Numerical Optimization
by DE and PSO

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

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

IEEE SSCI 2018
Bengaluru, India
20th November 2018

Overview

I. Some General Thoughts

II. Introduction to DE and PSO

III. Future Directions

IV. Further PSO Variants

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:

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

(limited to our own publications & CEC Competitions)

IEEE CEC 2019 in New Zealand in June

GECCO 2019 in Czech Republic in July

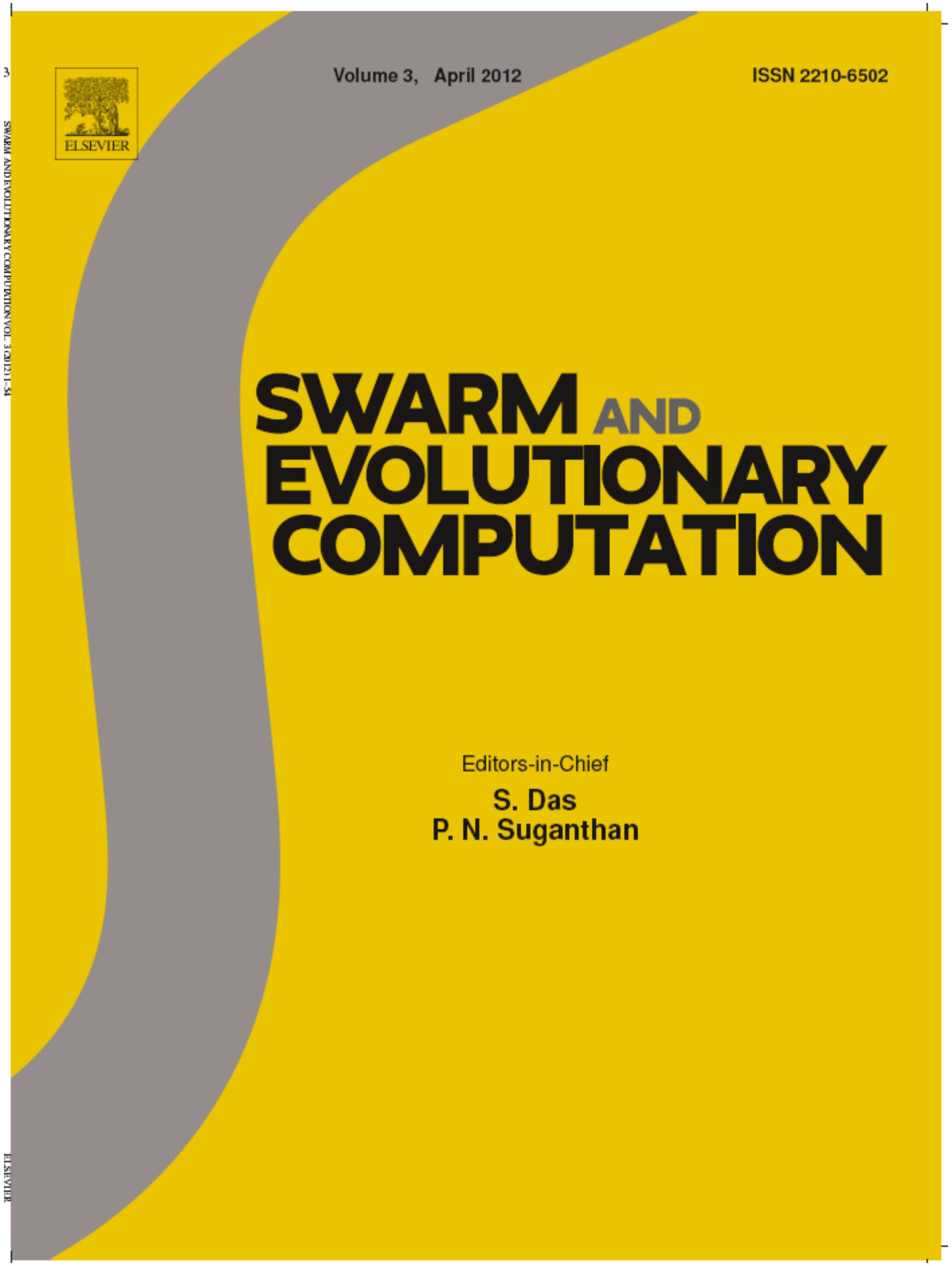
Randomization-Based ANN, Pseudo-Inverse Based Solutions, Kernel Ridge Regression, Random Forest and Related Topics

http://www.ntu.edu.sg/home/epnsugan/index_files/RNN-Moore-Penrose.htm
http://www.ntu.edu.sg/home/epnsugan/index_files/publications.htm

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

SCI Indexed from Vol. 1, Issue 1.

2018 IF:
2 years = 3.8



ELSEVIER

Volume 3, April 2012

ISSN 2210-6502

SWARM AND EVOLUTIONARY COMPUTATION

Editors-in-Chief
S. Das
P. N. Suganthan

Overview

I. Some General Thoughts

II. Introduction to DE and PSO

III. Future Directions

IV. Further PSO / DE Variants

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, i.e. if your new algorithm is not the best in each problem, you can use NFL to make reviewers accept your paper ☺

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) corresponding to the (local/global) optimum.
  - Single point based search methods: convergence is a must.
- 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, i.e. the same operators / parameters are good for diverse problems ? (NFL?)
- What are Reviewers' preferences? (one algo good for all problems!)
- **Or good to have several parameters that can be adaptively tuned on the fly to achieve good performance on diverse problems? YES**
- **If NFL says that a single algorithm is not the best for each problem in a diverse set of problems, then good to have many algorithmic parameters & operators to adapt for problems !!**

CEC 2015 Competitions: "Learning-Based Optimization"

Similar Literature: Thomas Stützle, Holger Hoos, …

Metaheuristics

A few points from: https://en.wikipedia.org/wiki/Metaheuristic

I. **Heuristic** (from Greek εὑρίσκω "I find, discover") is a technique designed for finding an approximate more quickly when classic methods are too slow, or fail to find any exact solution. This is achieved by trading optimality, completeness, accuracy, or precision for speed.

II. Meta – beyond, for example inspiration from nature.

III. Metaheuristics may make few assumptions about the optimization problem being solved, and so they may be usable for a variety of problems.

IV. Many metaheuristic methods have been published with claims of novelty and practical efficacy. While the field also features high-quality research, unfortunately many of the publications have been of poor quality; flaws include vagueness, lack of conceptual elaboration, poor experiments, and ignorance of previous literature.

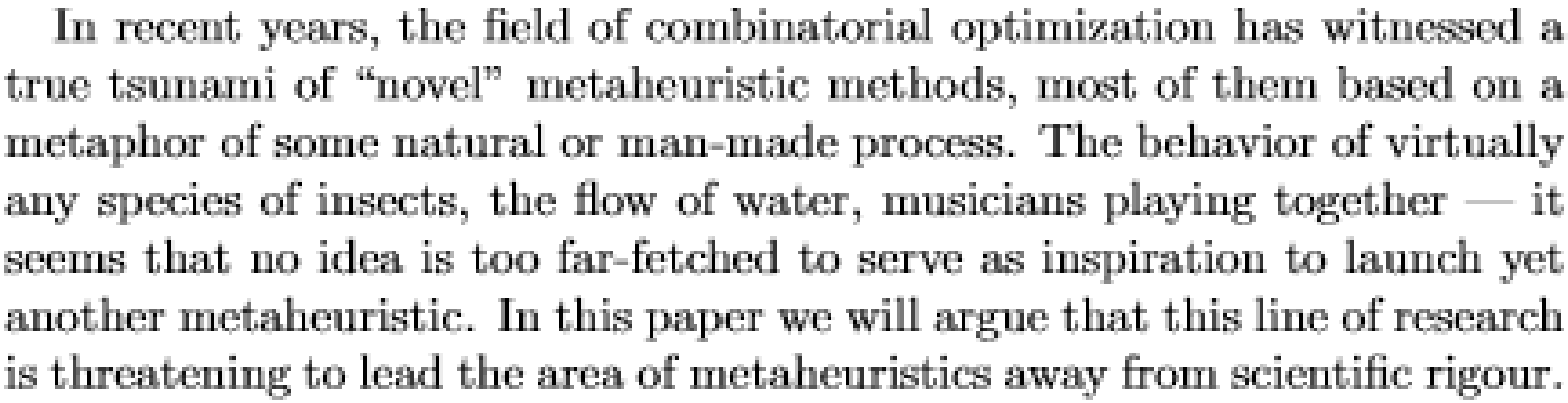
V. If you wish to be known as a "father of an invention", nature-inspired metaheuristics is a good field … ☺

Metaheuristics

I. When to use: Only when traditional methods with proofs, etc. can’t offer a satisfactory solution within the available time.
   - As these two research fields have diverged, cross-checking is not done frequently, i.e. recent development of traditional methods are ignored …

II. Proofs for Metaheuristics: A little contradictory as metaheuristics are developed to complement mathematical programming methods with proofs, etc.
   - Metaheuristics claims to make no assumption about the problem being solved while mathematical theories developed for these metaheuristics have several assumptions, etc.

III. From “Metaheuristics – the Metaphor Exposed”, by Kenneth Sorensen: https://web.archive.org/web/20131102075645/http://antor.ua.ac.be/system/files/mme.pdf



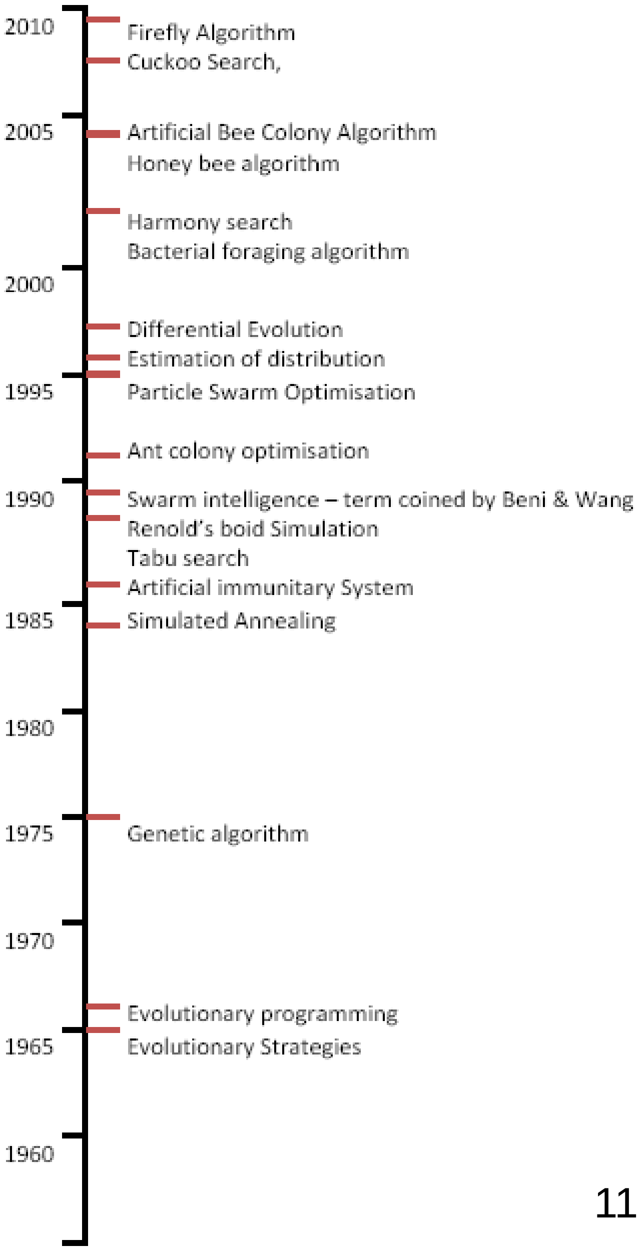
In recent years, the field of combinatorial optimization has witnessed a true tsunami of “novel” metaheuristic methods, most of them based on a metaphor of some natural or man-made process. The behavior of virtually any species of insects, the flow of water, musicians playing together — it seems that no idea is too far-fetched to serve as inspiration to launch yet another metaheuristic. In this paper we will argue that this line of research is threatening to lead the area of metaheuristics away from scientific rigour.

Some of the Metaheuristics

https://en.wikipedia.org/wiki/List_of_metaphor-based_metaheuristics

Evolution strategy (1960s), Genetic algorithms (1970

- 1.1 Simulated annealing (Kirkpatrick et al. 1983)
- 1.2 Ant colony optimization (Dorigo, 1992)
- 1.3 Particle swarm optimization (Kennedy & Eberhart 1995)
- 1.4 Harmony search (Geem, Kim & Loganathan 2001)
- 1.5 Artificial bee colony algorithm (Karaboga 200
- 1.6 Bees algorithm (Pham 2005)
- 1.7 Glowworm swarm optimization (Krishnanand Ghose 2005)
- 1.8 Shuffled frog leaping algorithm (Eusuff, Lans & Pasha 2006)
- 1.9 Imperialist competitive algorithm (Atashpaz-Gargari & Lucas 2007)



Some of the Metaheuristics

https://en.wikipedia.org/wiki/List_of_metaphor-based_metaheuristics

- 1.10 River formation dynamics (Rabanal, Rodríguez & Rubio 2007)
- 1.11 Intelligent water drops algorithm (Shah-Hosseini 2007)
- 1.12 Gravitational search algorithm (Rashedi, Nezamabadi-pour & Saryazdi 2009)
- 1.13 Cuckoo search (Yang & Deb 2009)
- 1.14 Bat algorithm (Yang 2010)
- 1.15 Spiral optimization (SPO) algorithm (Tamura & Yasuda 2011,2016-2017)
- 1.16 Flower pollination algorithm (Yang 2012)
- 1.17 Cuttlefish optimization algorithm (Eesa, Mohsin, Brifcani & Orman 2013)
- 1.18 Artificial swarm intelligence (Rosenberg 2014)
- 1.19 Duelist Algorithm (Biyanto 2016)
- 1.20 Killer Whale Algorithm (Biyanto 2016)
- 1.21 Rain Water Algorithm (Biyanto 2017)
- 1.22 Mass and Energy Balances Algorithm (Biyanto 2017)
- 1.23 Hydrological Cycle Algorithm (Wedyan et al. 2017)

Never possible to have a complete list of metaheuristics …

More on Metaheuristics

- Dr Xin She Yang’s Scholar page: https://scholar.google.co.uk/citations?user=fA6aTlAAAAAJ

- Dr Seyedali Mirjalili’s scholar page:

https://scholar.google.com/citations?user=TJHmrREAAAAJ&hl=en

- Cultural Algorithms, Brainstorming, Fireworks, Spider Monkey, Egyptian Vulture, Wolf, Grey Wolf, …

Overview

I. Some General Thoughts

II. Introduction to DE and PSO

III. Future Directions

IV. Further PSO / DE Variants

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.**

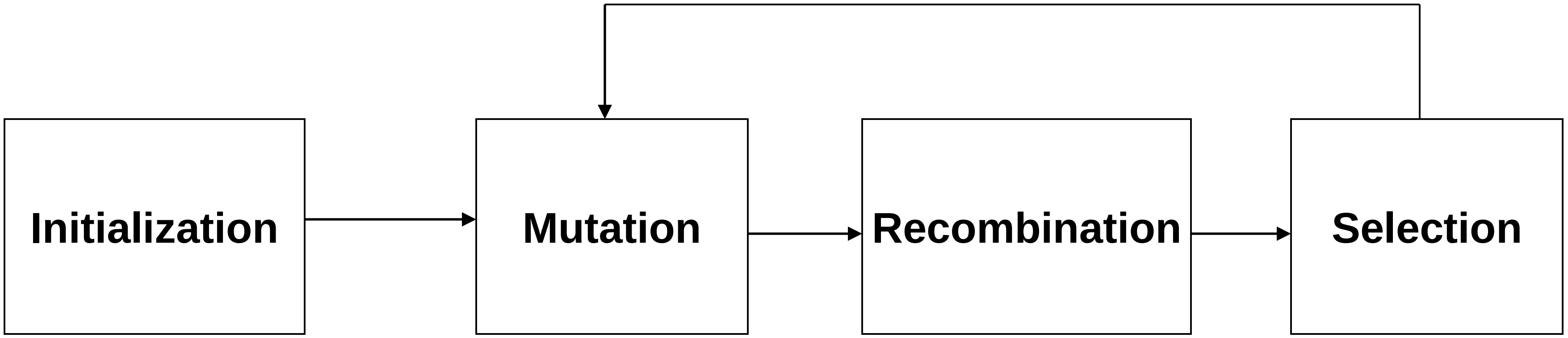
- **Very easy to implement in any standard programming language.**

- **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.**

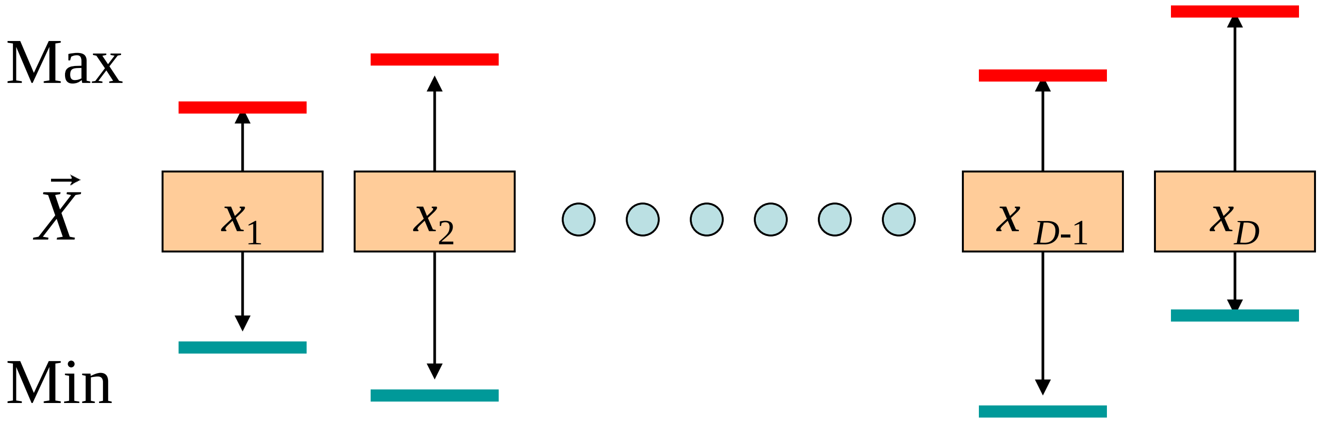
- DE is an Evolutionary Algorithm

- This Class also includes GA, Evolutionary Programming and Evolutionary Strategies



Basic steps of an Evolutionary Algorithm

Representation

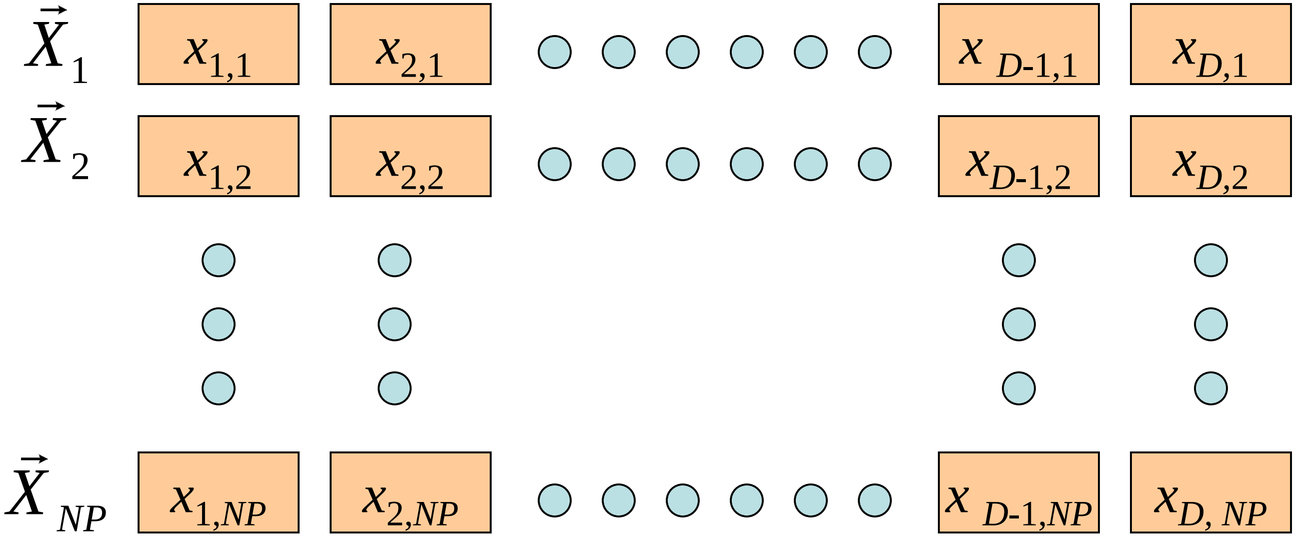


Solutions are represented as vectors of size *D* with each value taken from some domain.

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

Maintaining Population of Size *NP*

We will maintain a population of size *NP*



$$\begin{array}{cccccc}
\vec{X}_1 & x_{1,1} & x_{2,1} & \cdots & x_{D-1,1} & x_{D,1} \\
\vec{X}_2 & x_{1,2} & x_{2,2} & \cdots & x_{D-1,2} & x_{D,2} \\
 & \vdots & \vdots & & \vdots & \vdots \\
\vec{X}_{NP} & x_{1,NP} & x_{2,NP} & \cdots & x_{D-1,NP} & x_{D,NP}
\end{array}$$

The population size *NP*

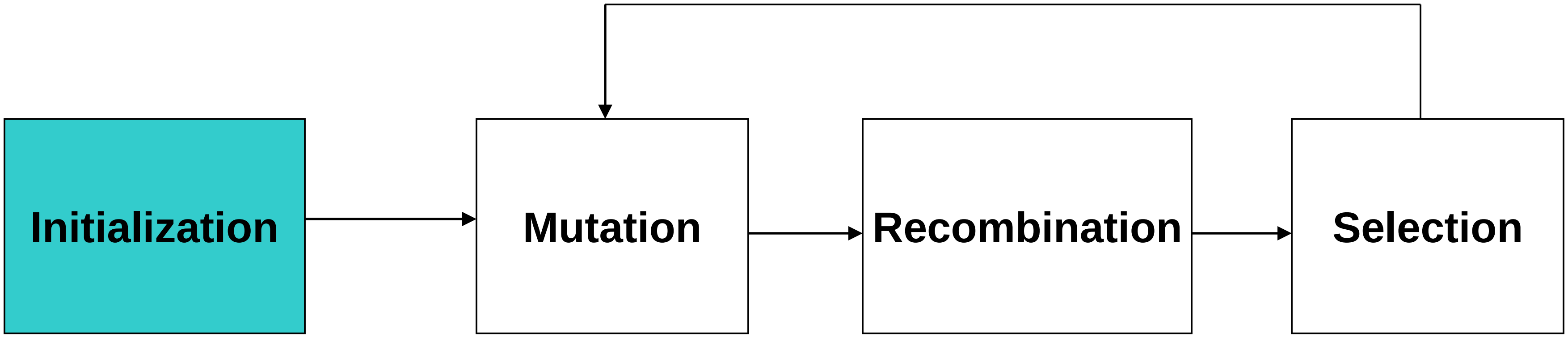
1) The influence of *NP* on the performance of DE is yet to be extensively studied and fully understood.

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.

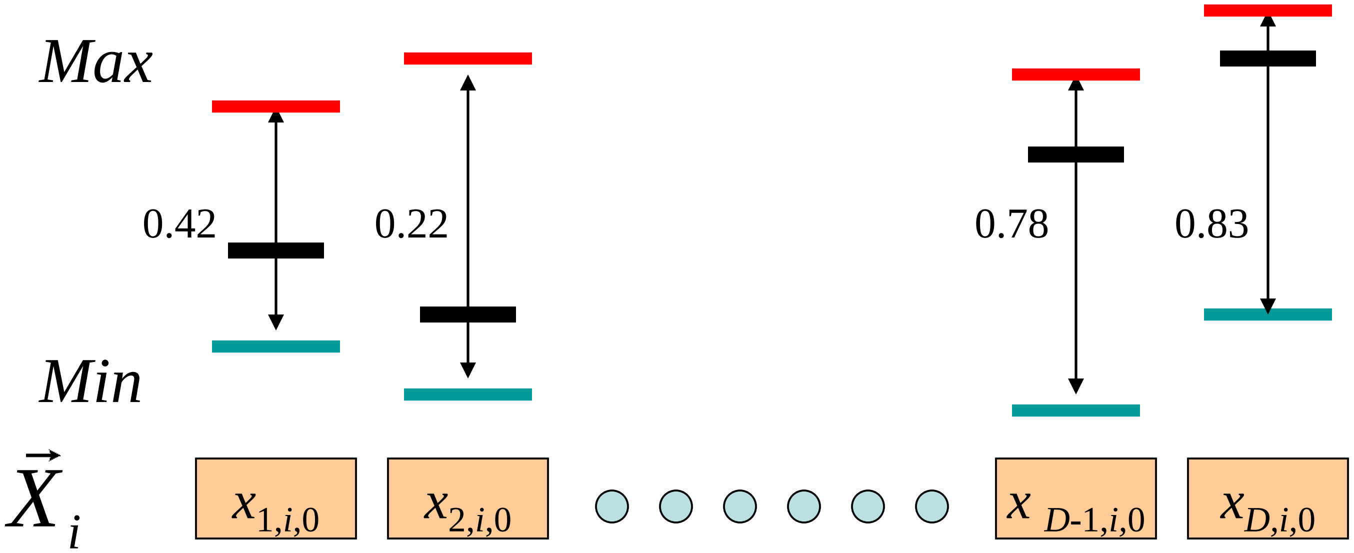


Initialization

Mutation

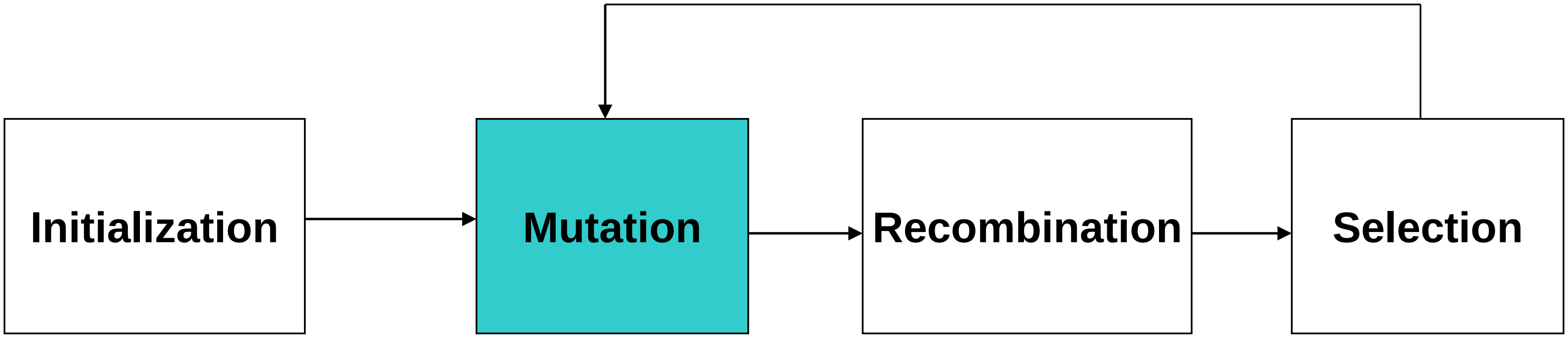
Recombination

Selection



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

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



- For each vector select three other parameter vectors randomly.

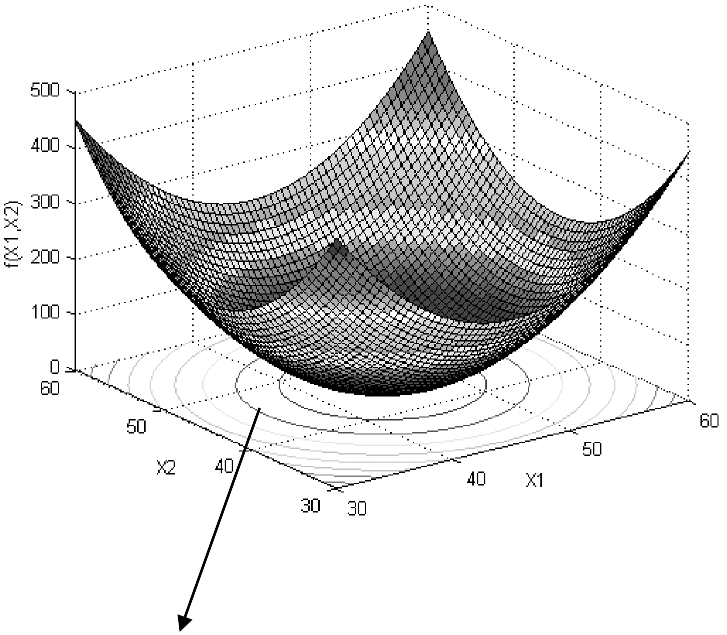
- Add the weighted difference of two of the parameter vectors to the third to form a donor vector (most commonly seen form of DE-mutation):

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

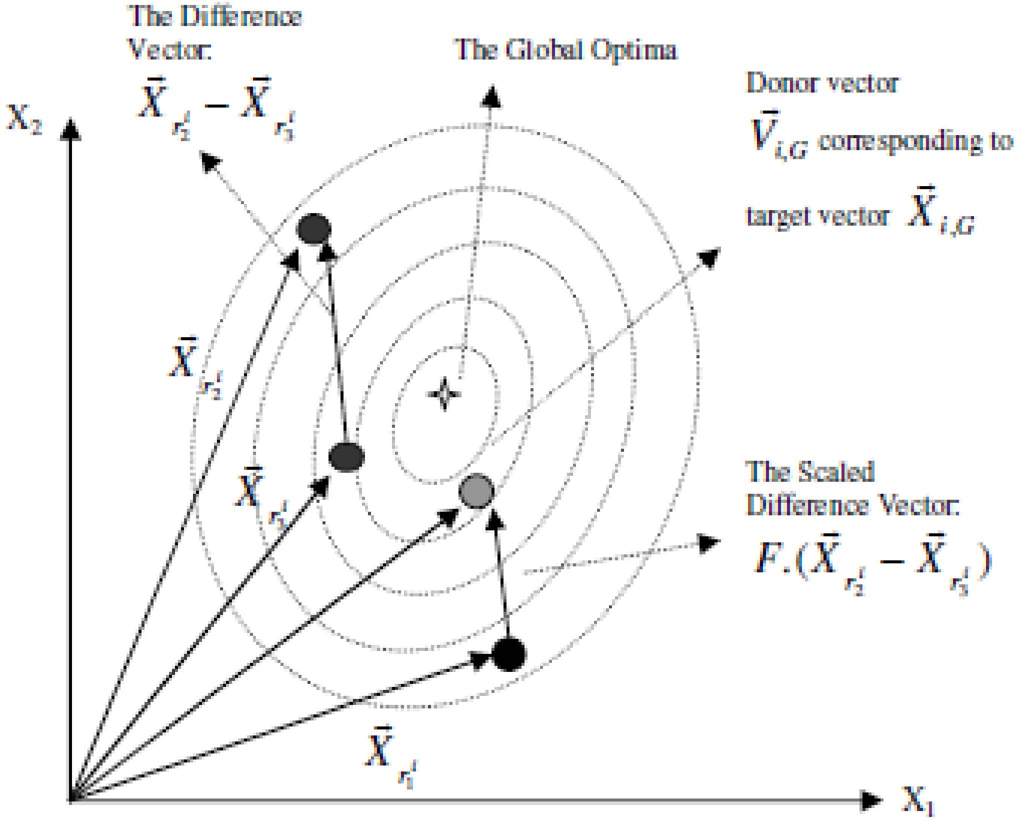
- The scaling factor ***F*** is a constant from (0, 2)

- Self-referential Mutation

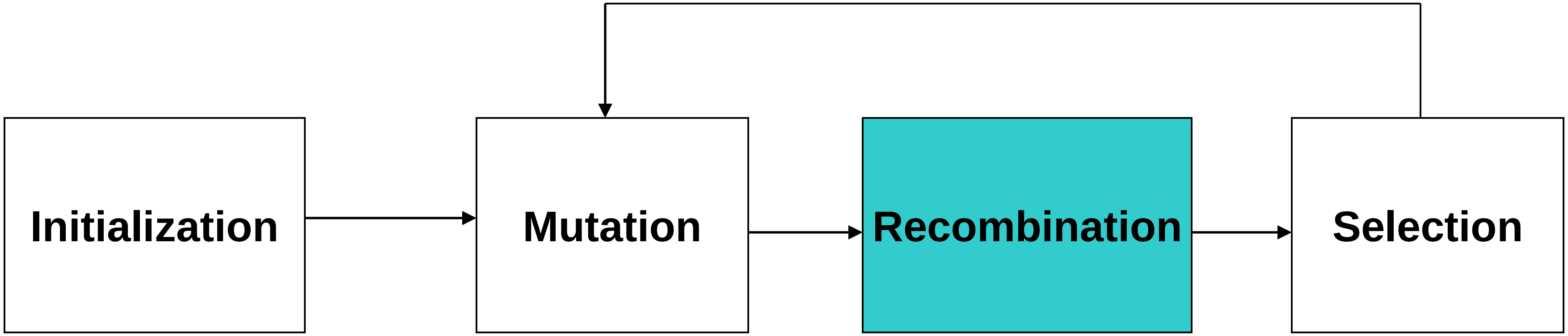
Example of formation of donor vector over two-dimensional constant cost contours



Constant cost contours of Sphere function



The Difference Vector: $\vec{X}_{r_2^i} - \vec{X}_{r_3^i}$
The Global Optima
Donor vector $\vec{V}_{i,G}$ corresponding to target vector $\vec{X}_{i,G}$
X_2
$\vec{X}_{r_2^i}$
$\vec{X}_{r_3^i}$
The Scaled Difference Vector: $F.(\vec{X}_{r_2^i} - \vec{X}_{r_3^i})$
$\vec{X}_{r_1^i}$
X_1



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*.

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

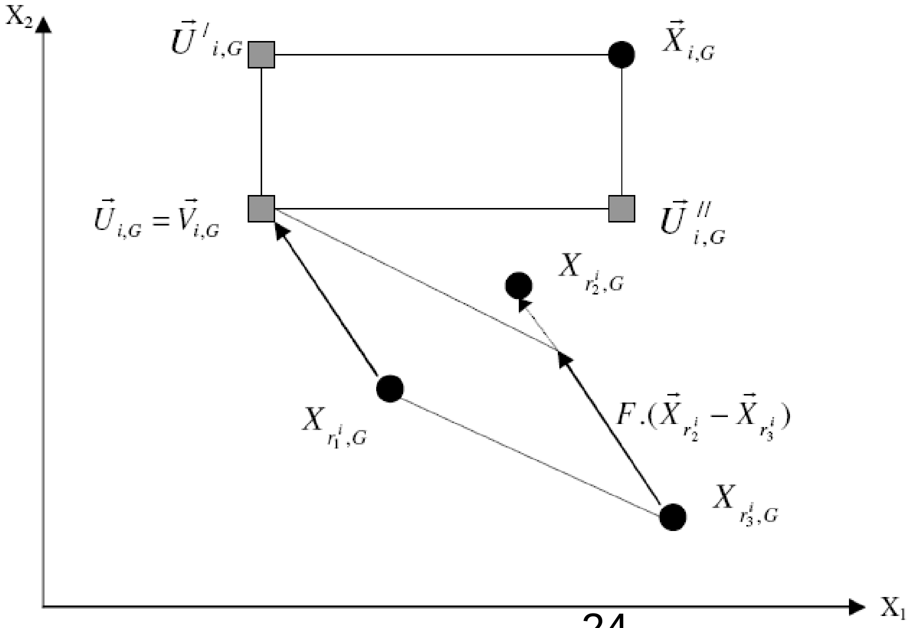
An Illustration of Binomial Crossover in 2-D Parametric Space:

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:*

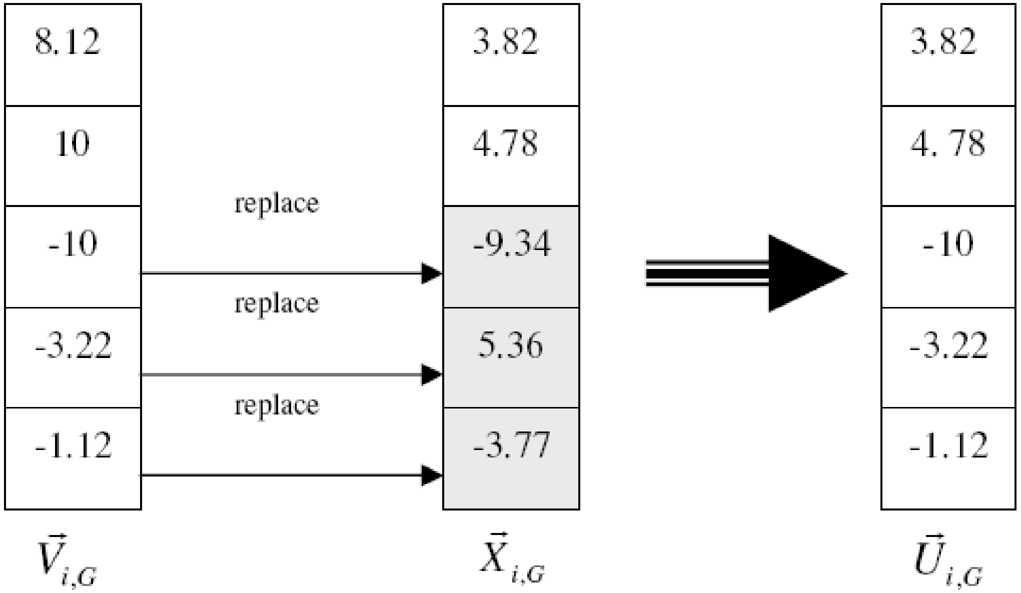
```
$L = 0;$
DO
{
    $L = L+1;$
} WHILE $(((rand[0,1) \leq Cr)$ AND $(L < D))$ ;
```

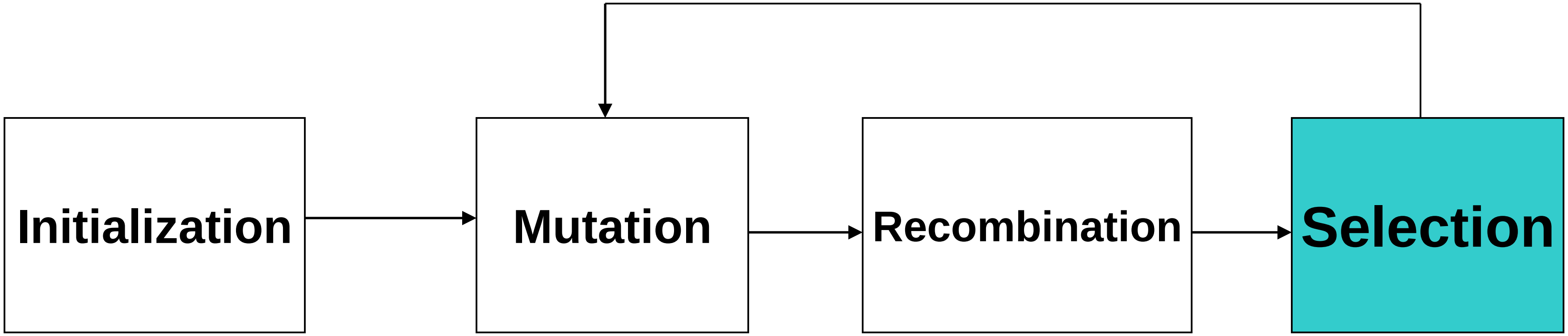
Exploits linkages among neighboring decision variables. If benchmarks have this feature, it performs well. Similarly, for real-world problems with neighboring linkages.

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.

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

- 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) = x²+y² $f(x,y) = x^2+y^2$ ***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) = $2^2 + (-1)^{2} = 5$

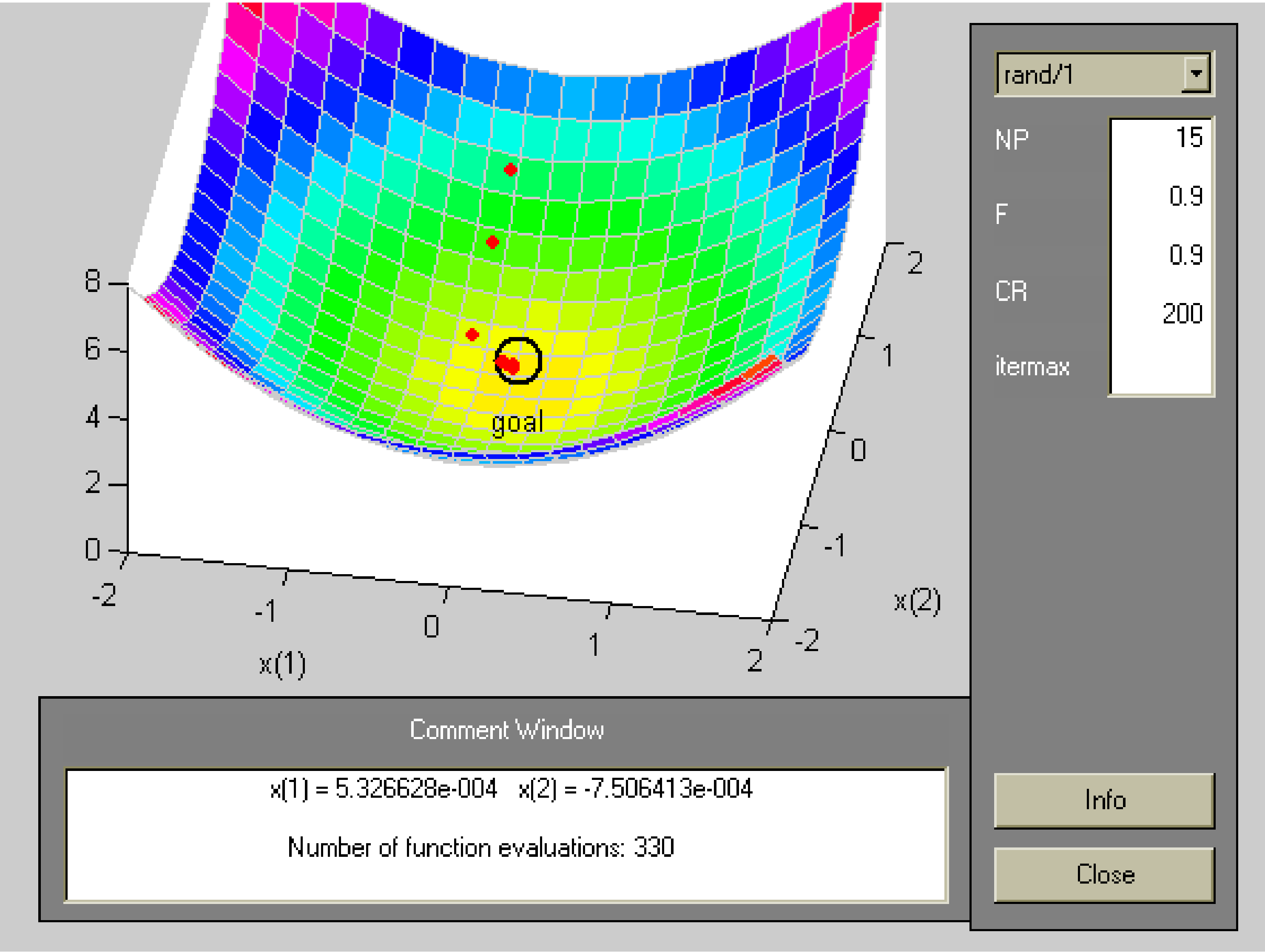
Fitness of offspring

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



rand/1

NP 15

F 0.9

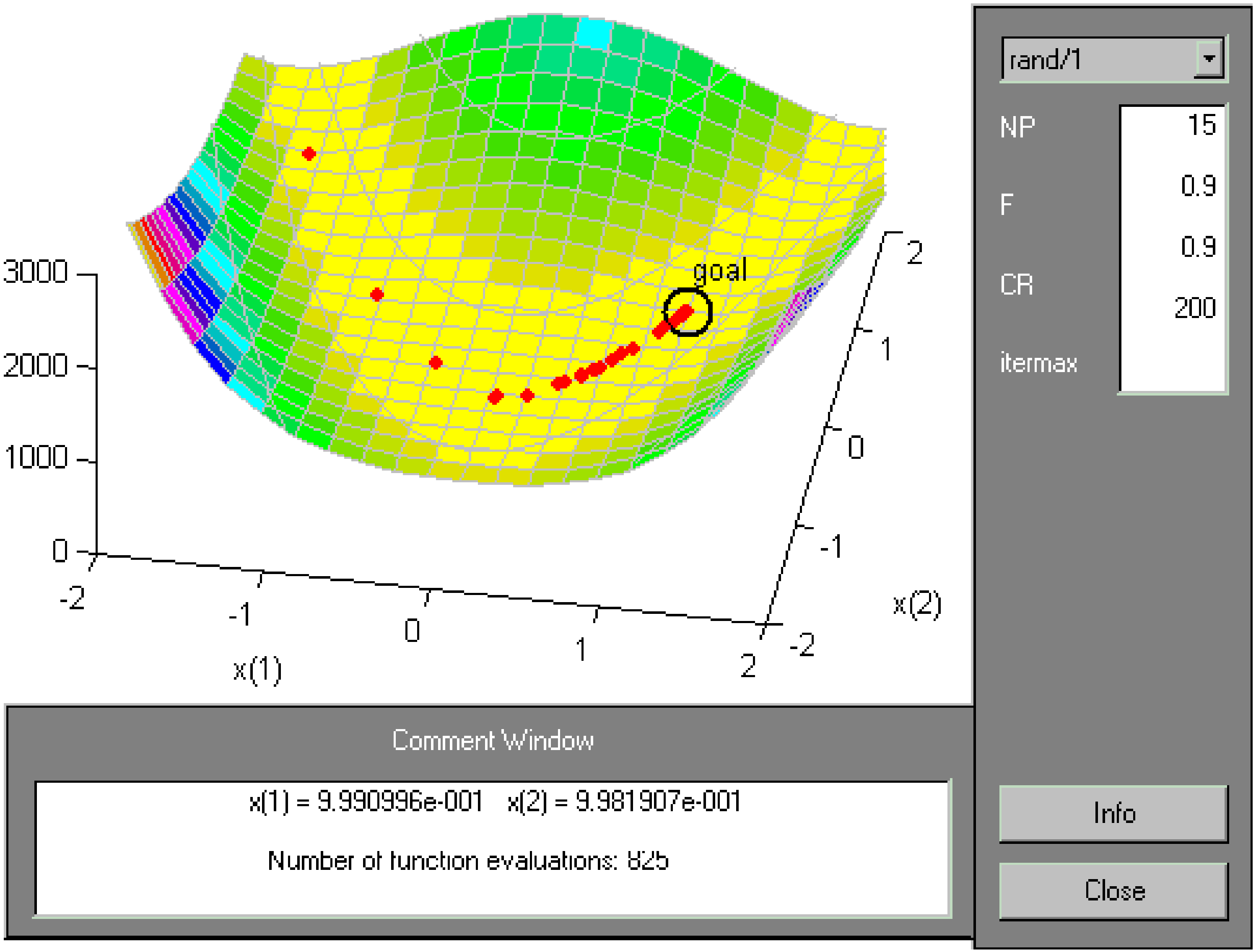
CR 0.9

itermax 200

```
x(1) = 5.326628e-004   x(2) = -7.506413e-004

Number of function evaluations: 330
```

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



rand/1

NP 15

F 0.9

CR 0.9

itermax 200

x(1) = 9.990996e-001   x(2) = 9.981907e-001

Number of function evaluations: 825

Additional 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”: $\vec{V}_i(t) = \vec{X}_{r_1^i}(t) + F_1.(\vec{X}_{r_2^i}(t) - \vec{X}_{r_3^i}(t)) + F_2.(\vec{X}_{r_4^i}(t) - \vec{X}_{r_5^i}(t)).$

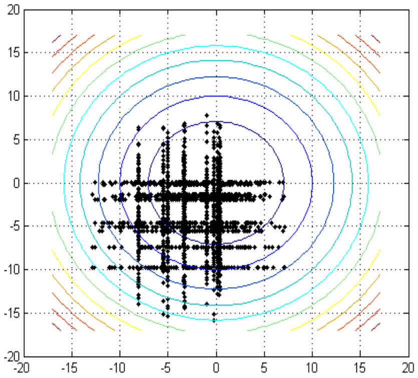
The general convention used for naming the various mutation strategies is DE/x/y/z, where DE stands for Differential Evolution, x represents a string denoting the vector to be perturbed, y is the number of difference vectors considered for perturbation of x, and z stands for the type of crossover being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

The Crossover Rate *Cr*

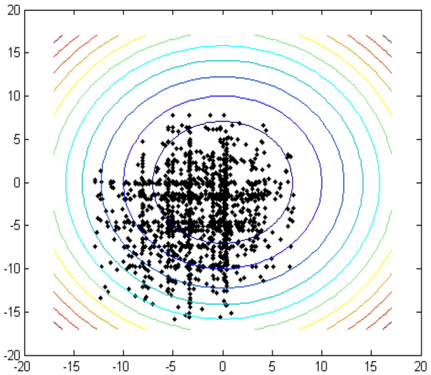
1) The parameter *Cr* controls how many parameters in expectation, are changed in a population member.
2) Low value of *Cr*, a small number of parameters are changed in each generation and the stepwise movement tends to be orthogonal to the current coordinate axes → Good for separable problems

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

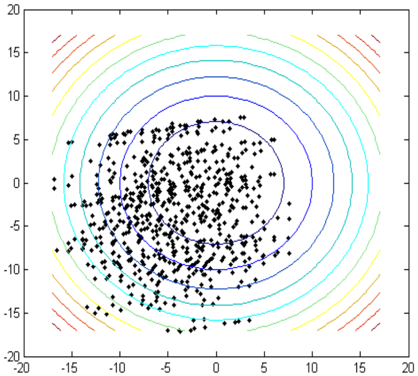
Empirical distribution of trial vectors for three different values of *Cr* has been shown. The plots were obtained by obtained by running DE on a single starting population of 10 vectors for 200 generations with selection disabled.



(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.

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

Particle Swarm Optimizer

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

Particle Swarm Optimizer

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

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

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

i – particle counter & *d* – dimension counter

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

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

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

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

Particle Swarm Optimizer

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

PSO variants

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

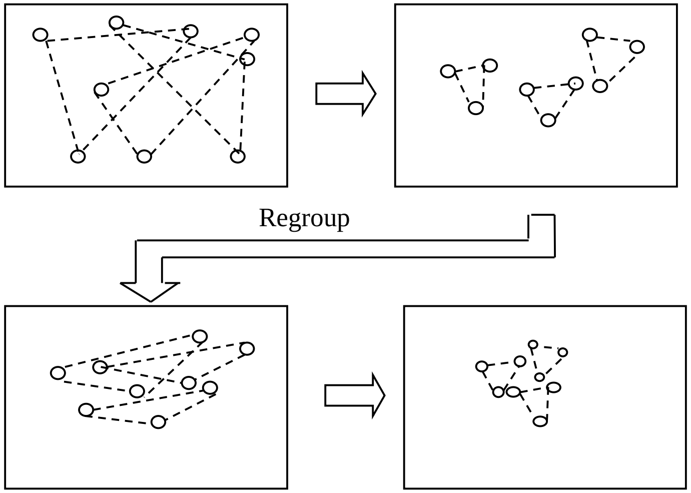
Dynamic Multi-Swarm PSO

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

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

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

Dynamic Multi-Swarm PSO



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

Dynamic Multi-Swarm PSO

- **Adaptive Self-Learning Strategy**

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

If $keep\_id_i^{\ d} = 0$

$$V_i^d \leftarrow 0.729 * V_i^d + 1.49445 * rand1_i^{\ d} * (pbest_i^{\ d} - X_i^{\ d})$$
$$+ 1.49445 * rand2_i^{\ d} * (lbest_k^{\ d} - X_i^{\ d})$$
$$V_i^d = \min(V_{\max}^d, \max(-V_{\max}^d, V_i^d))$$
$$X_i^{\ d} \leftarrow X_i^{\ d} + V_i^d$$

Otherwise

$$X_i^{\ d} \leftarrow pbest_i^{\ d}$$

Somewhat similar to DE

Dynamic Multi-Swarm PSO

- **Adaptive Self-Learning Strategy**
  - Assume **Pc** normally distributed in a range with mean(**Pc**) and a standard deviation of 0.1.
  - Initially, mean(**Pc**) is set at 0.5 and different **Pc** values conforming to this normal distribution are generated for each individual in the current population.
  - During every generation, the **Pc** values associated with the particles which find new **pbest** are recorded.
  - When the sub-swarms are regrouped, the mean of normal distribution of **Pc** is recalculated according to all the recorded **Pc** values corresponding to successful movements during this period. The record of the successful **Pc** values will be cleared once the normal distribution's mean is recalculated.
  - As a result, the proper **Pc** value range for the current problem can be learned to suit the particular problem.

Overview

I. Some General Thoughts

II. Introduction to DE and PSO

III. Future Directions

IV. Further PSO / DE Variants

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.

G Wu, R Mallipeddi, PN Suganthan, Ensemble strategies for population-based optimization algorithms–A survey, Swarm and Evolutionary Computation, 2019

Y. Ren, L. Zhang, and P. N. Suganthan, "Ensemble Classification and Regression – Recent Developments, Applications and Future Directions," IEEE Computational Intelligence Magazine, DOI: 10.1109/MCI.2015.2471235, Feb 2016.

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 …

Reference: N Lynn, MZ Ali, PN Suganthan, Population topologies for particle swarm optimization and differential evolution, Swarm and Evolutionary Computation, 39, 24-35, 2018.

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.

- Do we need 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

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 by frequent reuse in the future generations / iterations.
- 2nd approach is more successful and frequently used in DE.

Population Size Reduction

- Evolutionary algorithms are expected to explore the search space in the early stages

- In the final stages of search, exploitation of previously found good regions takes place.

- For exploration of the whole search space, we need a large population while for exploitation around the top solutions, we need a small population size.

- Hence, population size reduction will be effective for evolutionary algorithms.

Overview

I. Some General Thoughts

II. Introduction to DE and PSO

III. Future Directions

IV. Further PSO Variants

Comprehensive Learning PSO

- **Comprehensive Learning Strategy** :

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

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

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

Comprehensive Learning PSO

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

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

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

Comprehensive Learning PSO

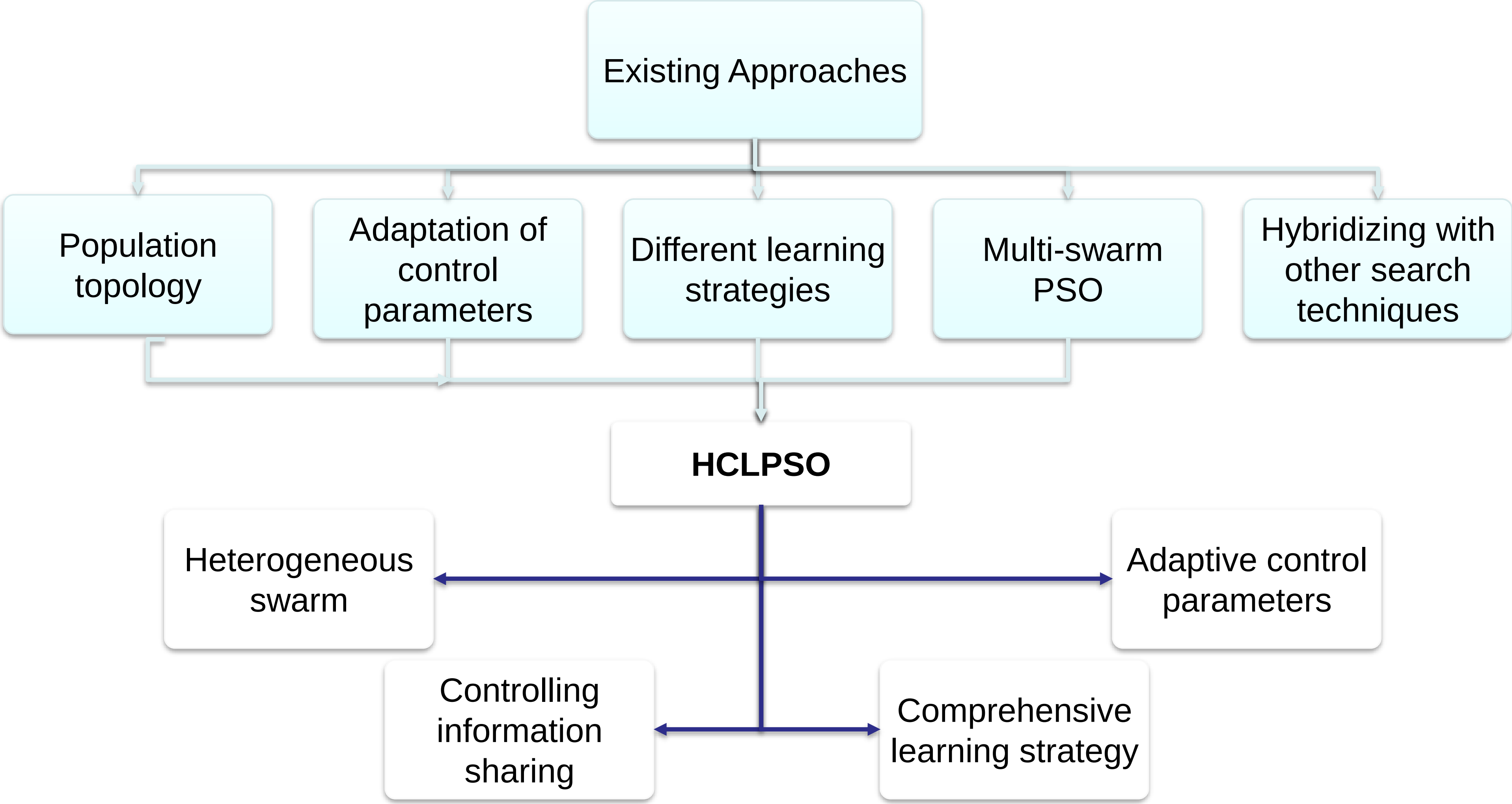
- We observe three main differences between the CLPSO and the original PSO:
  - Instead of using particle's own **pbest** and **gbest/lbest** as the exemplars, all particles' **pbest**s can potentially be used as the exemplars to guide a particle's flying direction.
  - Instead of learning from the same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

Comprehensive Learning PSO

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

Reference: N Lynn, MZ Ali, PN Suganthan, Population topologies for particle swarm optimization and differential evolution, Swarm and Evolutionary Computation, 39, 24-35, 2018.

Heterogeneous CLPSO



Existing Approaches

Population topology

Adaptation of control parameters

Different learning strategies

Multi-swarm PSO

Hybridizing with other search techniques

HCLPSO

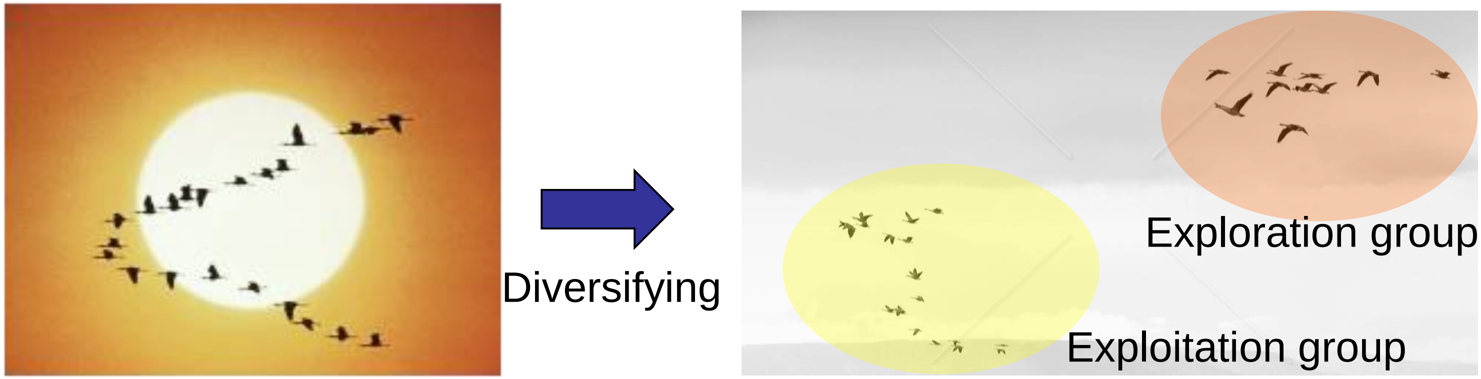
Heterogeneous swarm

Adaptive control parameters

Controlling information sharing

Comprehensive learning strategy

HCLPSO – Heterogeneous Swarm



- Velocity update for Exploration Subpopulation (group 1)

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

- Velocity update Exploitation Subpopulation (group 2)

$$V_i^d = w * V_i^d + c_1 * rand_{1i}^d * \left(pbest_{fi(d)}^d - X_i^d\right) + c_2 * rand_{2i}^d * \left(gbest^d - X_i^d\right)$$

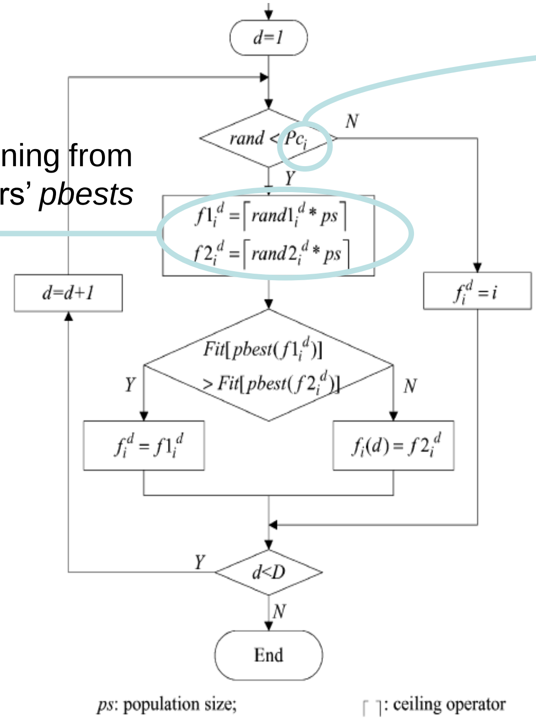
- Position update Subpopulation groups 1 and 2

$$X_i^d = X_i^d + V_i^d$$

HCLPSO – Comprehensive Learning Strategy

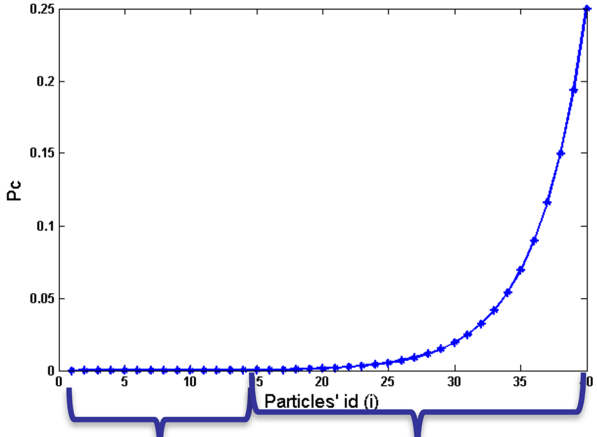
- Generating the exemplar $pbest_{fi(d)}^{d}$

Learning from others' *pbests*



Learning from its own *pbest*

Learning Probability values, P_{ci}



Subpopulation group 1: g_1

Subpopulation group 2: g_2

- ✓ Different particles has different levels of exploration and exploitation ability.
- ✓ g_1 : learns mostly from itself and has strong exploration ability.
- ✓ g_2 : learns from others and *gbest* has strong exploitation ability.

- Two particles are randomly selected and the particle with better fitness is chosen as the exemplar for that dimension.

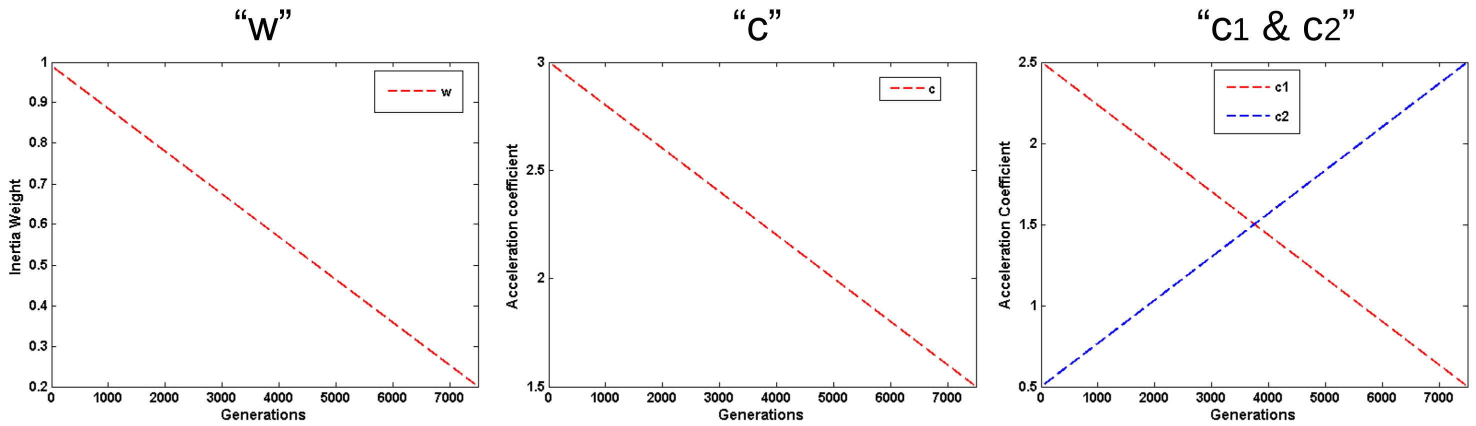
HCLPSO – Adaptive Control Parameters

- Exploration Subpopulation group 1

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

- Exploitation Subpopulation group 2

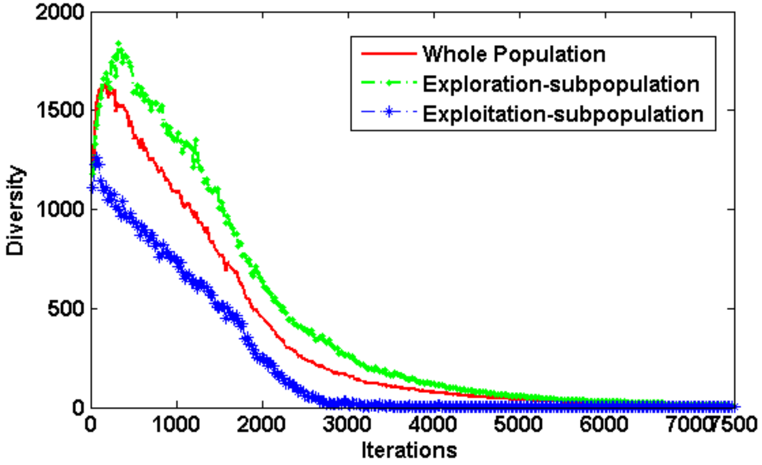
$$V_i^d = w * V_i^d + c_1 * rand_{1i}^d * \left(pbest_{fi(d)}^d - X_i^d\right) + c_2 * rand_{2i}^d * \left(gbest^d - X_i^d\right)$$



- ✓ "*w*" - Adjusting the swarm's behaviour from exploration of the entire search space to exploitation of promising regions
- ✓ "c" - Enhancing Exploration at subpopulation group 1
- ✓ "c1 and c2" - Enhancing Exploitation at subpopulation group 2

HCLPSO – Is it working as designed?

- Are the subpopulations preforming exploration and exploitation respectively?
- Swarm Diversity : "distance-to-average-point" measure
- Diversity measure on unimodal and multimodal problems



- ✓ The explorative particles are not allowed to access information from the exploitative particles, permitting higher diversity.
- ✓ Thus, even if exploitation group suffers from premature convergence, exploration group has potential to rescue the exploitation-oriented group from the local optimum.
- ✓ Exploration and exploitation processes in HCLPSO are enhanced without one process crippling the other.

HCLPSO – Performance Evaluation

- Evaluation is performed according to the evaluation criteria of CEC2005 and CEC2014 benchmark competitions.
  - ✓ Unimodal, multimodal and hybrid composition problems
  - ✓ 10 and 30 dimensional problems
  - ✓ Mean error values, standard deviations and ranking
  - ✓ Convergence characteristic graphs
  - ✓ Non-parametric Wilcoxon signed-rank test
- Compared with other PSO algorithms
  - ✓ PSO
  - ✓ fully-informed PSO (FIPS)
  - ✓ Unified PSO (UPSO)
  - ✓ Comprehensive Learning PSO (CLPSO)
  - ✓ Orthogonal Learning PSO (OLPSO)
  - ✓ Heterogeneous Particle Swarm Optimizer (sHPSO)
  - ✓ Self-organizing hierarchical PSO with time-varying acceleration coefficients (HPSO-TVAC)

HCLPSO – Performance Evaluation on CEC2005

- **Unimodal** (1~5), **Multimodal** (6~14), **Hybrid composition** (15~25) problems
- Best/2nd Best/Worst ranking based on mean and std values
- Wilcoxon signed rank test results - Better/equal/worse (+/0/-)
- Best - 🟥 Worst - □

10D

HCLPSO	PSO	FIPS	UPSO	CLPSO	HPSO-TVAC	sHPSO
1	1	1	1	1	1	7
1	1	1	1	5	1	6
2	5	6	4	7	1	3
3	2	1	5	4	6	7
2	6	3	5	1	4	7
1	3	4	6	2	5	7
1	5	4	2	3	6	7
3	5	7	6	4	2	1
2	5	3	6	1	4	7
2	4	3	5	1	7	6
1	2	4	6	3	5	7
1	7	4	3	2	5	6
2	3	6	5	1	4	7
1	2	3	6	4	5	7
1	5	3	6	2	4	7
1	4	2	5	3	7	6
1	4	2	5	3	6	7
1	3	4	5	2	7	6
2	3	4	5	1	7	6
2	1	4	5	3	6	7
1	2	4	5	3	6	7
1	3	4	5	2	7	6
1	3	4	5	1	6	7
2	3	4	5	1	7	6
1	3	4	5	2	7	6
15/9/0	4/3/1	3/2/1	2/1/1	7/7/2	3/1/7	3/0/12
	20/5/0	24/1/0	25/0/0	12/13/0	25/0/0	25/0/0

30D

HCLPSO	PSO-G	FIPS	UPSO	CLPSO	OLPSO	HPSO-TVAC	sHPSO
1	1	1	1	1	1	1	1
2	5	7	1	8	6	3	4
1	5	8	4	6	7	2	3
2	1	3	5	6	4	8	7
2	4	1	6	5	3	8	7
1	7	4	6	1	3	5	8
2	6	3	2	5	1	1	4
3	4	6	7	5	8	2	1
1	5	6	8	1	1	4	7
1	4	6	2	3	5	8	7
1	2	8	6	3	4	5	7
1	7	8	2	5	4	3	6
1	4	8	6	2	3	5	7
1	4	7	5	3	8	2	6
2	5	4	8	1	3	6	7
1	6	5	4	3	2	7	8
1	5	4	6	3	2	8	7
3	7	1	2	5	4	6	8
5	7	1	2	6	3	4	8
6	7	1	2	5	4	3	8
1	4	6	5	1	1	8	7
3	4	1	2	6	5	8	7
1	4	6	5	1	1	8	7
1	1	5	6	1	1	8	7
1	7	4	5	1	3	8	6
15/5/0	3/3/1	5/1/4	1/7/2	8/1/1	6/4/2	2/4/9	2/0/5
	21/4/0	19/1/5	18/2/5	17/7/1	15/8/2	18/5/2	21/4/0

HCLPSO – Performance Evaluation on CEC2014

- **Unimodal** (1~3),**Multimodal** (4~16),**Hybrid** (17~21),**Composition** (22-30) problems
- Best/2nd Best/Worst ranking based on mean and std values
- Wilcoxon signed rank test results - Better/equal/worse (+/0/-)
- Best - 🟥 Worst - □

10D

HCLPSO	PSO	FIPS	UPSO	CLPSO	HPSO-TVAC	sHPSO
2	4	3	1	7	6	5
2	6	3	5	1	4	7
2	3	5	4	1	6	7
3	7	5	4	1	2	6
1	7	3	5	2	4	6
2	4	1	5	3	6	7
3	5	4	2	1	6	7
2	5	3	6	1	4	7
2	4	3	5	1	6	7
2	4	3	7	1	5	6
2	4	3	6	1	5	7
1	6	7	4	2	3	5
1	5	2	4	3	6	7
1	3	4	5	2	7	6
1	4	5	3	2	6	7
1	3	4	5	2	6	7
1	5	3	4	6	2	7
2	6	4	3	1	5	7
3	5	2	4	1	6	7
1	3	6	4	2	5	7
2	1	3	6	4	5	7
2	3	4	5	1	6	7
5	5	3	3	2	7	1
3	4	1	5	2	6	7
2	5	1	4	3	6	7
1	3	2	7	4	5	6
1	5	4	3	2	7	6
2	5	4	7	3	1	6
3	7	4	5	2	1	6
2	3	4	7	1	5	6
10/14/0	1/0/3	3/3/1	1/1/4	12/10/1	2/2/3	1/0/19
	25/4/1	20/7/3	25/3/2	12/10/8	26/2/2	27/2/1

30D

HCLPSO	PSO	FIPS	UPSO	CLPSO	OLPSO	HPSO-TVAC	sHPSO
1	7	5	3	8	6	2	4
2	1	5	8	3	4	6	7
1	2	7	6	4	3	5	8
2	8	5	1	6	7	4	3
4	7	8	3	6	5	1	2
2	4	1	8	5	3	6	7
4	7	2	5	3	1	6	8
1	5	6	8	3	1	4	7
2	4	7	6	3	1	5	8
3	5	6	8	2	1	4	7
1	4	8	6	5	2	3	7
1	7	8	4	5	2	3	6
1	8	3	4	5	2	6	7
1	7	4	2	6	3	5	8
1	2	6	4	5	3	8	7
1	4	7	5	2	6	3	8
3	6	5	2	8	7	1	4
1	8	6	4	2	3	5	7
1	4	2	6	5	3	8	7
2	1	3	6	5	7	4	8
1	5	6	2	4	7	3	8
3	5	2	6	1	4	8	7
3	8	6	6	3	3	2	1
3	7	1	5	4	2	6	8
1	4	2	7	3	5	8	6
3	6	7	4	1	1	5	8
3	6	2	5	4	1	7	8
3	6	2	8	5	4	1	7
2	8	6	3	4	5	1	7
3	7	2	6	8	5	1	4
13/6/0	2/2/5	2/6/3	1/3/5	2/3/3	6/4/0	5/2/4	1/1/11
	26/3/1	20/7/3	23/4/3	21/8/1	N.A	24/1/5	27/2/1

HCLPSO – Performance Evaluation on CEC2013

- **Unimodal/Rotated Unimodal** (1~5), **Multimodal/Rotated Multimodal** (6~20),**Composition** (21-28) problems
- HCLPSO outperforms social learning PSO (SL-PSO) on 16 out of 28 CEC 2013 50 D problems

Functions	HCLPSO	SL-PSO
F1	1.18E-12 2.19E-13	**2.05E-13** **7.19E-14**
F2	4.06E+06 1.94E+06	**2.22E+06** **5.90E+05**
F3	3.70E+08 4.42E+08	**1.31E+07** **2.13E+07**
F4	**7.72E+03** **2.24E+03**	3.82E+04 3.11E+03
F5	5.51E-11 3.50E-11	**1.82E-13** **6.23E-14**
F6	4.65E+01 1.24E+00	**4.52E+01** **2.33E+00**
F7	4.37E+01 1.05E+01	**7.49E+00** **1.51E+00**
F8	**2.11E+01** **5.38E-02**	2.12E+01 3.44E-02
F9	3.94E+01 9.42E+00	**1.82E+01** **1.75E+00**
F10	1.09E+00 3.85E-01	**2.41E-01** **1.06E-01**
F11	**1.10E+01** **2.61E+00**	2.65E+01 5.87E+00
F12	**1.17E+02** **3.12E+01**	3.39E+02 2.11E+01
F13	**2.48E+02** **4.14E+01**	3.43E+02 2.45E+01
F14	**3.38E+02** **1.06E+02**	1.08E+03 3.95E+02
F15	**7.68E+03** **9.05E+02**	1.23E+04 3.16E+03
F16	**2.45E+00** **4.71E-01**	3.33E+00 3.27E-01
F17	**8.17E+01** **7.81E+00**	3.70E+02 1.41E+01
F18	**1.86E+02** **3.98E+01**	3.97E+02 9.63E+00
F19	**3.59E+00** **7.64E-01**	9.18E+00 5.24E+00
F20	**1.98E+01** **7.36E-01**	2.24E+01 3.47E-01
F21	**7.45E+02** **3.59E+02**	7.60E+02 4.06E+02
F22	**6.34E+02** **2.62E+02**	1.14E+03 3.05E+02
F23	**8.68E+03** **1.23E+03**	1.10E+04 4.27E+03
F24	2.72E+02 1.35E+01	**2.46E+02** **8.35E+00**
F25	3.57E+02 1.57E+01	**2.94E+02** **8.23E+00**
F26	**2.07E+02** **3.36E+01**	3.35E+02 7.91E+00
F27	1.17E+03 1.58E+02	**7.29E+02** **8.12E+01**
F28	4.00E+02 1.17E-05	4.00E+02 2.05E-13

Overview

I. Some General Thoughts

II. Introduction to DE and PSO

III. Future Directions

IV. Further DE Variants

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*., 2005, 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*).

A. K. Qin and P. N. Suganthan, “Self-adaptive Differential Evolution Algorithm for Numerical Optimization”, *IEEE Congress on Evolutionary Computation*, pp.1785-1791, Edinburgh, UK, 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.

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*, 13(5) pp. 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:

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

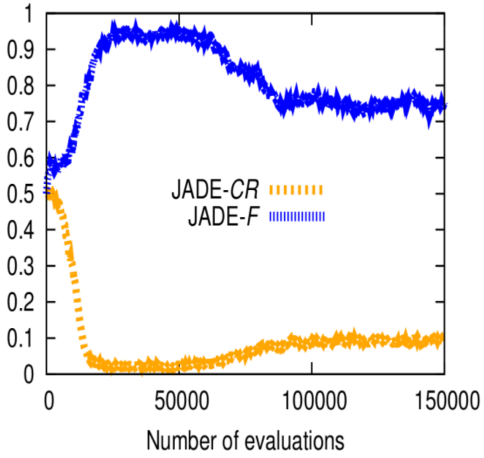
JADE usually performs best with 1/*c* chosen from [5, 20] and *p* from [5%, 20%]

Success-History based Adaptive DE (SHADE)

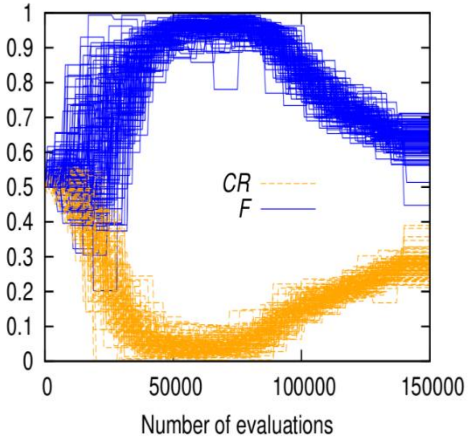
- An improved version of JADE
- Uses a success-history based adaptation
  - Based on a historical memory of successful parameter settings that were previously used found during the run
  - A historical memory M_{CR} , M_F are used, instead of adaptive parameter u_{cr} , u_F
  - This improves the robustness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

Fig. Their adaptation behaviors on Rastrigin (30 dimensions)

JADE uses a single pair u_{cr}, u_F

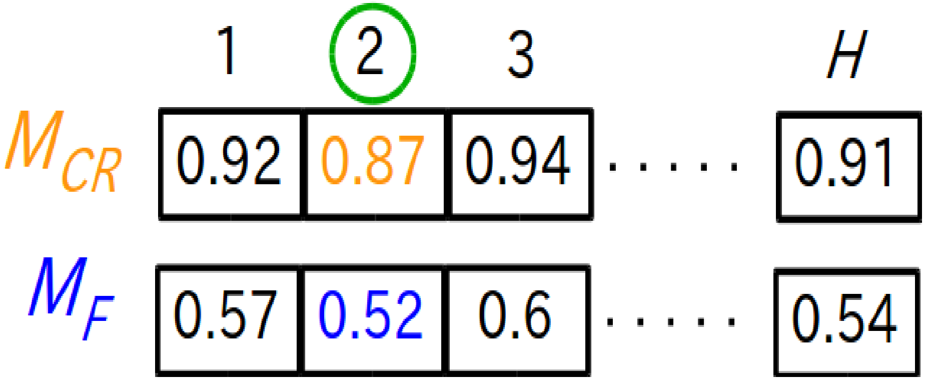


SHADE maintains a diverse set of parameters in a historical memory M_{CR} , M_F



SHADE

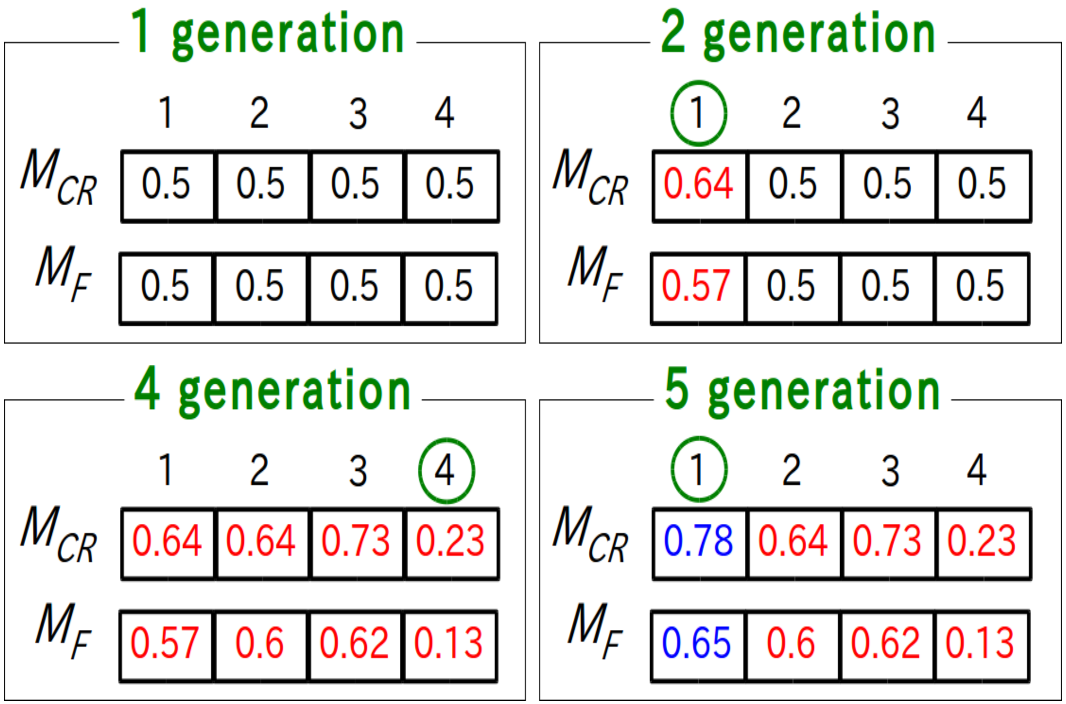
- The weighted Lehmer mean (in CEC’14 ver.) values of S_{CR} and S_F , which are successful parameters for each generation, are stored in a historical memory M_{CR} and M_F



	1	2	3		H
M_{CR}	0.92	0.87	0.94	· · · · ·	0.91
M_F	0.57	0.52	0.6	· · · · ·	0.54

- *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



1 generation

	1	2	3	4
M_{CR}	0.5	0.5	0.5	0.5
M_F	0.5	0.5	0.5	0.5

2 generation

	1	2	3	4
M_{CR}	0.64	0.5	0.5	0.5
M_F	0.57	0.5	0.5	0.5

4 generation

	1	2	3	4
M_{CR}	0.64	0.64	0.73	0.23
M_F	0.57	0.6	0.62	0.13

5 generation

	1	2	3	4
M_{CR}	0.78	0.64	0.73	0.23
M_F	0.65	0.6	0.62	0.13

- 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

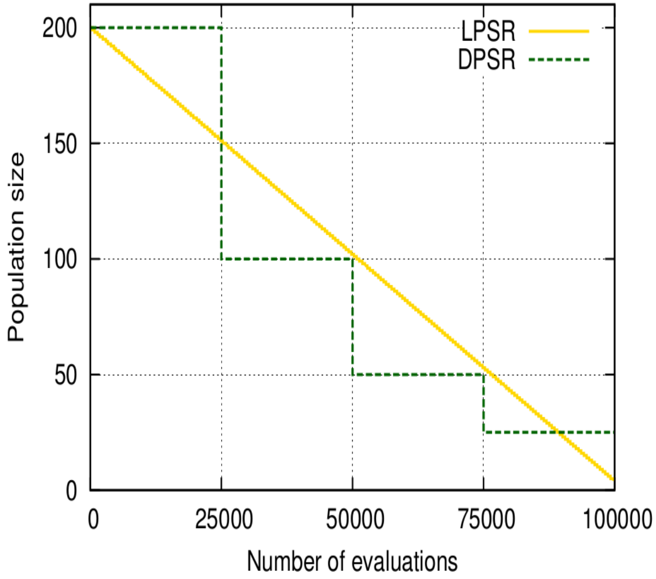


Fig. Comparison of population resizing schedule between LPSR and DPSR (# of reduction = 4)

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

jSO (2nd place in CEC 2017 Competition)

L-SHADE and iL-SHADE apply DE/current-to-*p*Best/1 mutation strategy to generate a trial vector:

$$\vec{v}_{i,g} = \vec{x}_{i,g} + F(\vec{x}_{pBest,g} - \vec{x}_{i,g}) + F(\vec{x}_{r_1,g} - \vec{x}_{r_2,g}), \qquad (2)$$

while jSO uses a new weighted version of mutation strategy, called DE/current-to-*p*Best-w/1, as follows:

$$\vec{v}_{i,g} = \vec{x}_{i,g} + F_w(\vec{x}_{pBest,g} - \vec{x}_{i,g}) + F(\vec{x}_{r_1,g} - \vec{x}_{r_2,g}), \qquad (3)$$

where F_w is calculated:

$$F_w = \begin{cases} 0.7 * F, & nfes < 0.2max\_nfes, \\ 0.8 * F, & 0.2max\_nfes < nfes < 0.4max\_nfes, \\ 1.2 * F, & \text{otherwise.} \end{cases} \qquad (4)$$

Comparison

- L-SHADE, 2014
- iL-SHADE, 2016
- jSO, 2017

Changes and extensions from the iL-SHADE to jSO are minors — the main features of iL-SHADE remain unchanged. And also, the differences between iL-SHADE and L-SHADE are not so big.

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.

Ensemble of differential evolution variants (EDEV)

Motivation

- Ensemble methods receive an increasing attention in designing high-quality DE algorithms
- Previous efforts are mainly devoted to the low-level ensemble of mutation strategies of DE, parameter values, etc.
- This work investigates the high-level ensemble of multiple existing efficient DE variants

(DE) Guohua Wu, X. Shen, H. Li, P.N. Suganthan, Ensemble of differential evolution variants , ***Information Sciences***, 423 (2018): 172-186.

(PSO) N. Lynn, P. N. Suganthan, “Ensemble particle swarm optimizer,” Applied Soft Computing, 55, 533-548, 2017. **Codes Available**

For combinatorial problems: “Algorithm portfolios” or “Ensemble of local search operators”, “hyper Heuristics”, etc.

EDEV-Implementation

Constituent DE variants

- **JADE** (adaptive differential evolution with optional external archive)
- **CoDE** (differential evolution with composite trial vector generation strategies and control parameters)
- **EPSDE** (differential evolution algorithm with ensemble of parameters and mutation strategies)

EDEV – Implementation

Ensemble Framework

•Each constituent DE variant has a minimum population resources **named indicator sub-population** denoted by ***pop***$_1$, ***pop***$_2$ and ***pop***$_3$, respectively.

•Recent best performing DE variant is rewarded by an extra population resources named **reward sub-population** denoted by ***pop***$_4$

Remarks:

•Three indicator sub-populations have relatively small size

•The reward sub-population has larger size.

EDEV – Implementation

Ensemble Framework

•At the beginning, $\boldsymbol{pop}_1$, $\boldsymbol{pop}_2$ and $\boldsymbol{pop}_3$, are assigned to three constituent DE variants, respectively.

•$\boldsymbol{pop}_4$ is randomly assigned to one DE variant.

•After every ***ng*** generations, the performance of each DE variant is evaluated by the metric

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

•The determined best-performed DE variant will occupy the reward population resource in the following ***ng*** generations.

EDEV – Implementation

Conclusions

- Through this framework, the most efficient DE variant is expected to obtain the most computational resources during the optimization process
- Population partition operator is triggered at every generation to facilitate to timely information sharing and tight cooperation
- Extensive experiments and comparisons on the CEC2005 and CEC2014 benchmark suit show the superiority of EDEV

Overview

I. Some DE and PSO Variants for Single Objective Optimization

II. Constrained Optimization by DE

III. Multimodal Optimization by DE and PSO

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

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

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

  subjected to

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

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

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 poor 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)

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

- The recommended parameter settings are

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

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.

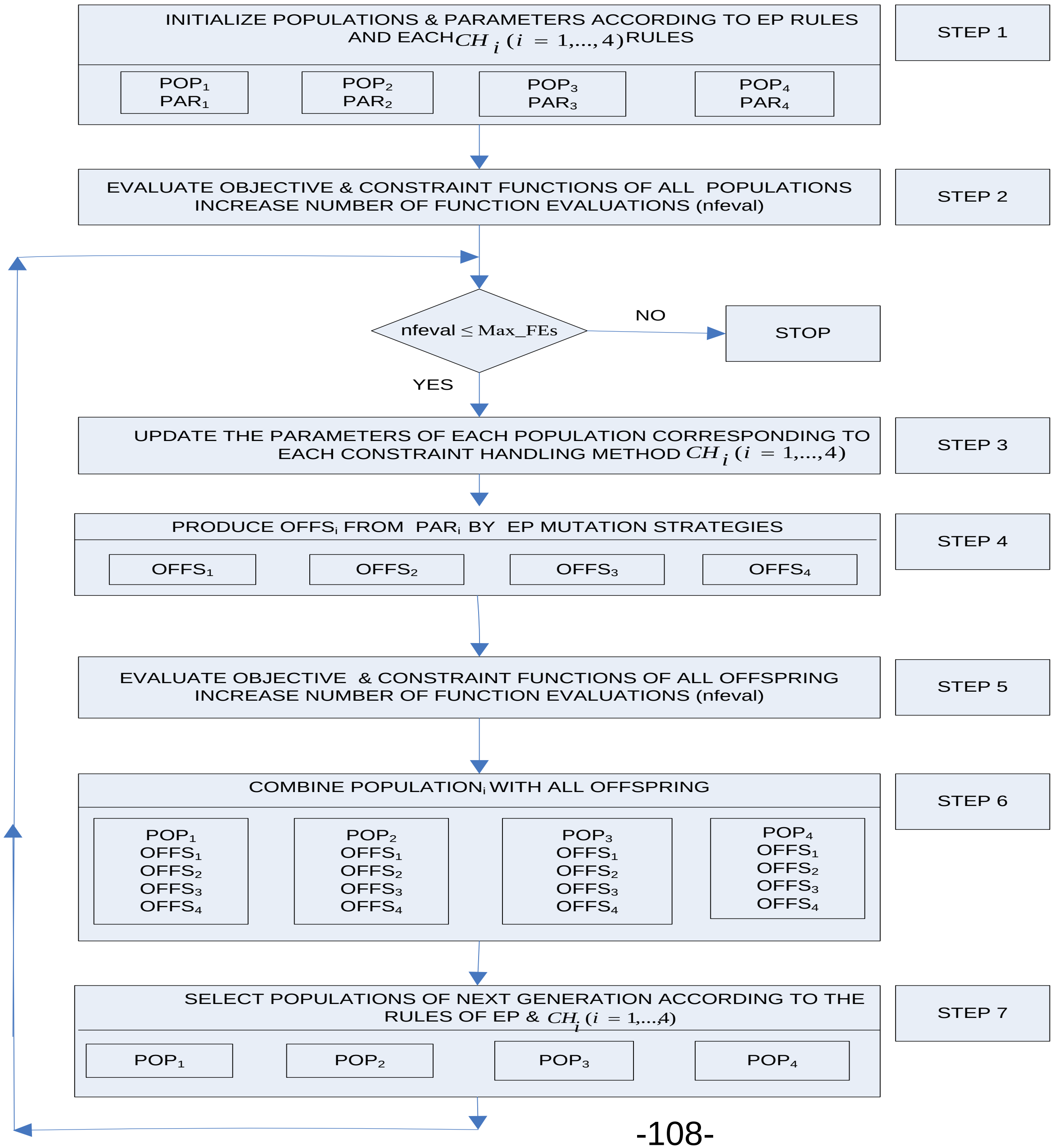
ECHT: MOTIVATION

- 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



INITIALIZE POPULATIONS & PARAMETERS ACCORDING TO EP RULES AND EACH $CH_i \ (i = 1,\dots,4)$ RULES

POP₁ PAR₁ | POP₂ PAR₂ | POP₃ PAR₃ | POP₄ PAR₄

STEP 1

EVALUATE OBJECTIVE & CONSTRAINT FUNCTIONS OF ALL POPULATIONS
INCREASE NUMBER OF FUNCTION EVALUATIONS (nfeval)

STEP 2

nfeval $\leq$ Max_FEs

NO → STOP

YES

UPDATE THE PARAMETERS OF EACH POPULATION CORRESPONDING TO EACH CONSTRAINT HANDLING METHOD $CH_i \ (i = 1,\dots,4)$

STEP 3

PRODUCE OFFS$_i$ FROM PAR$_i$ BY EP MUTATION STRATEGIES

OFFS₁ | OFFS₂ | OFFS₃ | OFFS₄

STEP 4

EVALUATE OBJECTIVE & CONSTRAINT FUNCTIONS OF ALL OFFSPRING
INCREASE NUMBER OF FUNCTION EVALUATIONS (nfeval)

STEP 5

COMBINE POPULATION$_i$ WITH ALL OFFSPRING

POP₁ OFFS₁ OFFS₂ OFFS₃ OFFS₄ | POP₂ OFFS₁ OFFS₂ OFFS₃ OFFS₄ | POP₃ OFFS₁ OFFS₂ OFFS₃ OFFS₄ | POP₄ OFFS₁ OFFS₂ OFFS₃ OFFS₄

STEP 6

SELECT POPULATIONS OF NEXT GENERATION ACCORDING TO THE RULES OF EP & $CH_i \ (i = 1,\dots,4)$

POP₁ | POP₂ | POP₃ | POP₄

STEP 7

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)

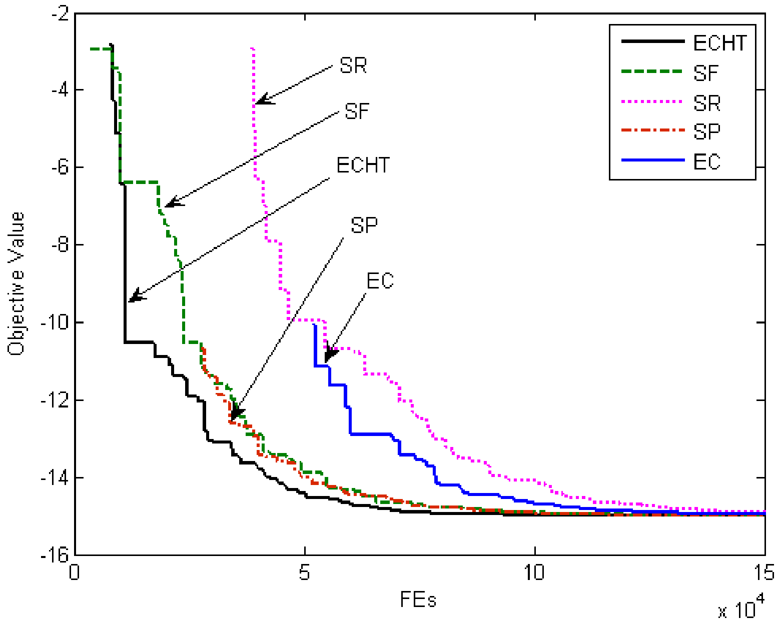
3.Stochastic Ranking (SR)

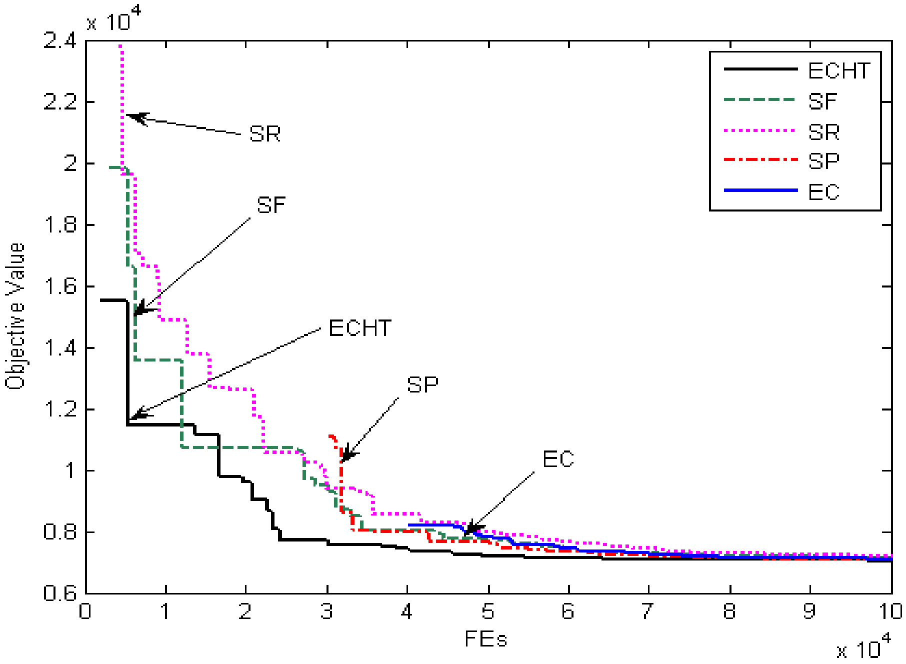
4.Epsilon Constraint handling (EC)

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 $h_j(X)=0$ $(1\leq j\leq m)$, 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: $g_i(X) \le 0$, $i = 1, \ldots, p$

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

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

Variable reduction operate

Minimize: $f(X)$

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

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

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

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

VRS: Implementation

- Naïve example, considering:

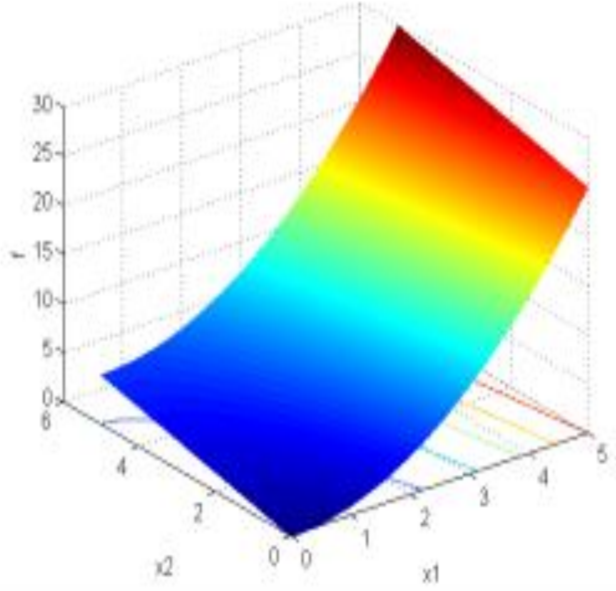
$$\begin{aligned}&\min {x_1}^2 + {x_2}^2\\&x_1 + x_2 = 2\\&0 \le x_1 \le 5, 0 \le x_2 \le 5\end{aligned}$$

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

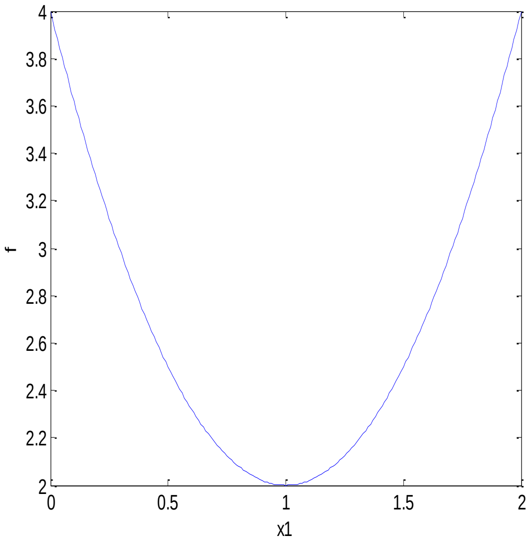
$$\begin{aligned}&\min 2{x_1}^2 - 4x_1 + 4\\&0 \le x_1 \le 2\end{aligned}$$

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
- Expand it and we get

$$AX = b$$

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

$$m < n$$

- We can transform the expanded form into

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

- Let

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

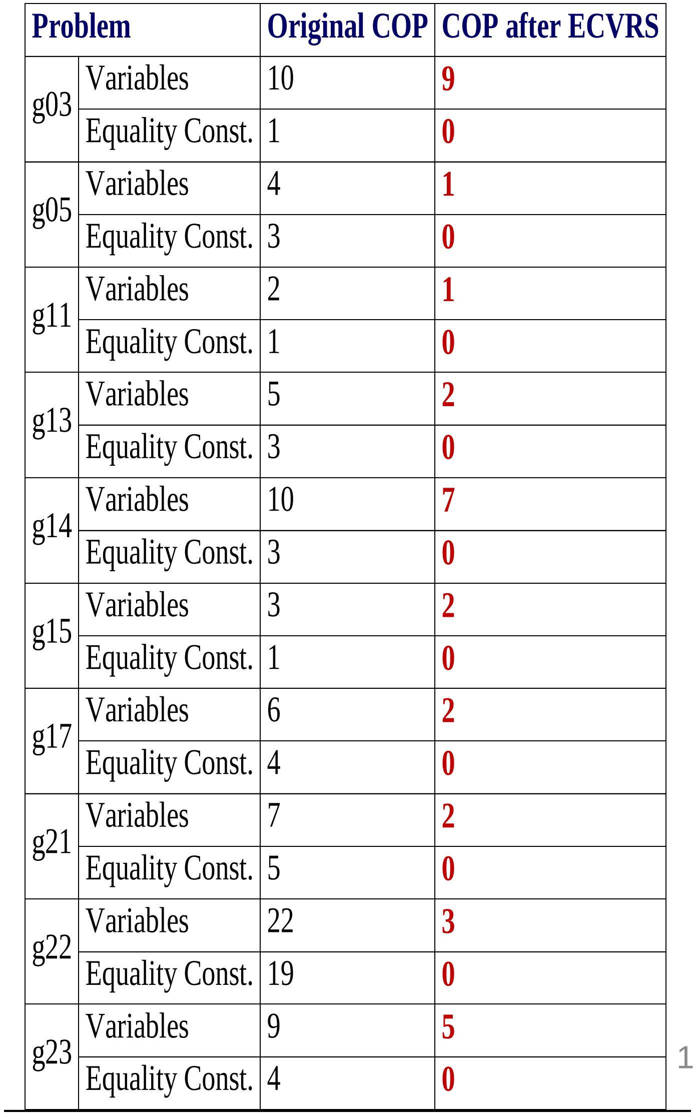
- We have

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

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**

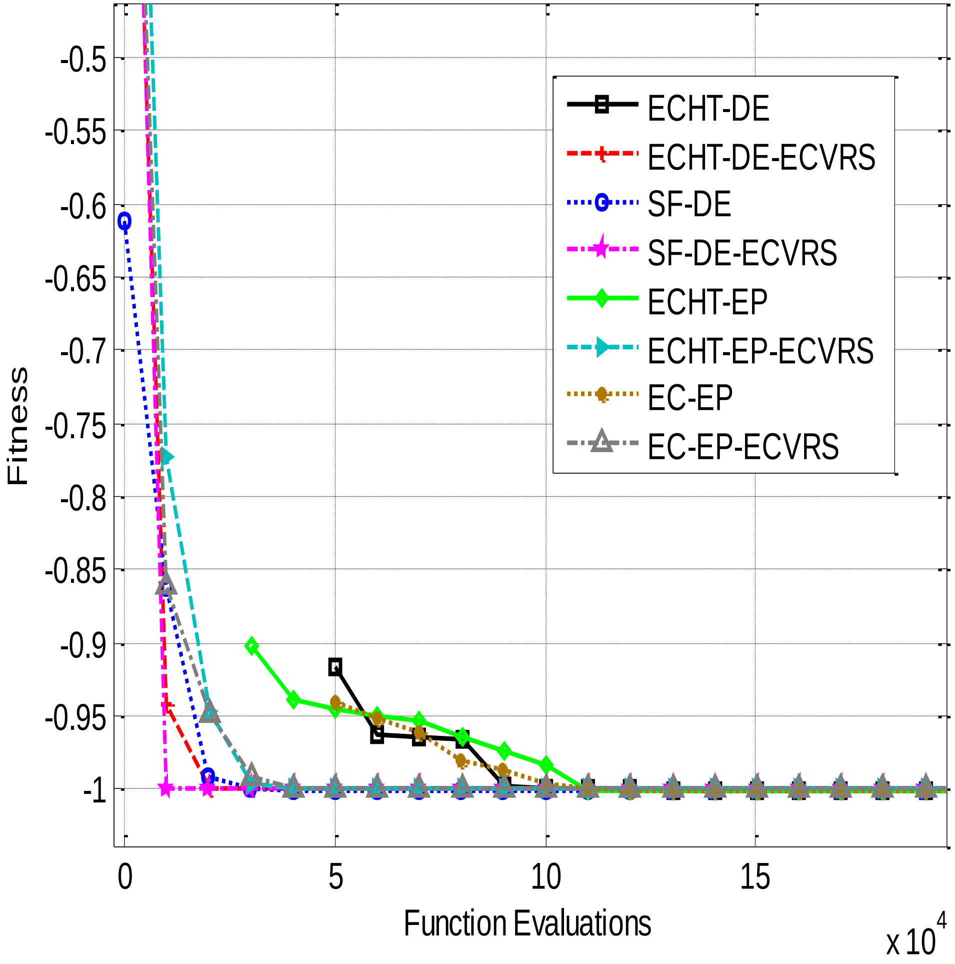
VRS: Results

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.

Using VRS to deal with active inequality constraints

The expression of an **active inequality constraint** is equivalent to zero with respect to the optimal solution.

Active inequality constraints act as **equality constraints** when solutions found by EAs are approaching the optimal solution.

It is difficult to handle **active inequality constraint:**

- First, it is hard for EAs to precisely locate the bounds of active inequality constraints;
- Second, solutions found by EAs are more likely to be infeasible in the presence of active inequality constraints.

G Wu, W Pedrycz, PN Suganthan, H Li, Using Variable Reduction Strategy to Accelerate Evolutionary Optimization, Applied Soft Computing, 61, 283-293, 2017.

Using VRS to deal with active inequality constraints

During the search process of a used EA, if the current best solution X_g is feasible or very close to being feasible and the i th inequality constraint satisfies

$$|g_i(X_g)| \leq \xi \qquad i \in \{1, \ldots, p\}$$

g_i is assumed to be active.

We set g_i to be equivalent to zero

$$g_i(X) = 0$$

A series of variable relationships may be derived from the equation above.

For any variable x_k , $k \in \{1, \ldots, n\}$, if there is a relationship described as

$$x_k = R_{i,k}(x_i \mid i = 1, \ldots, n, i \neq k) \qquad k \in \{1, \ldots, n\}$$

Using VRS to deal with active inequality constraints

we can generate a new solution by recalculating the value of x_k with the aid of the values of other variables in the current best solution and relationship

$$x_k = R_{i,k}(x_i \mid i = 1,\ldots,n, i \neq k) \qquad k \in \{1,\ldots,n\}$$

If explicit variable relationships cannot be obtained in nonlinear active inequality constraints, one-dimension search methods (e.g. golden section search) can be used to calculate the value of the variable to be reduced.

If there are *q* variables involved in an active inequality constraint, *q* additional solutions can be generated accordingly

Using VRS to deal with active inequality constraints

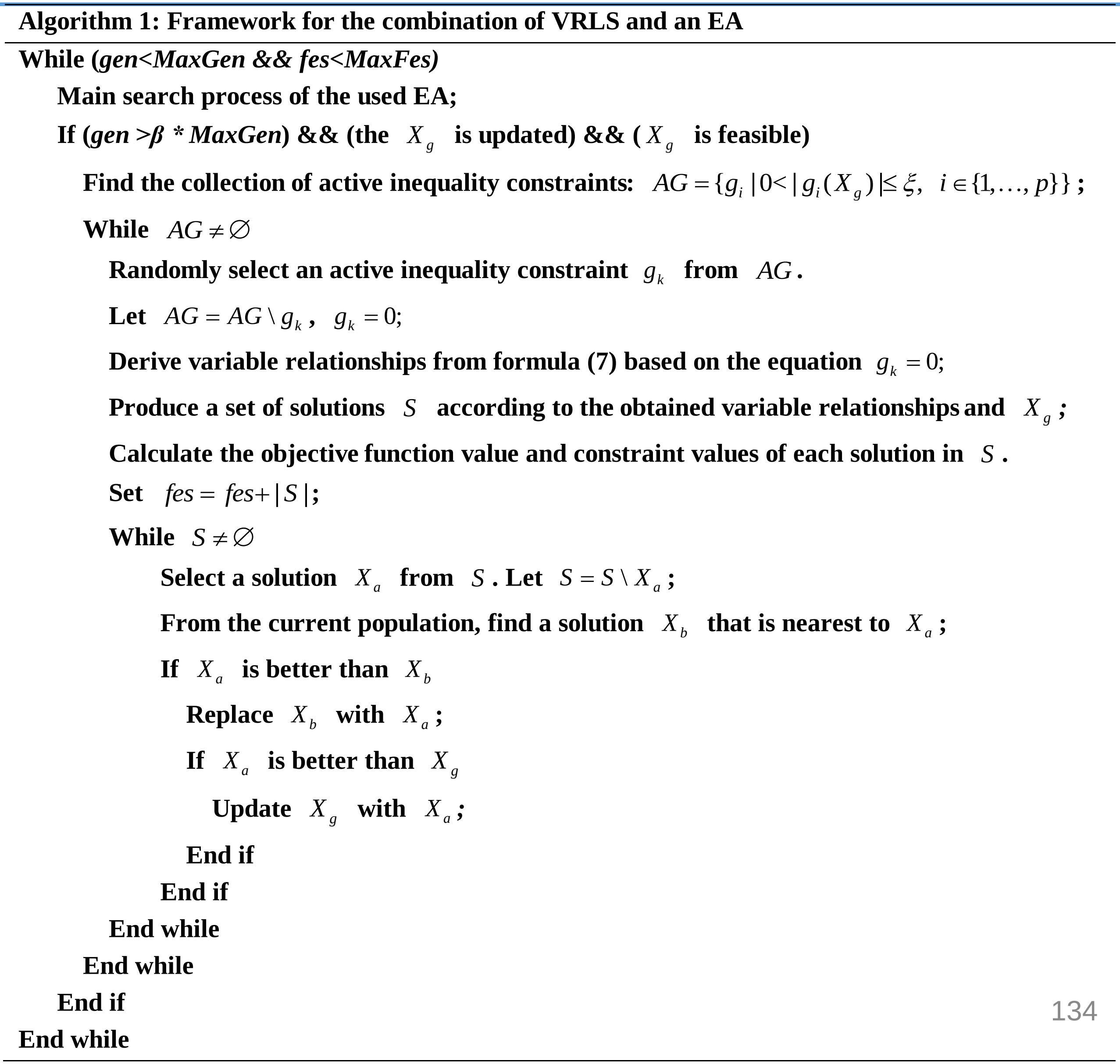
Each generated solution is compare it with the solutions in the current population.

Through this way, the current best solution near the optimal basin could converge to the optimal solution more quickly and precisely, and thus the exploitation capabilities of EAs are enhanced. This technique is referred to as variable relationship based local search (VRLS).

In practice, VRLS can be combined with any EA. As it is a local search method, it is suggested to be used in the late evolutionary stage.

Using VRS to deal with active inequality constraints

Framework for the combination of VRLS and an EA



Algorithm 1: Framework for the combination of VRLS and an EA

```
While (gen<MaxGen && fes<MaxFes)
    Main search process of the used EA;
    If (gen >β * MaxGen) && (the $X_g$ is updated) && ($X_g$ is feasible)
        Find the collection of active inequality constraints: $AG = \{g_i \mid 0 < |g_i(X_g)| \le \xi,\ i \in \{1, \ldots, p\}\}$;
        While $AG \ne \varnothing$
            Randomly select an active inequality constraint $g_k$ from $AG$.
            Let $AG = AG \setminus g_k$, $g_k = 0$;
            Derive variable relationships from formula (7) based on the equation $g_k = 0$;
            Produce a set of solutions $S$ according to the obtained variable relationships and $X_g$;
            Calculate the objective function value and constraint values of each solution in $S$.
            Set $fes = fes + |S|$;
            While $S \ne \varnothing$
                Select a solution $X_a$ from $S$. Let $S = S \setminus X_a$;
                From the current population, find a solution $X_b$ that is nearest to $X_a$;
                If $X_a$ is better than $X_b$
                    Replace $X_b$ with $X_a$;
                    If $X_a$ is better than $X_g$
                        Update $X_g$ with $X_a$;
                    End if
                End if
            End while
        End while
    End if
End while
```

Overview

I. Some DE and PSO Variants for Single Objective Optimization

II. Constrained Optimization by DE

III. Multimodal Optimization by DE and PSO

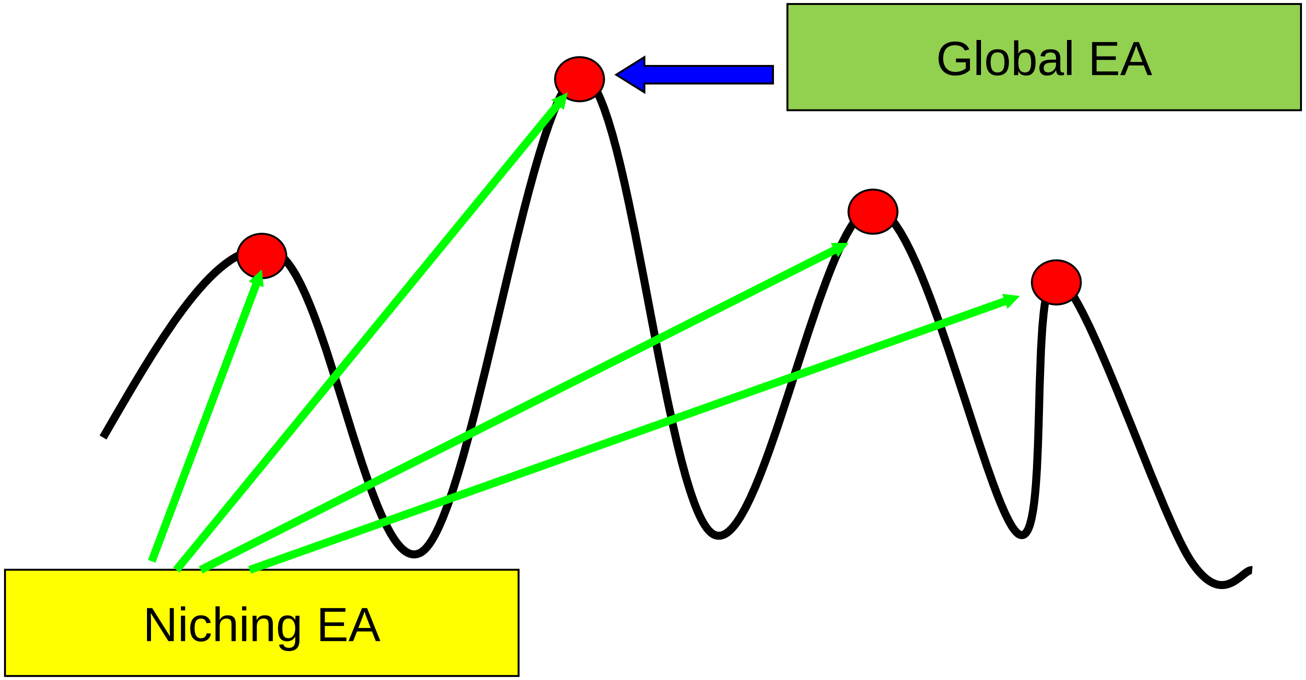
S. Das, S. Maity, B-Y Qu, P. N. Suganthan, "Real-parameter evolutionary multimodal optimization — A survey of the state-of-the-art", Vol. 1, No. 2, pp. 71-88, *Swarm and Evolutionary Computation*, June 2011.

CEC10 Special Session on Niching **Introduces novel scalable test problems:**

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

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 Eucidean Neighborhood Topology based DE/PSO

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

Multi-modal Optimization - RTS

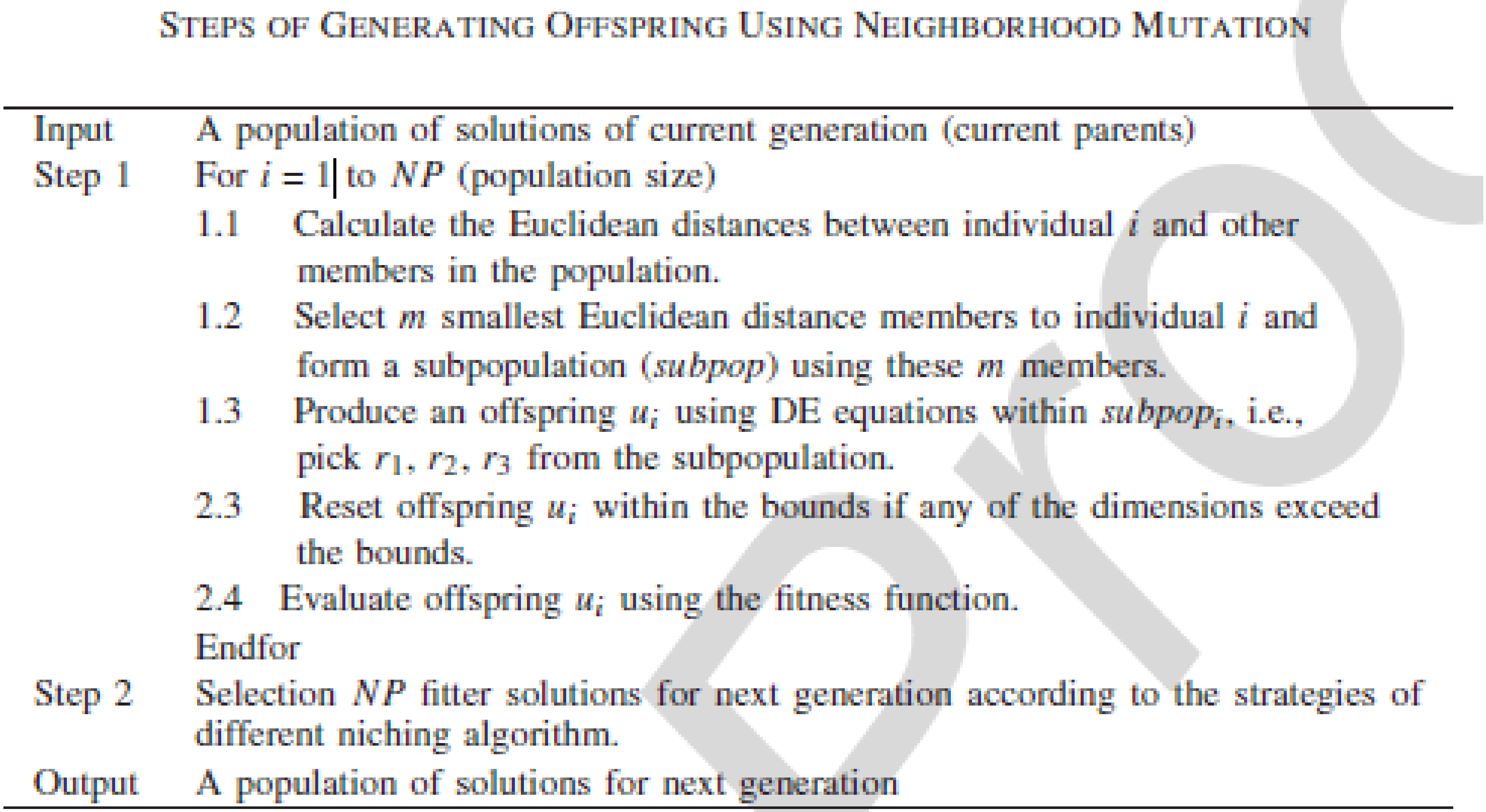
- Restricted Tournament Selection (RTS)
  - ○ Proposed by Harick
  - ○ Similar to Crowding
  - ○ Corresponding to each offspring randomly choose *w* individuals from the population

  w – window size

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

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

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.

Advantages of LIPS

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

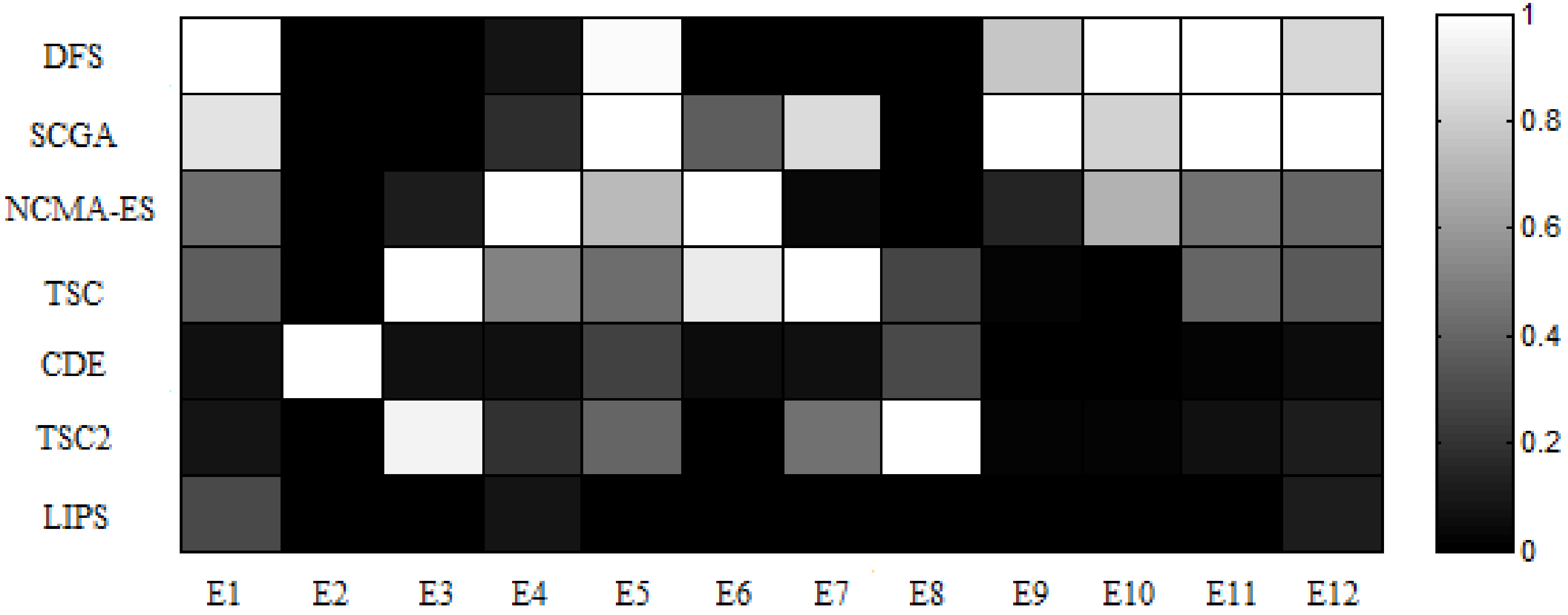
2. Euclidean distance based neighborhood selection to ensure the neighborhoods are from the same niches which increase the algorithm's local search and fine tuning ability

B-Y Qu, P. N. Suganthan, S. Das, "**A Distance-Based Locally Informed Particle Swarm Model for Multi-modal Optimization,**", *IEEE Trans on Evolutionary Computation,* **DOI:** 10.1109/TEVC.2012.2203138. (Supplementary file), 2013. Codes Available

Steps of LIPS

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

Results



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

THANK YOU

Q & A