

  (For a search method and problem during a point in the search process, the offspring population produced by the population of the best suited constraint handling method dominates and enters other populations. In subsequent generations, these superior offspring will become parents in other populations too)

- No trial and error search

- Performance of ECHT can be improved by selecting diverse and competitive constraint handling methods

  (If the constraint handling methods in ensemble are similar in nature populations associated may lose diversity and the search ability of ECHT may be deteriorated)

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: Implementation

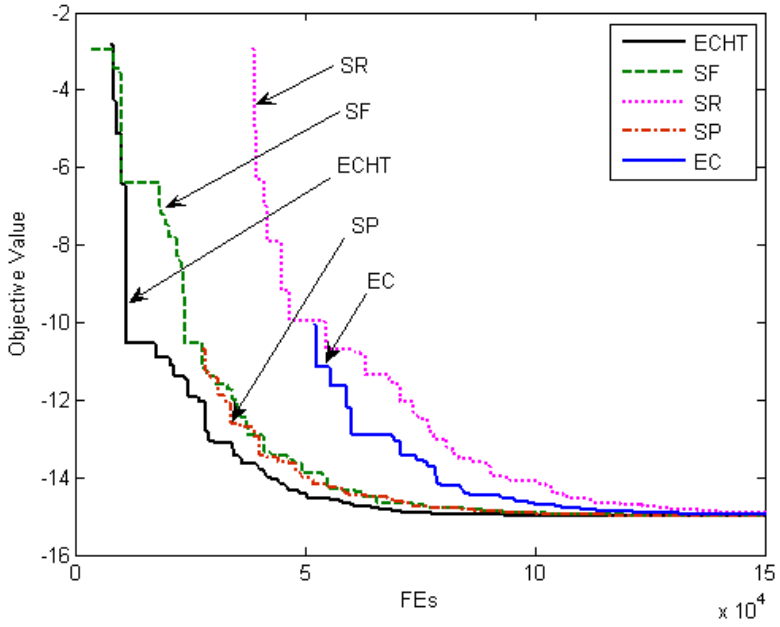
- The constraint handling methods used in the ensemble are

1. Superiority of Feasible (SF)
2. Self-Adaptive penalty (SP)
3. Stochastic Ranking (SR)
4. Epsilon Constraint handling (EC)

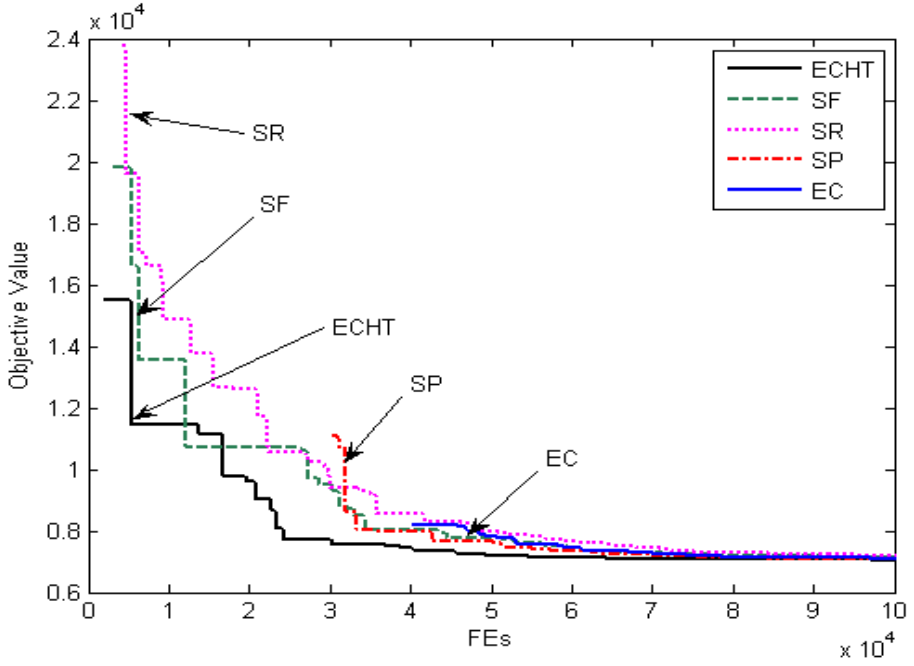
Detailed Results in:

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

ECHT: Results



Problem G01



Problem G10

Convergence Plots

Variable reduction strategy (VRS)

- Although EAs can treat optimization problems as black-boxes (e.g., academic benchmarks), evidences showing that the exploitation of specific problem domain knowledge can improve the problem solving efficiency.
- Technically, optimization can be viewed a process that an algorithm act on a problem. To promote this process, we can
  - Enhance the capability of optimization algorithms,
  - Make use of the domain knowledge hidden in the problem to reduce its complexity.
- We may think
  - Whether there exists general domain knowledge?
  - How to use such knowledge?

Guohua Wu , Witold Pedrycz, P. N. Suganthan, R. Mallipeddi, "A Variable Reduction Strategy for Evolutionary Algorithms Handling Equality Constraints," *Applied soft computing*, 37 (2015): 774-786

VRS: Motivation

- We utilize the domain knowledge of equality optimal conditions (EOCs) of optimization problems.
  - EOCs are expressed by equation systems;
  - EOCs have to be satisfied for optimal solutions;
  - EOCs are necessary conditions;
  - EOCs are general (e.g. equality constraints in constrained optimization and first derivative equals to zero in unconstrained optimization problem with fist-order derivative).
- Equality constraints are much harder to be completely satisfied when an EA is taken as the optimizer.
- The equality constraints of constrained optimization problems (COPs) are treated as EOCs to reduce variables and eliminate equality constraints.

VRS: Motivation

- 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: number of inequality constraints & m-q : the number of equality constraints.

- When using EAs to solve COP, the equality constraints are converted into inequality constraints as:

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

- To obtain highly feasible solutions, we need small ε. However, evidences show that a too small ε makes it harder to find feasible solutions, let alone high-quality ones.

VRS: Implementation

Assume Ω_j denotes the collection of variables involved in equality constraint $h_j(X) = 0$

From $h_j(X) = 0$ $(1 \le j \le m)$, if we can obtain a relationship

$$x_k = R_{k,j}(\{x_l \mid l \in \Omega_j, l \neq k\})$$

x_k can be actually calculated by relationship $R_{k,j}$ and the values of variables in $\{x_l \mid l \in \Omega_j, l \neq k\}$

Moreover, equality constraint $h_j(X)$ is always satisfied.

As a result, both $h_j(X)$ and x_k are eliminated.

VRS: Implementation

- **Some essential concepts:**
  - **Core variable(s):** The variable(s) used to represent other variables in terms of the variable relationships in equality constraints.
  - **Reduced variable(s):** The variable(s) expressed and calculated by core variables.
  - **Eliminated equality constraint(s):** The equality constraint(s) eliminated along with the reduction of variables due to full satisfaction by all solutions.

The aim of the variable reduction strategy is to find a set of **core variables** with minimum cardinality, such that maximum number of **equality constraints and variables** are reduced.

VRS: Implementation

Minimize: $f(X)$

Subject to: $g_i(X) \le 0, \quad i = 1,\ldots,p$

$h_j(X) = 0, \quad j = 1,\ldots,m$

$l_k \le x_k \le u_k \quad k = 1,\ldots,n$

Variable reduction operate

Minimize: $f(X)$

Subject to: $g_i(X) \le 0, \qquad i = 1,\ldots,p$

$h_j(X_k \mid k \in C) = 0, \qquad j \in M_2$

$l_k \le R_{k,j}(\{x_l \mid l \in \Omega_j, l \ne k\}) \le u_k, \quad k \in C_1 \quad j \in M_1$

$l_k \le x_k \le u_k, \qquad k \in C_2$

VRS: Implementation

- Naïve example, considering:

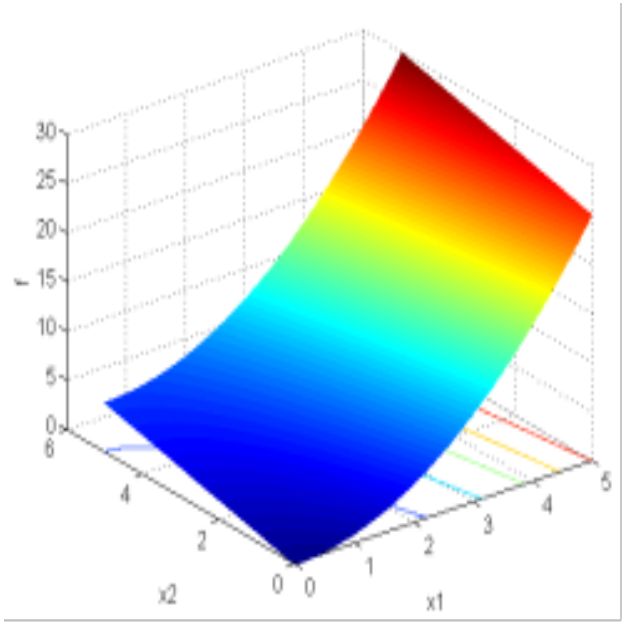
$$\min x_1^2 + x_2^2$$
$$x_1 + x_2 = 2$$
$$0 \le x_1 \le 5, 0 \le x_2 \le 5$$

- We can obtain the variable relationship $x_2 = 2 - x_1$ and substitute it into original problem, then we get

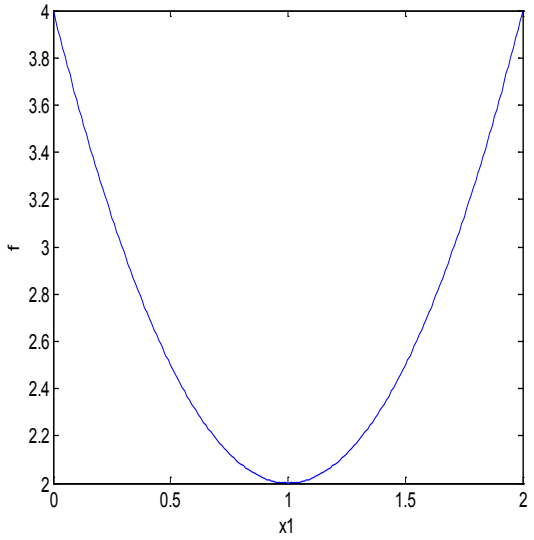
$$\min 2x_1^2 - 4x_1 + 4$$
$$0 \le x_1 \le 2$$

VRS: Implementation

- Solution space before and after the variable reduction process



(a) Original solution space



(b) Solution after VRS.

VRS: Implementation

- Since equality constraints can be nonlinear and very complex, it is still an open to design a unified method for variable reduction.
- Empirically, we can reduce in following situations
  - One variable less than or equal to second-order;
  - All linear equality constraints and variables;
  - Variables separately operated only by one operator (e.g. x^n,$\cos x$, $\ln x$, a^x).

VRS: Implementation

- A formal method automatically reducing linear equality constraint and variables.
- Matrix form of linear equality constraint

$$AX = b$$

- Expand it and we get

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n &= b_2 \\ \vdots \qquad\quad & \vdots \qquad\quad \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n &= b_m \end{aligned} \qquad m < n$$

- We can transform the expanded form into

$$a_{11}x_1 + a_{12}x_2 + \ldots + a_{1m}x_m = b_1 - (a_{1,m+1}x_{m+1} + a_{1,m+2}x_{m+2} + \ldots + a_{1,n}x_n)$$
$$a_{21}x_1 + a_{22}x_2 + \ldots + a_{2m}x_m = b_2 - (a_{2,m+1}x_{m+1} + a_{2,m+2}x_{m+2} + \ldots + a_{2,n}x_n)$$
$$\vdots \qquad\qquad \vdots \qquad\qquad\qquad\qquad \vdots$$
$$a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mm}x_m = b_m - (a_{m,m+1}x_{m+1} + a_{m,m+2}x_{m+2} + \ldots + a_{m,n}x_n)$$

- Let

$$A' = \begin{pmatrix} a_{11} & \cdots & a_{1m} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mm} \end{pmatrix} \qquad X' = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{pmatrix} \qquad b' = \begin{pmatrix} b_1 - (a_{1,m+1}x_{m+1} + a_{1,m+2}x_{m+2} + \ldots + a_{1,n}x_n) \\ b_2 - (a_{2,m+1}x_{m+1} + a_{2,m+2}x_{m+2} + \ldots + a_{2,n}x_n) \\ \vdots \\ b_m - (a_{m,m+1}x_{m+1} + a_{m,m+2}x_{m+2} + \ldots + a_{m,n}x_n) \end{pmatrix}$$

- We have

$$A'X' = b' \quad \Rightarrow \quad X' = (A')^{-1}b'$$

X' is reduced and all linear equality constraints are eliminated.

VRS: Results

- Impact of VRS on the number of variables and equality constraints for Benchmark COPs

Problem		Original COP	COP after ECVRS
g03	Variables	10	9
	Equality Const.	1	0
g05	Variables	4	1
	Equality Const.	3	0
g11	Variables	2	1
	Equality Const.	1	0
g13	Variables	5	2
	Equality Const.	3	0
g14	Variables	10	7
	Equality Const.	3	0
g15	Variables	3	2
	Equality Const.	1	0
g17	Variables	6	2
	Equality Const.	4	0
g21	Variables	7	2
	Equality Const.	5	0
g22	Variables	22	3
	Equality Const.	19	0
g23	Variables	9	5
	Equality Const.	4	0

VRS: Results

Problems		ECHT-DE	ECHT-DE-ECVRS	ECHT-EP	ECHT-EP-ECVRS	SF-DE	SF-DE-ECVRS	EC-EP	EC-EP-ECVRS
g03	Best	-1.0005	**-1.0000**	-1.0005	**-1.0000**	-1.0005	**-1.0000**	-1.0005	**-1.0000**
	Mean	-1.0005	**-1.0000**	-1.0005	**-1.0000**	-1.0005	**-1.0000**	-1.0005	**-1.0000**
	Worst	-1.0005	**-1.0000**	-1.0004	**-1.0000**	-1.0005	**-1.0000**	-1.0005	**-1.0000**
	Std	2.3930e-10	**2.1612e-16**	1.6026e-05	**8.7721e-14**	3.3013e-16	**2.8816e-16**	1.4728e-06	**2.8658e-16**
	Violation	3.6892e-04	**0.0**	2.6684e-04	**0.0**	2.6631e-04	**0.0**	1.0000e-04	**0.0**
g05	Best	5.1265e+03	**5.1265e+03**	5.1265e+03	**5.1265e+03**	5.1265e+03	**5.1265e+03**	5.1265e+03	**5.1265e+03**
	Mean	5.1265e+03	**5.1265e+03**	5.1265e+03	**5.1265e+03**	5.1265e+03	**5.1265e+03**	5.1265e+03	**5.1265e+03**
	Worst	5.1265e+03	**5.1265e+03**	5.1265e+03	**5.1265e+03**	5.1265e+03	**5.1265e+03**	5.1266e+03	**5.1265e+03**
	Std	2.0865e-12	**9.33125e-13**	2.0212e-07	**9.3312e-13**	1.9236e-12	**9.3312e-13**	3.2214e−02	**9.3312e-13**
	Violation	3.0543e-04	**0.0**	4.6534e-04	**0.0**	8.0598e-04	**0.0**	6.0558e-04	**0.0**
g11	Best	7.4990e-01	**7.5000e-01**	7.4990e-01	**7.5000e-01**	7.4990e-01	**7.5000e-01**	7.4990e-01	**7.5000e-01**
	Mean	7.4990e-01	**7.5000e-01**	7.4990e-01	**7.5000e-01**	7.4990e-01	**7.5000e-01**	7.4990e-01	**7.5000e-01**
	Worst	7.4990e-01	**7.5000e-01**	7.4990e-01	**7.5000e-01**	7.4990e-01	**7.5000e-01**	7.4990e-01	**7.5000e-01**
	Std	1.1390e-16	**0.0**	1.1390e-16	**0.0**	1.1390e-16	**0.0**	2.1630e-09	**0.0**
	Violation	1.0989e-04	**0.0**	1.0000e-05	**0.0**	1.0539e-04	**0.0**	9.9999e-05	**0.0**
g13	Best	5.3942e-02	**5.3949e-02**	5.3942e-02	**5.3949e-02**	5.3942e-02	**5.3949e-02**	5.4137e-02	**5.3949e-02**
	Mean	1.3124e-01	**5.3949e-02**	5.3942e-02	**5.3949e-02**	3.5288e-01	**5.3949e-02**	5.4375e-02	**5.3949e-02**
	Worst	4.4373e-01	**5.3949e-02**	5.3942e-02	**5.3949e-02**	4.6384e-01	**5.3949e-02**	5.6346e-02	**5.3949e-02**
	Std	1.5841e-01	**6.3675e-18**	5.3942e-02	**1.5597e-17**	1.4745e-01	**7.9594e-18**	6.3439e-03	**1.2834e-17**
	Violation	3.9935e-04	**0.0**	5.0444e-09	**0.0**	1.0000e-04	**0.0**	2.7237e-02	**0.0**
g14	Best	-4.7765e+01	**-4.7761e+01**	-4.7761e+01	**-4.7761e+01**	-4.7765e+01	**-4.7761e+01**	-4.7765e+01	**-4.7761e+01**
	Mean	-4.7765e+01	**-4.7761e+01**	-4.7703e+01	**-4.7761e+01**	-4.7765e+01	**-4.7761e+01**	-4.5142e+01	**-4.7706e+01**
	Worst	-4.7765e+01	**-4.7761e+01**	-4.7405e+01	**-4.7761e+01**	-4.7765e+01	**-4.7761e+01**	-4.3762e+01	**-4.7361e+01**
	Std	2.1625e-05	**1.6703e-14**	7.8687e-02	**2.9722e-12**	1.8728e-14	**3.7355e-14**	7.3651e-01	**2.1542e-02**
	Violation	2.9212e-04	**0.0**	2.9999e-04	**0.0**	3.0000e-004	**0.0**	2.9999e-04	**0.0**

VRS: Results

g15	Best	9.6172e+02	**9.6172e+02**	9.6172e+02	**9.6172e+02**	9.6172e+02	**9.6172e+02**	9.6233e+02	**9.6172e+02**
	Mean	9.6172e+02	**9.6172e+02**	9.6172e+02	**9.6172e+02**	9.6172e+02	**9.6172e+02**	9.6248e+02	**9.6172e+02**
	Worst	9.6172e+02	**9.6172e+02**	9.6172e+02	**9.6172e+02**	9.6172e+02	**9.6172e+02**	9.6265e+02	**9.6172e+02**
	Std	5.8320e-13	**1.1664e-13**	6.1830e-13	**1.1664e-13**	5.8320e-13	**1.1664e-13**	7.3719e+01	**1.1664e-13**
	Violation	1.9995e-04	**0.0**	1.9999e-04	**0.0**	2.0000e-04	**0.0**	2.1737e-01	**0.0**
g17	Best	8.8535e+03	**8.8535e+03**	8.8535e+03	**8.8535e+03**	8.8535e+03	**8.8535e+03**	9.1573e+03	**8.8535e+03**
	Mean	8.8535e+03	**8.8535e+03**	8.8535e+03	**8.8535e+03**	8.8757e+03	**8.8535e+03**	9.1791e+03	**8.8535e+03**
	Worst	8.8535e+03	**8.8535e+03**	8.8535e+03	**8.8535e+03**	8.9439e+03	**8.8535e+03**	9.2005e+03	**8.8535e+03**
	Std	3.7324e-12	**1.8662e-12**	2.0301e-08	**1.8662e-12**	3.8524e+01	**1.8662e-12**	1.2346e+01	**1.8662e-12**
	Violation	3.2953e-04	**0.0**	2.6943e-04	**0.0**	1.7744e-04	**0.0**	3.1295e-04	**0.0**
g21	Best	1.9372e+02	**1.9379e+02**	1.9372e+02	**1.9379e+02**	1.9372e+02	**1.9379e+02**	1.9872e+02	**1.9379e+02**
	Mean	1.9984e+02	**1.9379e+02**	1.9498e+02	**1.9379e+02**	2.0682e+02	**1.9379e+02**	2.3474e+02	**1.9379e+02**
	Worst	3.1604e+02	**1.9379e+02**	2.0661e+02	**1.9379e+02**	3.2470e+02	**1.9379e+02**	2.7589e+02	**1.9379e+02**
	Std	2.7351e+01	**2.8632e-12**	3.8129e+00	**3.8765e-12**	4.0314e+01	**6.8507e-10**	2.6621e+01	**3.8625e-12**
	Violation	4.0195e-04	**0.0**	4.8585e-04	**0.0**	9.9999e-05	**0.0**	3.0432e-03	**0.0**
g22	Best	1.8857e+03	**2.3637e+02**	3.9184e+02	**2.3637e+02**	3.9643e+03	**2.3637e+02**	2.2545e+03	**2.3637e+02**
	Mean	1.0158e+04	**2.3637e+02**	7.7786e+02	**2.3637e+02**	1.3812e+04	**2.3637e+02**	1.2854e+04	**2.3637e+02**
	Worst	1.7641e+04	**2.3637e+02**	1.4844e+03	**2.3637e+02**	1.9205e+04	**2.3637e+02**	1.6328e+04	**2.3637e+02**
	Std	4.2890e+03	**1.4580e-13**	3.0970e+02	**2.2875e-13**	5.0860e+03	**7.3769e-14**	3.2582e+03	**1.9875e-13**
	Violation	4.1562e+03	**0.0**	2.7186e+03	**0.0**	1.3192e+04	**0.0**	4.156e+03	**0.0**
g23	Best	-3.9072e+02	**-4.0000e+02**	-3.4556e+02	**-4.0000e+02**	-3.9158e+02	**-4.0000e+02**	−3.8625e+02	**-4.0000e+02**
	Mean	-3.6413e+02	**-4.0000e+02**	-3.0952e+02	**-4.0000e+02**	-2.4367e+02	**-4.0000e+02**	−3.4864e+02	**-4.0000e+02**
	Worst	-2.3426e+02	**-4.0000e+02**	-2.5807e+02	**-4.0000e+02**	-1.0004e+02	**-4.0000e+02**	−2.7235e+02	**-4.0000e+02**
	Std	3.4129e+01	**1.1664e-13**	2.5417e+01	**4.8217e-09**	1.9487e+01	**0.0**	2.3654e+01	**1.7496e-13**
	Violation	3.5951e-04	**0.0**	1.7373e-04	**0.0**	2.5635e-02	**0.0**	8.8827e-02	**0.0**

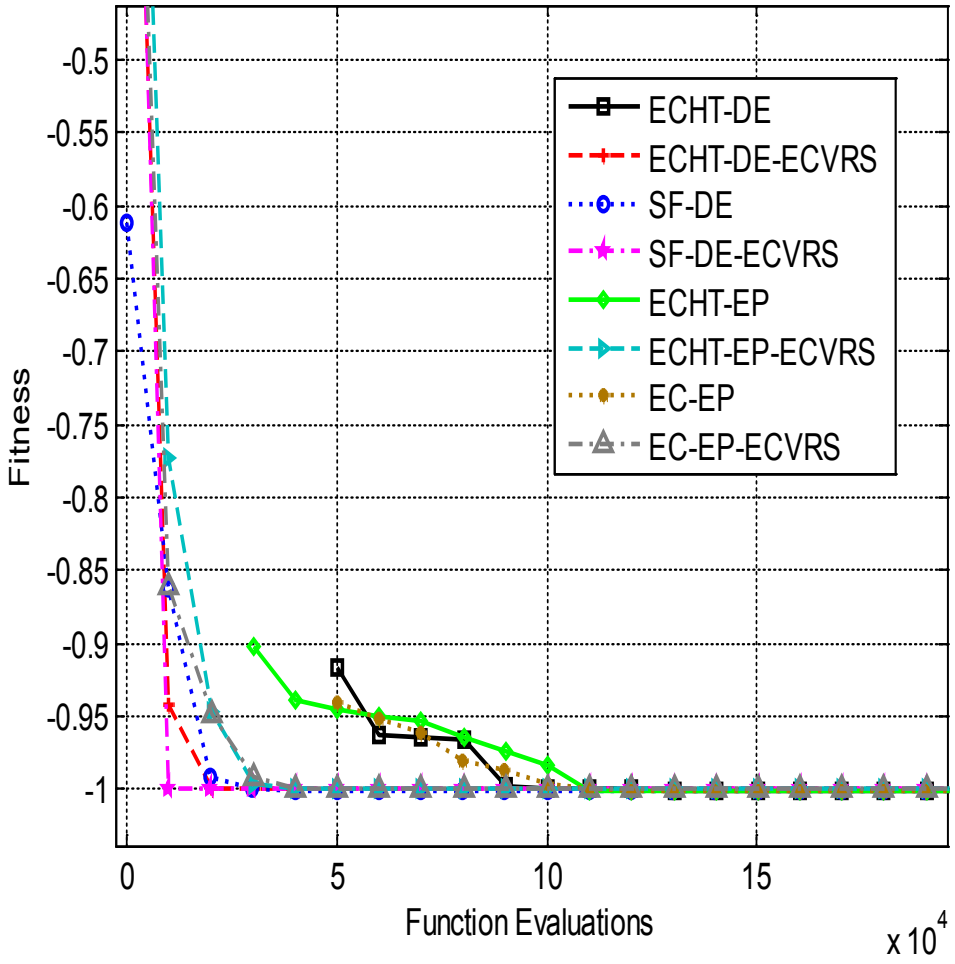
VRS: Results

Number of function objective evaluations required by each EA with or without ECVRS to reach the near optimal objective function values

Problems	ECHT-DE		ECHT-DE- ECVRS		ECHT-EP		ECHT-EP -ECVRS		SF-DE		SF-DE -ECVRS		EC-EP		EC-EP -ECVRS	
	FEs	Suc	FEs	Suc	FEs	Suc	FEs	Suc	FEs	Suc	FEs	Suc	FEs	Suc	FEs	Suc
g03	80160	25	16630	25	118280	25	18540	25	22570	25	6955	25	10250	25	8555	25
g05	109760	25	400	25	119610	25	400	25	27290	25	200	25	12290	25	400	25
g11	36810	25	400	25	120200	25	400	25	13090	25	200	25	121410	25	400	25
g13	89300	18	840	25	126600	25	860	25	140550	5	620	25	56000	15	1040	25
g14	139590	25	23490	25	86720	25	39250	25	46440	25	23650	25	141470	25	82385	25
g15	103460	25	400	25	120200	25	400	25	26750	25	200	25	----	0	400	25
g17	108340	25	400	25	115640	25	400	25	40860	25	200	25	----	0	400	25
g21	107500	22	6160	25	148560	22	8280	25	25870	20	18800	25	13556	12	9580	25
g22	----	0	660	25	----	0	1860	25	----	0	455	25	----	0	4526	25
g23	----	0	28060	25	----	0	37200	25	----	0	650	25	----	0	24630	25

VRS: Results

Illustration of the convergence process of each EA on the benchmark COPs.



VRS: Remarks

- It is generally impossible to exactly solve the equation systems expressing equality optimal conditions of an problem;
- We are not to pursue the exact solution of equality optimal conditions, but to utilize equality optimal conditions to derive variable relationships and exploit them to reduce the problem complexities (e.g., reduce variables and eliminate equality constraints);
- General and theoretical approaches to deal with complex and nonlinear equality optimal conditions are needed.

Overview

I. Introduction to Real Variable Optimization & DE

II. Future of Real Parameter Optimization

III. Single Objective Optimization

IV. Multimodal Optimization

V. Dynamic Optimization

VI. Multi-objective Optimization

VII. Large Scale Optimization

VIII. Constrained Optimization

Dynamic Optimization with DE

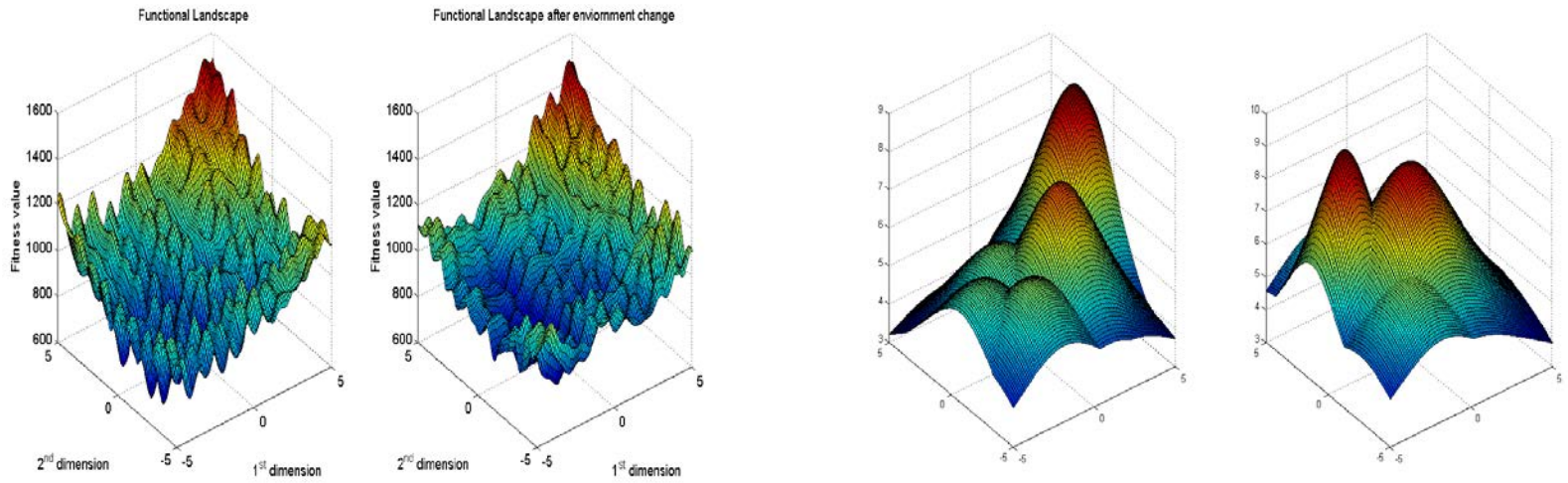
Optima of the problem to be solved shift with time.

Examples: 1) Price Fluctuations

2) Stochastic arrival of new tasks in a scheduling problem

3) Machine breakdown/maintenance during process control

4) Optimal routing with changing traffic conditions



An EA should track the changing optima as closely as possible.

T. T. Nguyen, S. Yang, J. Branke, "**Evolutionary dynamic optimization: A survey of the state of the art**", *Swarm and Evolutionary Computation*, Vol. 6, Oct 2012, pp. 1–24.

Requirements for adopting an EA in Dynamic Environments

- **Boost-up diversity after an environmental change (e.g. hypermutation, variable local search).**
- **Maintain diversity throughout the runs with a hope that a well spread-out population can cope with dynamic changes more easily (e.g. random immigrants, thermodynamic GA).**
- **Memory based approaches (particularly useful when the optimum repeatedly returns to its previous locations).**
- **Multi-population approach – Use of different subpopulations that can maintain information about several promising regions of the search space – act as a distributed and adaptive memory (e.g. multinational GAs)**

Reuse of information from previous generations may help is the environmental change is not too severe.

The GDBG System Benchmarks

-proposed under IEEE CEC 2009 Competition and Special Session on Dynamic optimization

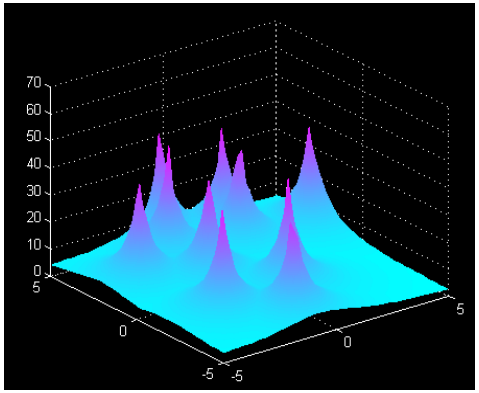
DOP expression: $F = f(\vec{X}, \varphi, t),$

Parametric variation to introduce dynamicity: $\varphi(t+1) = \varphi(t) \oplus \Delta\varphi,$

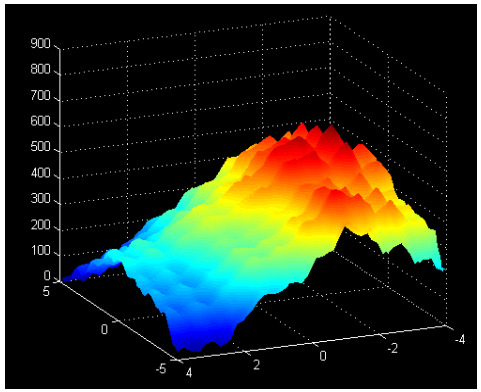
GDBG change types	
T1	Small step: $\Delta\phi = \alpha \cdot \| \phi \| \cdot r \cdot \phi_{severity},$
T2	Large step: $\Delta\phi = \| \phi \| \cdot (\alpha \cdot sign(r) + (\alpha_{\max} - \alpha) \cdot r) \cdot \phi_{severity},$
T3	Random: $\Delta\phi = N(0,1) \cdot \phi_{severity},$
T4	Chaotic: $\phi(t+1) = A \cdot (\phi(t) - \phi_{\min}) \cdot (1 - (\phi(t) - \phi_{\min}) / \| \phi \|),$
T5	Recurrent: $\phi(t+1) = \phi_{\min} + \| \phi \| (\sin(\frac{2\pi}{P} t + \varphi) + 1) / 2,$
T6	Recurrent with noise: $\phi(t+1) = \phi_{\min} + \| \phi \| (\sin(\frac{2\pi}{P} t + \varphi) + 1) / 2 + N(0,1) \cdot noisy_{severity},$
T7	Dimensional change: $D(t+1) = D(t) + sign \cdot \Delta D,$

CEC'09 Dynamic Optimization Benchmark Functions	
F1	Rotation peak function
F2	Composition of Sphere functions
F3	Composition of Rastrigin's functions
F4	Composition of Griewank's functions
F5	Composition of Ackley's functions
F6	Hybrid Composition function

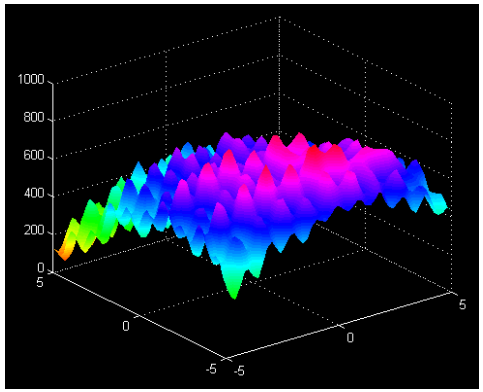
The GDBG System Benchmarks (Contd..)



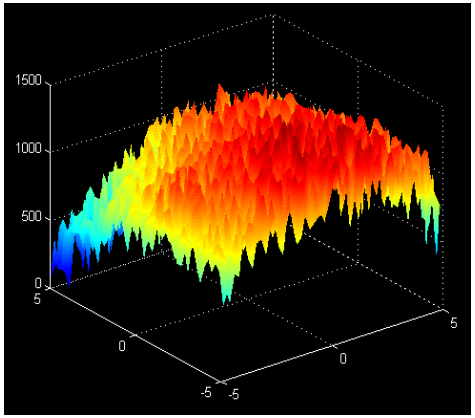
(a) Example plot of Rotation Peak function (F1)



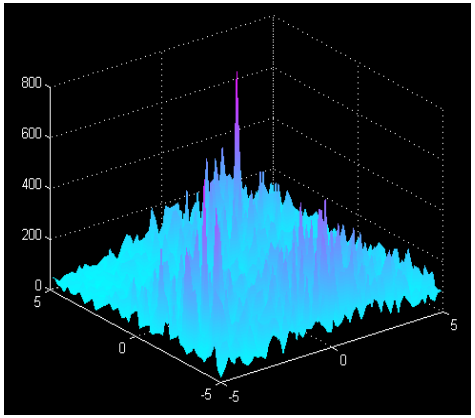
(b) Example plot of Composition of Sphere functions (F2) [inverted]



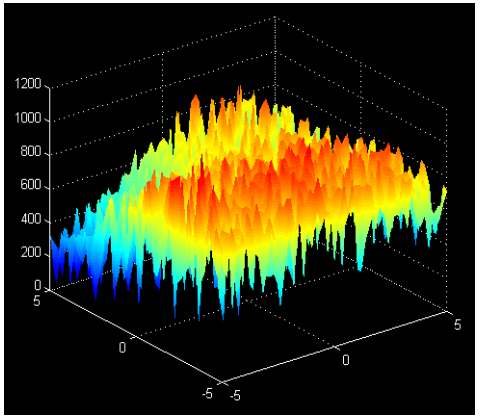
(c) Example plot of Composition of Rastrigin's function (F3) [inverted]



(d) Example plot of Composition of Griewank's function (F4) [inverted]



(e) Example plot of Composition of Ackley's function (F5) [inverted]



(f) Example plot of Hybrid Composition function (F6) [inverted]

Performance Metrics:

Mean offline error: $e_{off} = \frac{1}{R \times K} \sum_{r=1}^{R} \sum_{k=1}^{K} e_{r,k}^{last}$,

Standard deviation: $STD = \sqrt{\frac{\sum_{r=1}^{R} \sum_{k=1}^{K} (e_{r,k}^{last} - e_{off})}{R \times K - 1}}$,

where R = no. of runs, K = no. of environmental changes per run and

$$e_{r,k}^{last} = \left| f(\vec{X}_{best}(r,k)) - f(\vec{X}^{*}(r,k)) \right|$$

Mark obtained by an algorithm in *i*-th test case:

$$mark_i = \frac{w_i}{R \times K} \sum_{r=1}^{R} \sum_{k=1}^{K} \left(\frac{r_{rk}^{last}}{1 + \frac{1}{S} \cdot \sum_{s=1}^{S} (1 - r_{rk}^{s})} \right)$$

where

$$r_{rk}^{s} = \begin{cases} \frac{f(\vec{X}_{best}(r,k,s))}{f(\vec{X}^{*}(r,k))}, & \text{for } f = F1, \\ \frac{f(\vec{X}^{*}(r,k))}{f(\vec{X}_{best}(r,k,s))}, & \text{for } f \in F2, F3, F4, F5, F6. \end{cases}$$

Overall performance of an algorithm calculated as: $performance = 100 \times \sum_{i=1}^{49} mark_i$.

Overview

I. Introduction to Real Variable Optimization & DE

II. Future of Real Parameter Optimization

III. Single Objective Optimization

IV. Multimodal Optimization

V. Dynamic Optimization

VI. Multi-objective Optimization

VII. Large Scale Optimization

VIII. Constrained Optimization

Non-Domination Sorting

- Multi-objective optimization

  Minimize: $F(x) = (f_1(x), f_2(x), ..., f_m(x))^T$

  Subject to: $x_i^L \le x_i \le x_i^U$

- Non-domination sorting: x_1 dominates x_2 if

  The solution x_1 is no worse than x_2 in all objectives.

  The solution x_1 is strictly better than x_2 in at least one objective.
- Most current MOEAs are based on non-domination sorting. The standard DE algorithm (i.e. xover & mutation) is used with different MOEA-tuned selection mechanisms.
- Recently, decomposition method was proposed which decomposes an MOP into a large number of single objective problems.

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.

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 subproblems. Solve these *N* subproblems 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 subproblems.**

x^1 x^2 x^N

Min $g(x,\lambda^1)$ ***Min*** $g(x,\lambda^2)$ ***Min*** $g(x,\lambda^N)$

- **During the search, neighboring agents collaborate with each other.**

MOEA/D framework

- **Agent *i* records x^i, the best solution found so far for this particular subproblem.**
- **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 its best solution by y, if y is better than the current best solution (measured by its own weighted objective).**

ENS-MOEA/D

- **For effective performance of MOEA/D, neighborhood size (NS) parameter has to be tuned:**
  - **A large NS promotes collaboration among dissimilar subproblems, which advances the information exchange among the diverse subproblems, 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 beneficial for diversity recovery; while a small NS is also able to facilitate the convergence:**
  - **For instance, during the evolution, some subproblems may get trapped in a locally optimal regions. In order to force those subproblems escape from the 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, 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 a 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).**

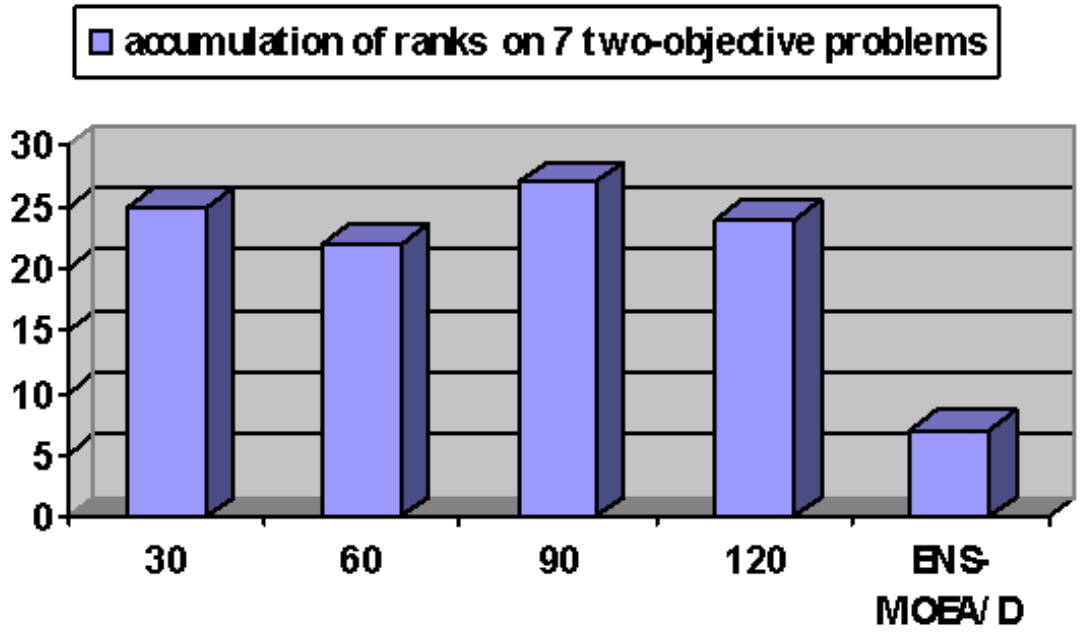
- **The IGD performance measure is used as in the CEC 2009 MOEA competition.**

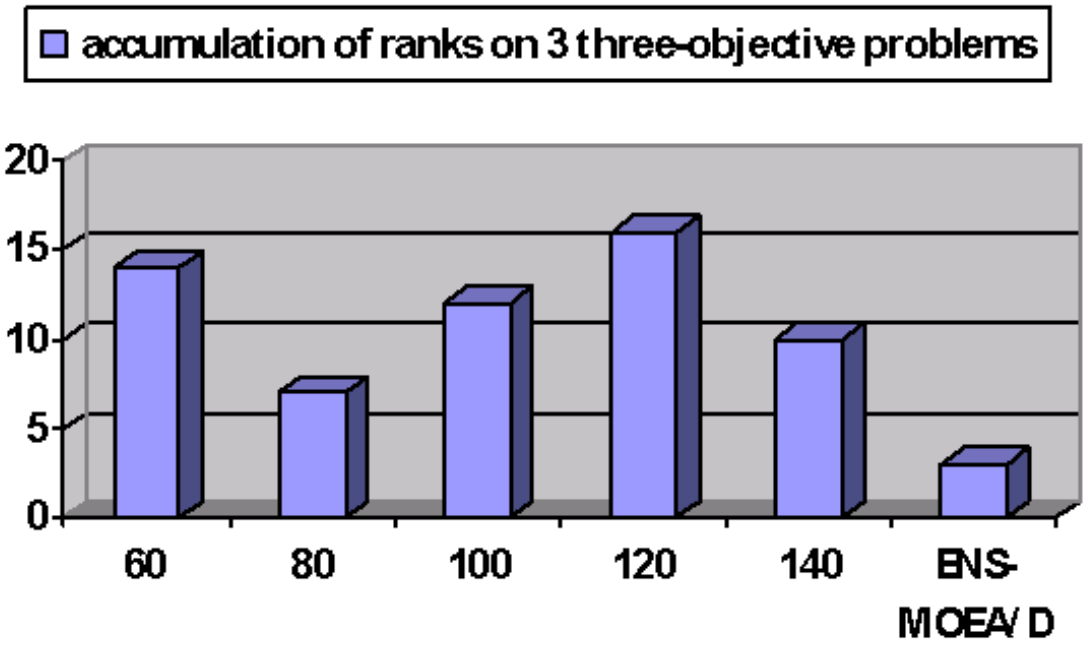
- **The four different NSs for the two-objective problems are 30, 60, 90 and 120, where NS=60 is the original parameter setting in the MOEA/D in the NSs for the three-objective problems are 60, 80, 100, 120 and 140, where 100 is the original parameter setting for NS in the MOEA/D.**

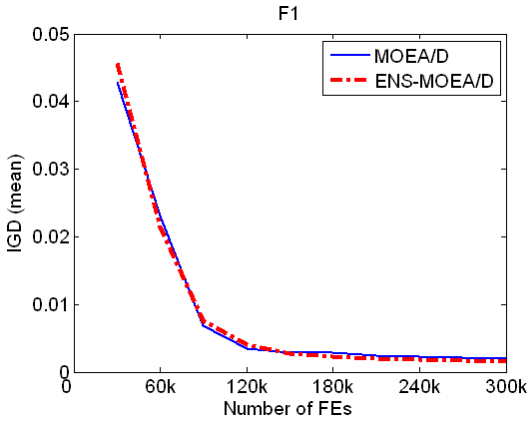
ENS-MOEA/D Experimental results

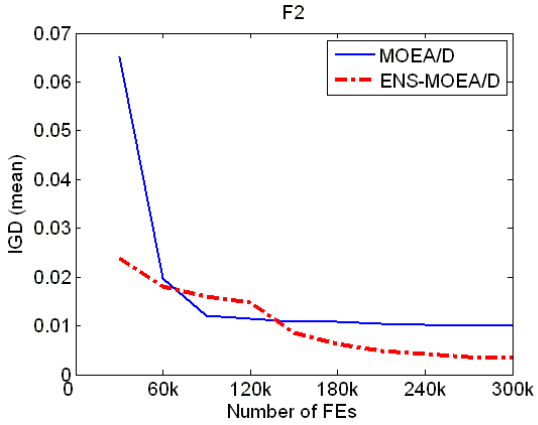
- **We conducted a parameter sensitivity investigation of *LP* for ENS-MOEA/D using four different values (10, 25, 50 and 75) on the 10 benchmark instances. By observing the mean of IGD values over 25 runs we can conclude that the *LP* is not so sensitive to most of the benchmark functions, and it is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

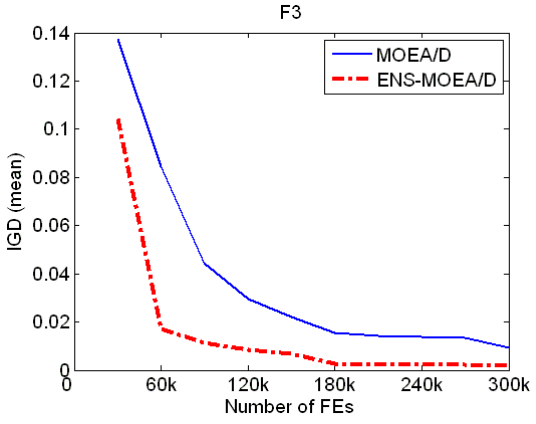
- **The mean of IGD values over 25 runs among all the variants of MOEA/D with different fixed *NS* and ENS-MOEA/D are ranked. Smaller ranks, better performance.**

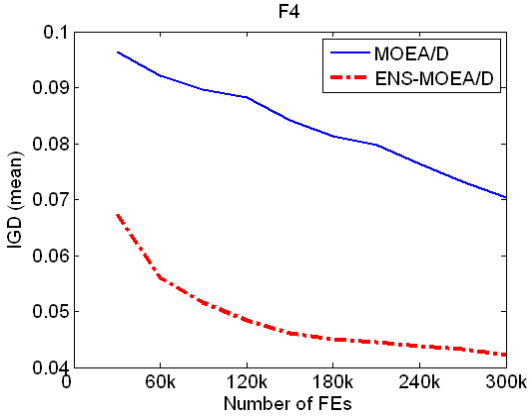

accumulation of ranks on 7 two-objective problems
30
25
20
15
10
5
0
30
60
90
120
ENS-MOEA/D

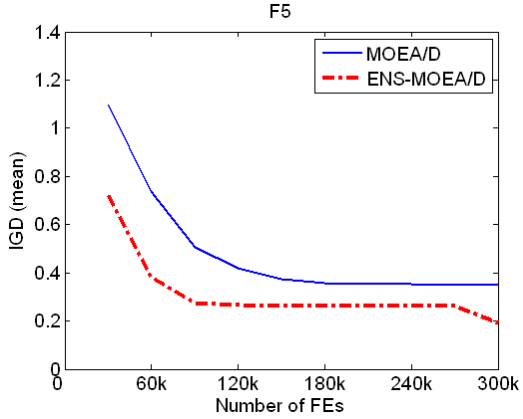

accumulation of ranks on 3 three-objective problems
20
15
10
5
0
60
80
100
120
140
ENS-MOEA/D

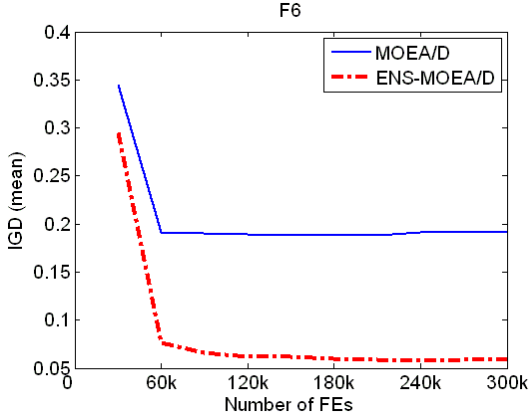

F1
MOEA/D
ENS-MOEA/D
IGD (mean)
Number of FEs
0.05
0.04
0.03
0.02
0.01
0
60k
120k
180k
240k
300k

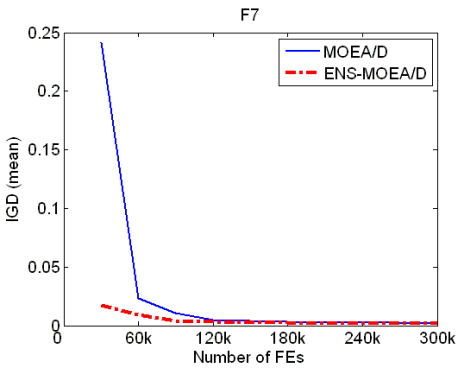

F2
MOEA/D
ENS-MOEA/D
IGD (mean)
Number of FEs
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
60k
120k
180k
240k
300k

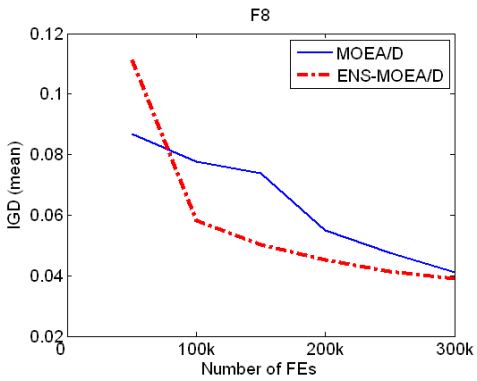

F3
MOEA/D
ENS-MOEA/D
IGD (mean)
Number of FEs
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
60k
120k
180k
240k
300k

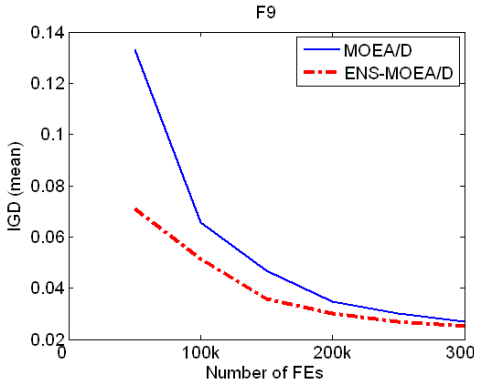

F4
MOEA/D
ENS-MOEA/D
IGD (mean)
Number of FEs
0.1
0.09
0.08
0.07
0.06
0.05
0.04
0
60k
120k
180k
240k
300k

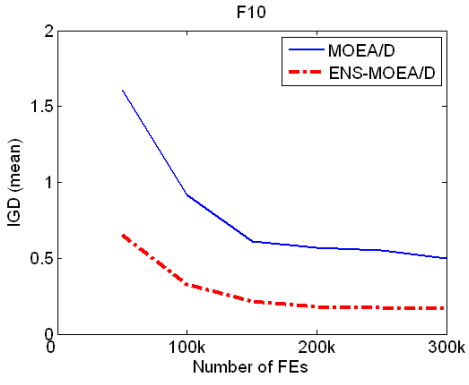

F5
MOEA/D
ENS-MOEA/D
IGD (mean)
Number of FEs
1.4
1.2
1
0.8
0.6
0.4
0.2
0
60k
120k
180k
240k
300k


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

Overview

I. Introduction to Real Variable Optimization & DE

II. Future of Real Parameter Optimization

III. Single Objective Optimization

IV. Multimodal Optimization

V. Dynamic Optimization

VI. Multi-objective Optimization

VII. Large Scale Optimization

VIII. Constrained Optimization

Large Scale Optimization

- **Optimization algorithms perform differently when solving different optimization problems due to their distinct characteristics. Most optimization algorithms lose their efficacy when solving high dimensional problems. Two main difficulties are:**
  - **The high demand on exploration capabilities of the optimization methods. When the solution space of a problem increases exponentially with increasing dimensions, more efficient search strategies are required to explore all promising regions within a given time budget.**
  - **The complexity of a problem characteristics may increase with increasing dimensionality, e.g. unimodality in lower dimensions may become multi-modality in higher dimensions for some problems (e.g. Rosenbrock's)**

Large Scale Optimization

- **Due to these reasons, a successful search strategy in lower dimensions may no longer be capable of finding good solutions in higher dimension.**
- **Four LSO algorithms based on DE with the best performances –** MOS, jDElscop, SaDE-MMTS and *m*DE-bES
- **From the special issue of the Soft Computing Journal on Scalability of Evolutionary Algorithms and other Meta-heuristics for Large Scale Continuous Optimization Problems.**

SaDE-MMTS – Two levels of self-adaptation

- SaDE benefits from the self-adaptation of trial vector generation strategies and control parameter adaptation schemes by learning from their previous experiences to suit different characteristic of the problems and different search requirements of evolution phases.

- Every generation, a selection among the JADE mutation strategy with two basic crossover operators (binomial crossover and exponential crossover) as well as no crossover option is also adaptively determined for each DE population member based on the previous search experiences.

S. Z. Zhao, P. N. Suganthan, and S. Das, "Self-adaptive differential evolution with multi-trajectory search for large scale optimization", *Soft Computing*, 15, pp. 2175-2185, 2011.

SaDE-MMTS – Two levels of self-adaptation

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 based on the performance, and this LRF is applied to scale AveDis over the evolution.**

SaDE-MMTS

High Level Self-adaptation between SaDE & MMTS:

- **The MMTS is used periodically for a certain number of function evaluations along with SaDE, which is determined by an adaptive manner.**
- **At the beginning of optimization procedure, the SaDE and the MMTS are firstly conducted sequentially within one search cycle by using the same number of function evaluations. Then the success rates of both SaDE and MMTS are calculated. Subsequently, function evaluations are assigned to SaDE and MMTS in each search cycle proportional to the success rates of both search methods.**

Experiments Results

- **Algorithms were tested on 19 benchmark functions prepared for a Special Issue on Scalability of Evolutionary Algorithms and other Metaheuristics for Large Scale Continuous Optimization Problems. (http://sci2s.ugr.es/eamhco/CFP.php)**
- **The benchmark functions are scalable. The dimensions of functions were 50, 100, 200, 500, and 1000, respectively, and 25 runs of an algorithm were needed for each function.**
- **The optimal solution results, f (x*), were known for all benchmark functions.**

*m*DE-*b*ES

- We boost the population diversity while preserving simplicity by introducing a multi-population DE to solve large-scale global optimization problems
  - the population is divided into independent subgroups, each with different mutation and update strategies
  - A novel mutation strategy that uses information from either the best individual or a randomly selected one is used to produce quality solutions to balance exploration and exploitation.

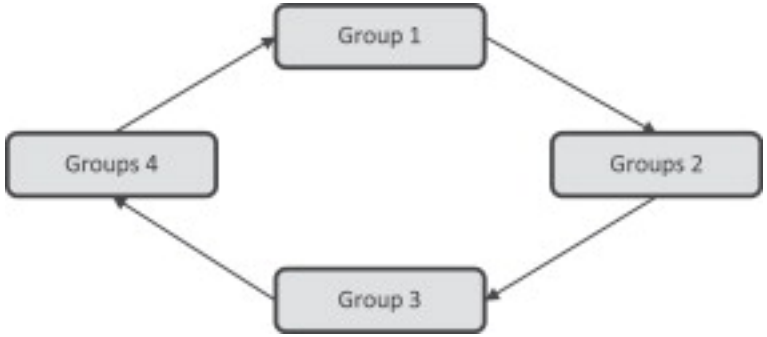
- **Mostafa Z. Ali, Noor H. Awad, Ponnuthurai N. Suganthan "Multi-population differential evolution with balanced ensemble of mutation strategies for large-scale global optimization" Appl. Soft Comput. 33: 304-327 (2015)**

*m*DE-*b*ES

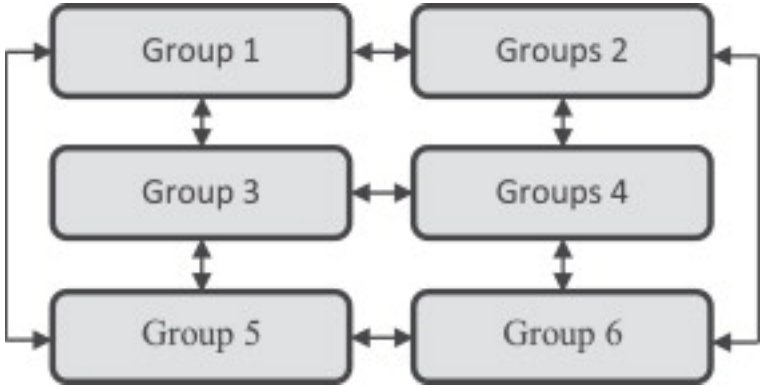
- At the end of each epoch, individuals between the subgroups are exchanged to facilitate information exchange at a slow pace.
- The only link between these subgroups is the sharing of individuals' experiences through the migration of individuals at constant intervals to restore balance of the search direction affording a higher probability of escaping basin of attraction of locally optimal solutions.

Migration phase

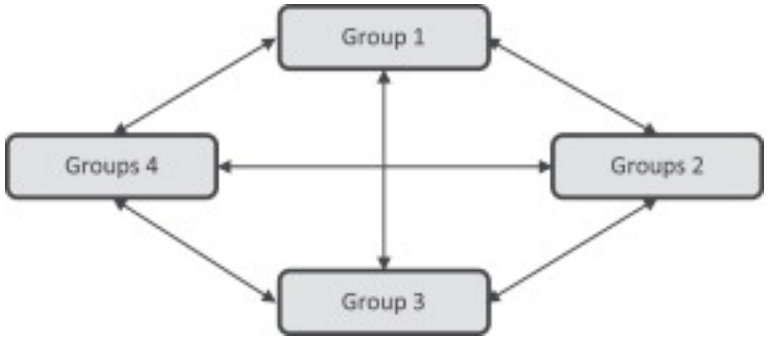
The structure for the migration has several possibilities



Migration with ring topology.



Migration with custom neighborhood topology.



Migration with complete net topology (free form).

Choice of mutation strategies for groups in *m*DE-*b*ES

- DE/rand/1/exp
- DE/current-to-best/1/exp
- DE/best/1/exp
- Modified base vector in mutation strategy
  - This modified mutation uses one of two schemes based on the difference between current best $X_{best}(t)$ and the best from the previous generation $X_{best}(t-1)$.

Experiments Results

- **Algorithms were tested on 19 benchmark functions prepared for a Special Issue on Scalability of Evolutionary Algorithms and other Metaheuristics for Large Scale Continuous Optimization Problems. (http://sci2s.ugr.es/eamhco/CFP.php)**
- **The benchmark functions are scalable. The dimensions of functions were 50, 100, 200, 500, and 1000, respectively, and 25 runs of an algorithm were needed for each function.**
- **The optimal solution results, $f(x^*)$, were known for all benchmark functions.**

Overview

I. Introduction to Real Variable Optimization & DE

II. Future of Real Parameter Optimization

III. Single Objective Optimization

IV. Constrained Optimization

V. Dynamic Optimization

VI. Multi-objective Optimization

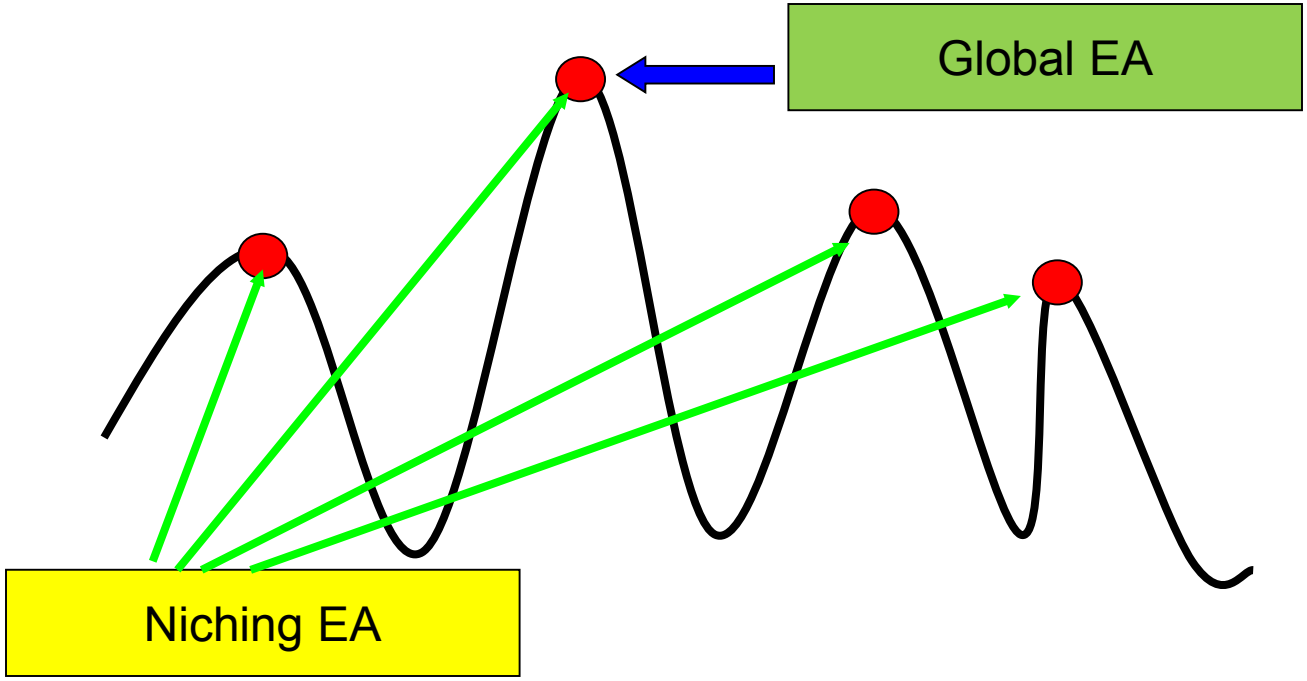
VII. Large Scale Optimization

VIII. Multimodal Optimization

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.

Niching and Multimodal Optimization with DE

- Traditional evolutionary algorithms with elitist selection are suitable to locate a single optimum of **functions**.
- Real problem may require the identification of optima along with several optima.
- For this purpose, **niching methods** extend the simple evolutionary algorithms by ***promoting the formation of subpopulations in the neighborhood of the local optimal solutions.***



Multi-modal Optimization Methods

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

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.

Adaptive 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 IEEE TEC articles published in 2010-2012 period.

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, Oct. 2012.

Classical Speciation Technique

Step 1: Initialize at random a population of size N within the range of [$\mathbf{X}^{Lower}$, $\mathbf{X}^{Upper}$] in D dimensions

Step 2: Compute Euclidean distance for all members

$$\mathbf{dist}_{ij}=\sqrt{\sum(x_{i,j})^2},\ j=1,\ \ldots D,\ i=1,2\ldots N$$

Step 3: Sort all individuals in descending order of fitness

Step 4: Set species number S=1

WHILE sorted population is not empty

1. Identify the fittest member from sorted population and remove it as the species seed for species number S.
2. Mark all members within the speciation radius $r_{species}$ as members of the same species as S
3. Remove the speciated members from the sorted population
4. Check the number of members within species. If the specie size exceeds that of specified, remove the excess members in order of weakest fitness. On the other hand if insufficient members are in the same species, randomly initialize members within $r_{species}$ of the species seed.
5. Increment S until all members classified into species.

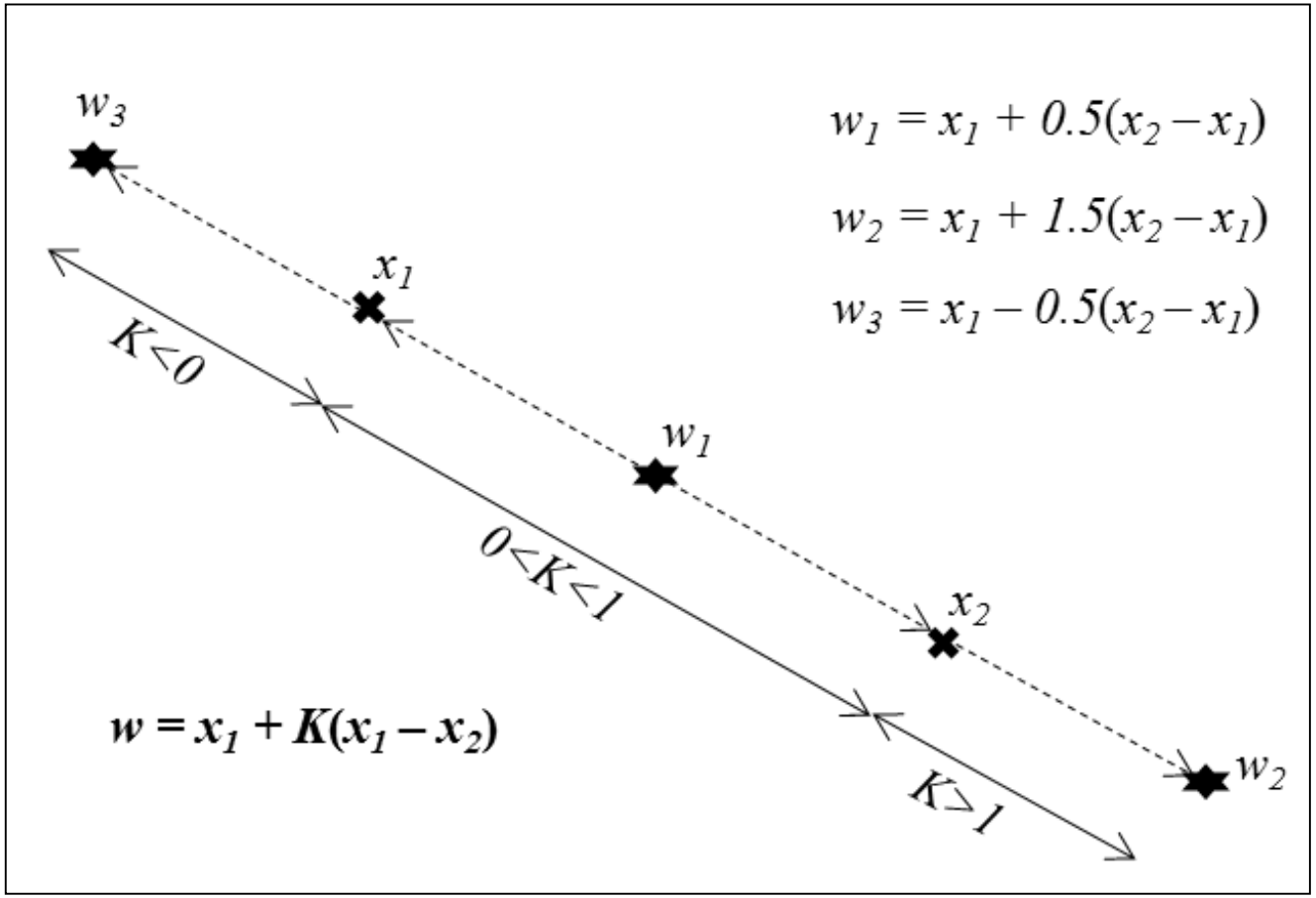
END WHILE

Step 5: Perform normal DE process within each species

Step 6: Repeat Steps 2 to 5 until a termination criterion

- Fixed speciation radius, $R_{species}$
- Removes population members from niches without regard for local fitness environment

Arithmetic Recombination

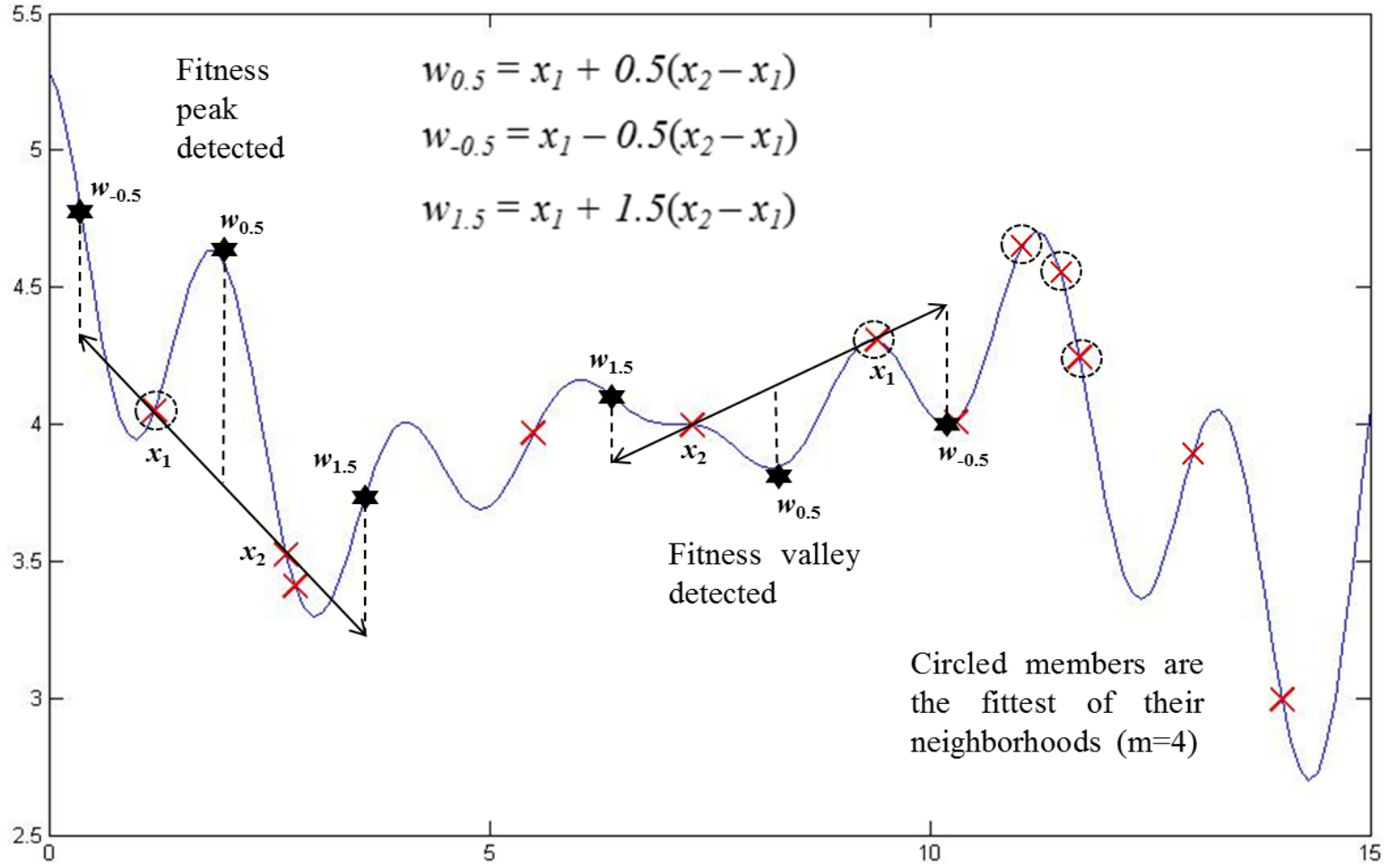


Search region covered by line Arithmetic Recombination for *K= [-0.5, 0.5, 1.5]*

Neighborhood Arithmetic Recombination-based Speciation DE (based on DoI: . 10.1109/TCYB.2015.2394466)

- Classical niching and clustering methods highly sensitive to parameter settings and initial population distribution in addition to niche size
  - e.g Speciation radius $R_{species}$, Fitness-sharing radius $R_{sharing,}$ Crowding factor, CF
- Difficult to separate the initial population into niches in uneven and rugged regions,
  - i.e when niching radius contains more than one peak.
- A guaranteed way to identify separate niches: detecting fitness valleys and peaks
- Arithmetic Recombination interpolates and extrapolates between neighbors from niche centers (local fittest member)
- Self-adaptive generalization across different fitness terrains
- Reduction of multiple niching parameters to only neighborhood popln size, *m*

Interpolating and Extrapolating members using Arithmetic Recombination



X – popln member ; Star: solutions generated Arith. Recomb.

Neighborhood Arithmetic Recombination-based Speciation DE

Step 1: Initialize a randomly distributed population of size N within the range of $[\mathbf{X}^{\text{Lower}}, \mathbf{X}^{\text{Upper}}]$ in D dimensions

Step 2: Compute Euclidean distance for all members

$$\mathbf{dist}_{ij}=\sqrt{\sum(x_{i,j})^2}, j=1, \ldots D, i=1,2\ldots N$$

Step 3: Sort all individuals in descending order of their fitness values. This is the speciation pool.

Step 4: Set niches number S=1

WHILE sorted population is not empty, execute steps 4.1 to 4.6:

1. Identify the fittest member (*lbest*) from speciation pool and remove it as the specie seed for specie number *S*.
2. Extract *m* nearest neighbors to the *lbest* from speciation pool for arithmetic recombination.
3. Detect the presence of fitness valleys and better fitness between *lbest* and the neighbors in the specie using $0<K<1$.
    a) Reject neighbors that are separated by fitness valleys from the *lbest* of the current specie. Rejected neighbors are returned to the speciation pool.
    b) Check for existence of better fitness between *lbest* and neighbors and store these as new members of the same species
4. Explore regions beyond *lbest* and the remaining neighbors using $K>1$ and $K<0$. Solutions with $K>1$ are included into the common speciation pool while $K<0$ are included in the current specie *S*. These solutions will be subjected to steps 4.3 & 4.4 in the next iteration.
5. Check the number of members within specie. If the population size exceeds that of specified, remove the excess members in order of weakest fitness. On the other hand if insufficient members are in the same species, randomly initialize members within Euclidean distance of the furthest specie member (or nearest non-specie neighbor) from the specie seed.
6. Increment *S* until all members are classified into species.

END WHILE

Step 5: Perform Ensemble-DE operations within every specie separately.

Step 6: Repeat Steps 2 to 5 until a termination criterion.

S. Hui, P N Suganthan, "Ensemble and Arithmetic Recombination-Based Speciation Differential Evolution for Multimodal Optimization," IEEE T. Cybernetics, Online. 10.1109/TCYB.2015.2394466

Ensemble Parameters of EARSDE

- Neighborhood size, $m = 6$
- Scaling factor $F_i \in [0.3, 0.5, 0.9]$.
- Crossover probability $C_i \in [0.1, 0.5]$.
- Binomial Crossover

$$U_{i,j} = \begin{cases} v_{i,j} \; if \; (rand_j[0,1) \le Cr) \; or \; (j = j_{rand}) \\ x_{i,j} \qquad\qquad otherwise \end{cases}$$

- Mutation Strategies
  1. DE/rand/1

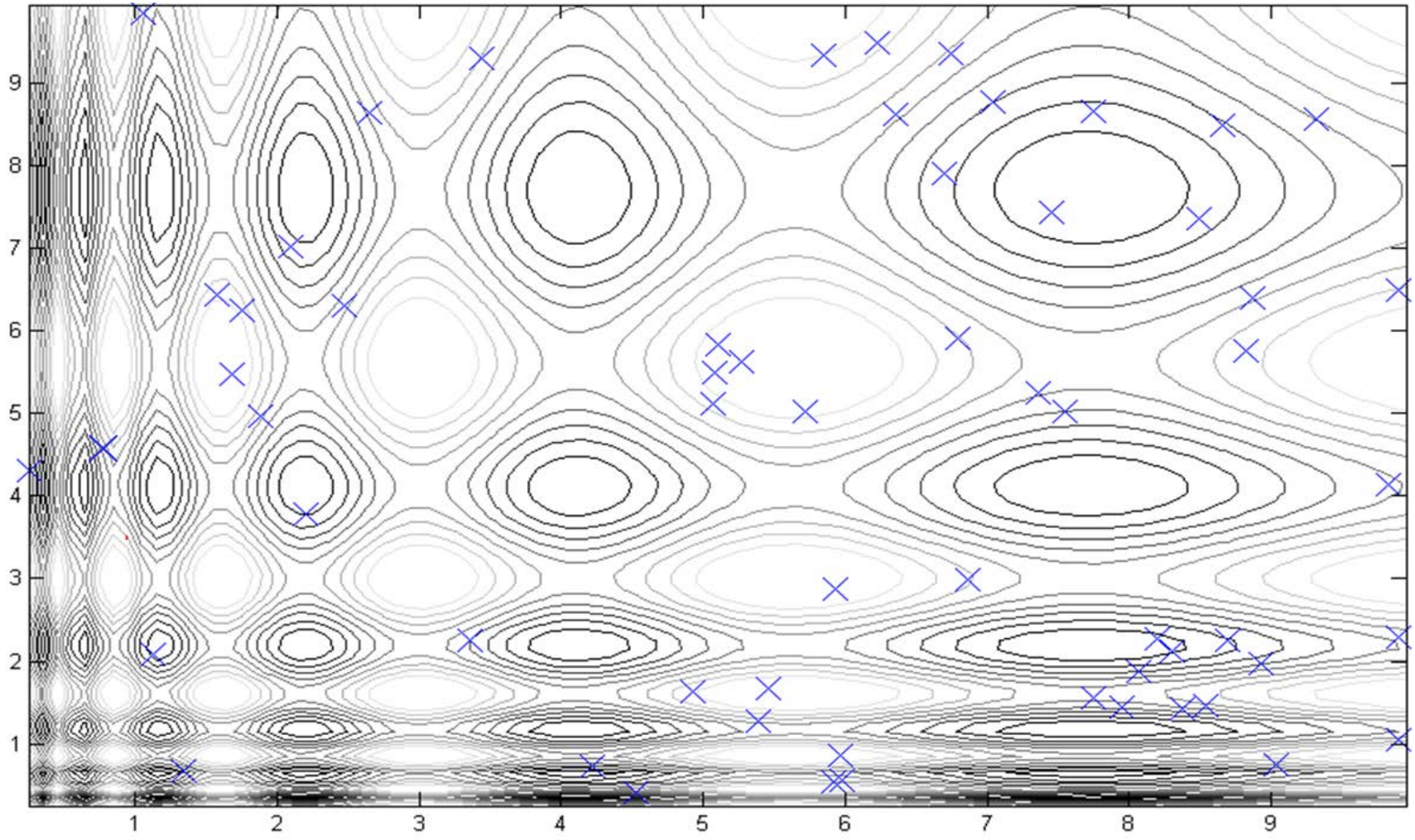
     $$\boldsymbol{v_i^G} = \boldsymbol{x}_{rand1,i}{}^G + F(\boldsymbol{x}_{rand2,i}{}^G - \boldsymbol{x}_{rand3,i}{}^G)$$

  2. DE/ best/1

     $$\boldsymbol{v}_i{}^G = \boldsymbol{x}_{best}{}^G + F(\boldsymbol{x}_{rand1,i}{}^G - \boldsymbol{x}_{rand2,i}{}^G)$$

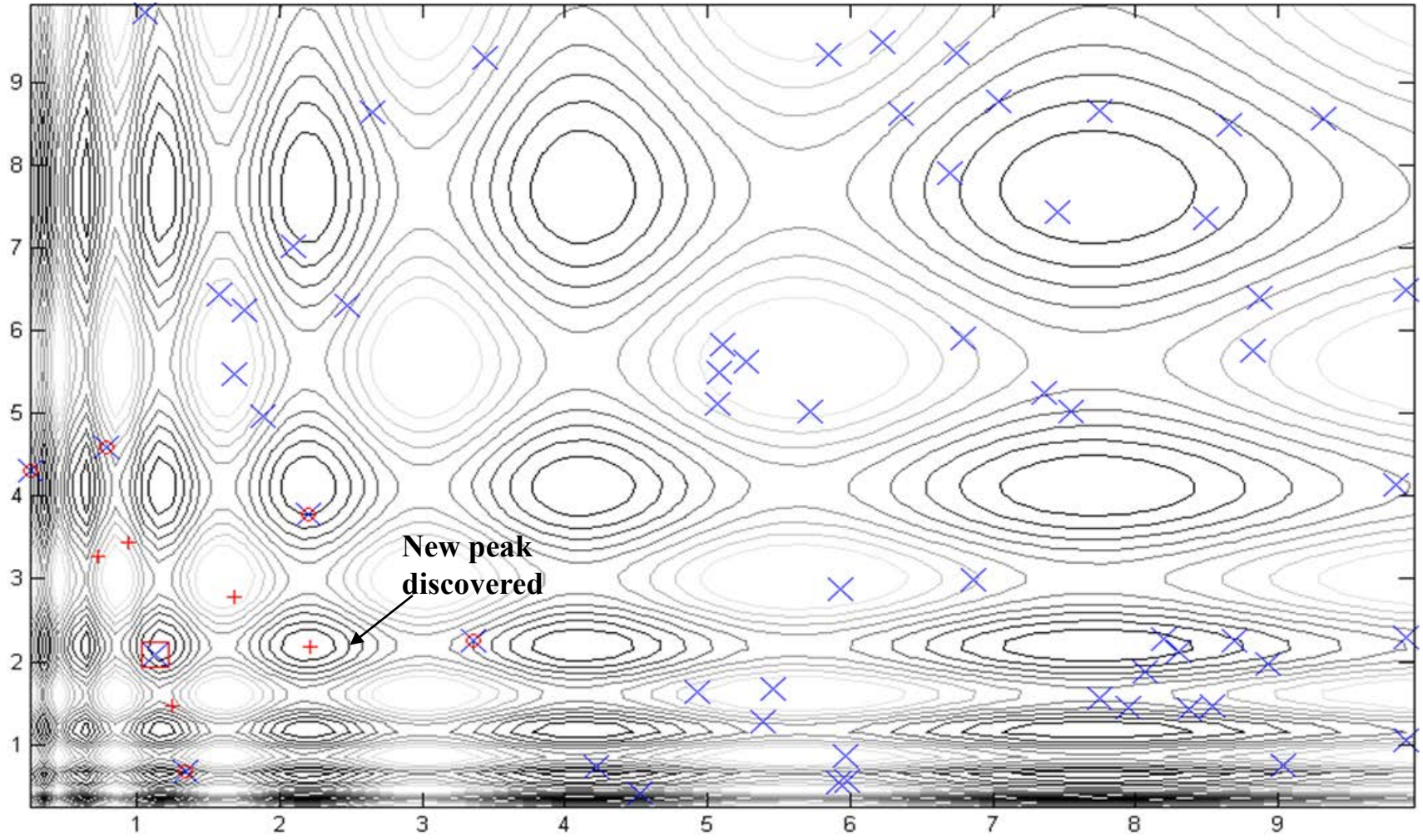
- Arithmetic Recombination scaling factor, $K = [-0.5, 0.5, 1.5]$ with additive variable, $\pm\Delta$
- $\Delta \in [0, 0.1]$

Speciation process in EARSDE in 2D Vincent problem



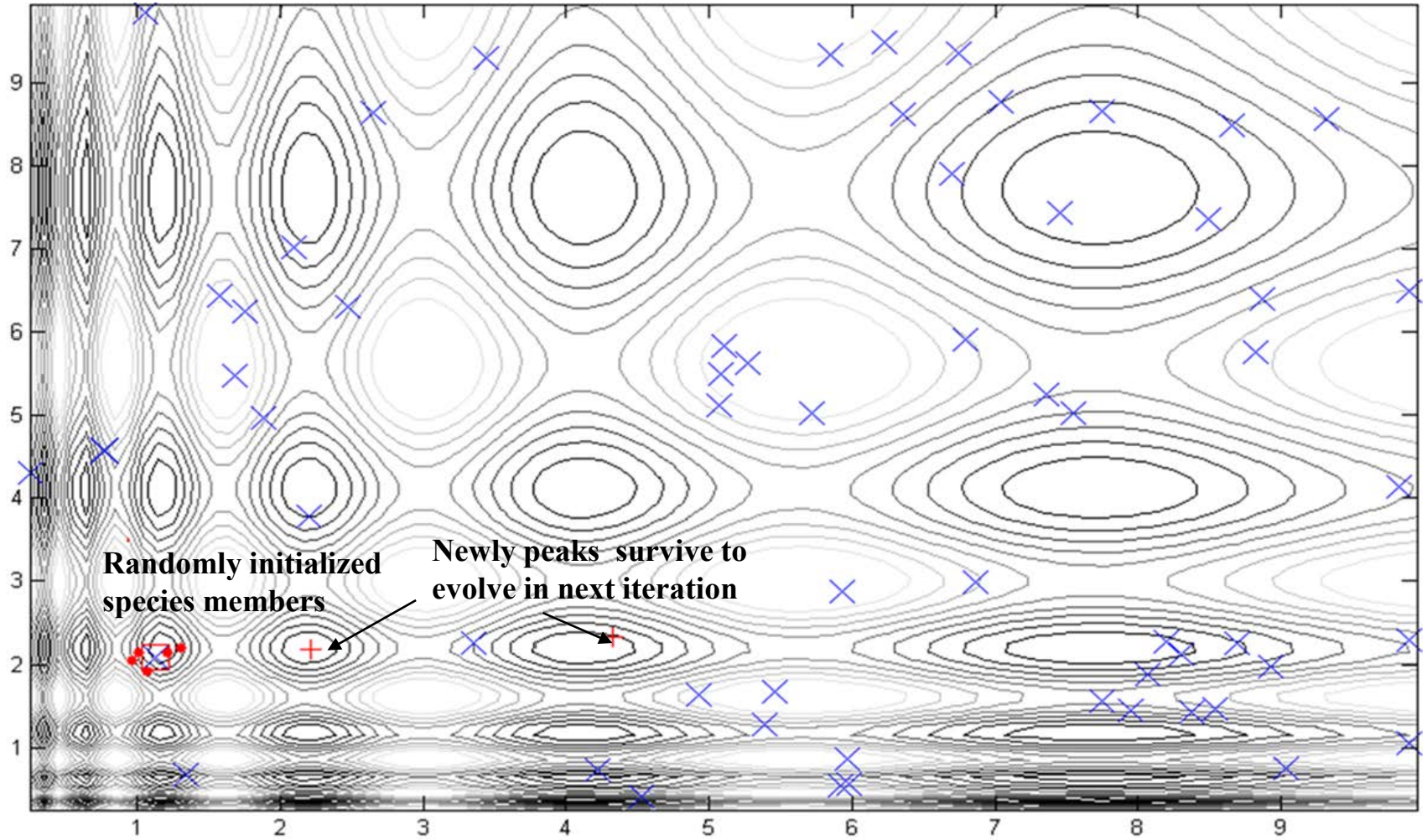
- Initial population - 'X', initial pop-size = 60, neighborhood size = 6
- Fittest regions are represented by the darkest contours.
- Fittest member is first selected as the species seed for AR operations

Speciation process in EARSDE in 2D Vincent problem



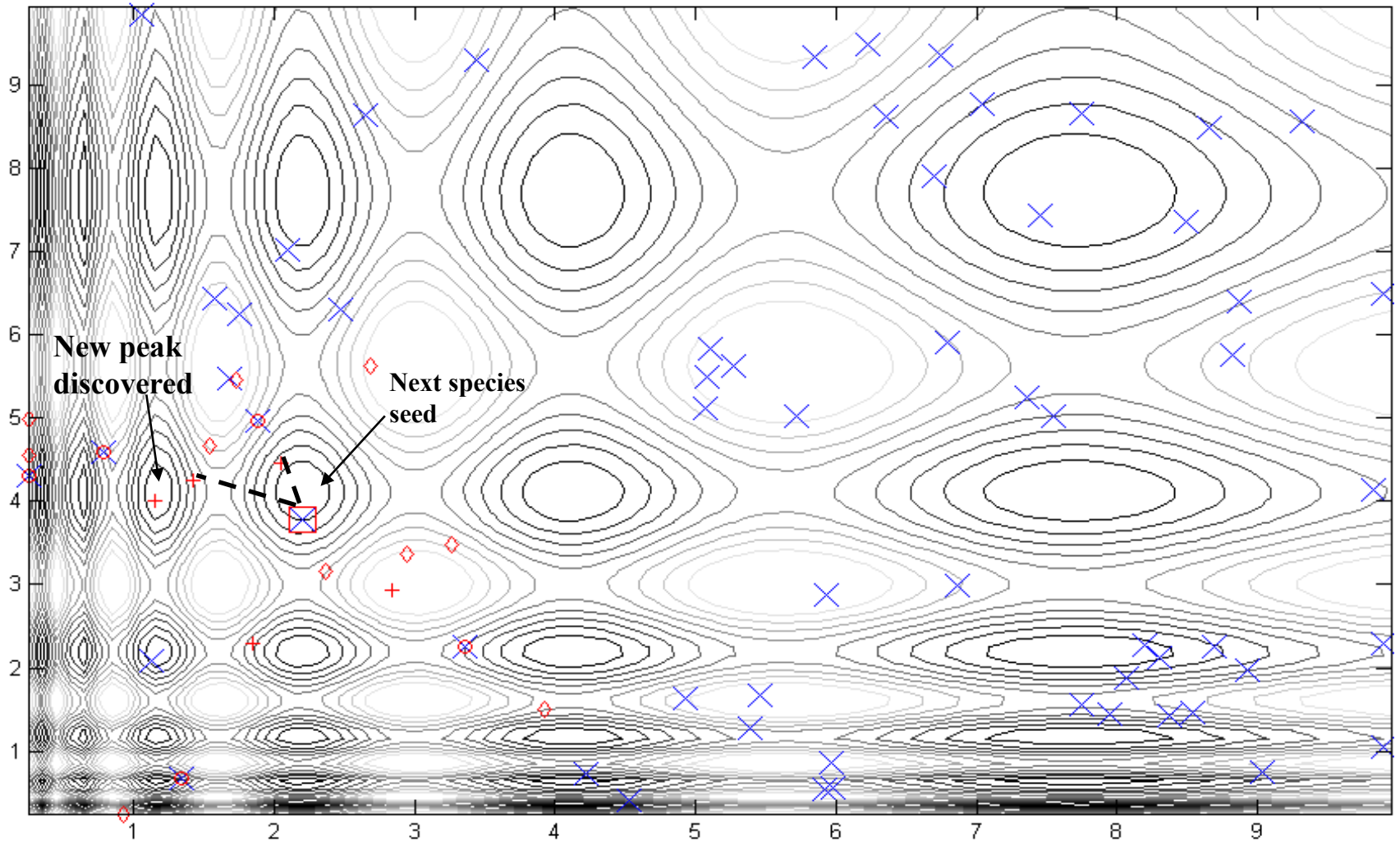
- species seed is highlighted with a red square '□'
- 5 nearest neighbors are highlighted with red circle 'o'
- *AR* (*K*=0.5) applied to check for any fitness valleys.
- Midpoints represented with red plus signs '+'

Speciation process in EARSDE in 2D Vincent problem



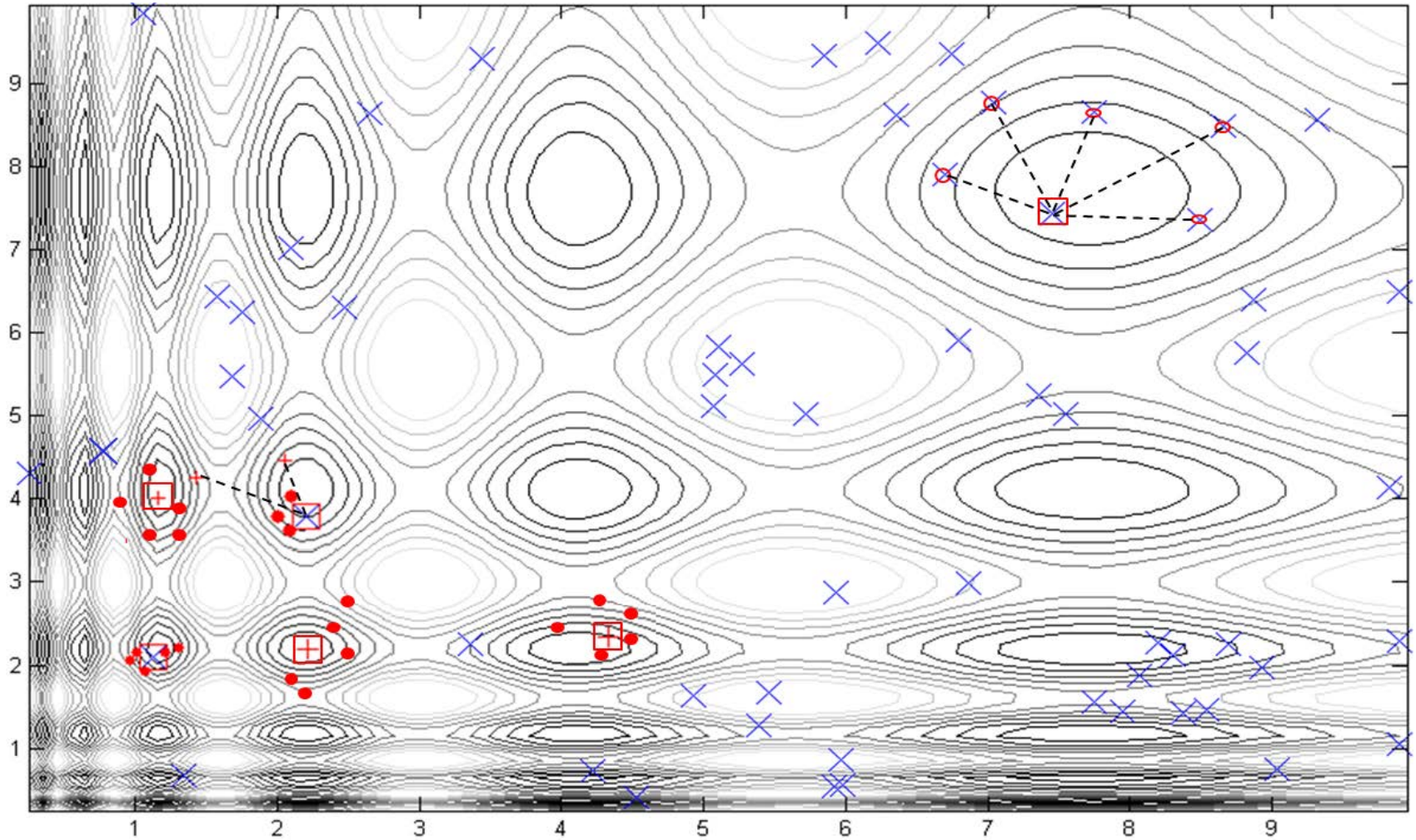
- Existing neighbors of the species seed could not be grouped into the same species due to fitness valleys or doubled peaks
- Random initialization executed around original species seed within 0.5 of the distance to the nearest neighbor separated by fitness valley.
- New random members are represented by the red circles '●'

Speciation process in EARSDE in 2D Vincent problem



- A neighbor rejected by the previous species now selected for speciation. Repeat process
- Dotted black lines to indicate association to species

Speciation process in EARSDE in 2D Vincent problem



- New peaks (highlighted by a red square '□' with red plus signs '+') would only enter speciation process in the next iteration
- Random members populated around new peaks.

THANK YOU

Q & A