Evolutionary Optimization Algorithms (EOAs): Comparison, Selection and Application

Presenter: Kai QIN
MISTIS Team
INRIA Grenoble - Rhône-Alpes

October 19th 2010
EOA: A First Glance

- Computer science
  - Artificial intelligence  - Computational intelligence  - Evolutionary computation  - EOA

- Research forums
  - Journals
    - Evolutionary Computation (2009 SCI impact factor: 3.103)
      At the 2nd among 130 "Computer Science, Artificial Intelligence" journals, the 3rd among 91 "Computer Science, Theory & Methods" journals, and the 4th among all 426 Computer Science journals. According to the 5 year impact factor (7.621), IEEE TEVC was ranked the 2nd, 2nd and 4th, respectively, in the above categories.
    - Genetic Programming and Evolvable Machines (2009 SCI impact factor: 1.091)

  - Conferences
    - The Genetic and Evolutionary Computation Conference (GECCO)
    - IEEE Congress on Evolutionary Computation (CEC)
    - The Evo* conference (the main European events on evolutionary computation)

- Most cited literature
  
  
  Google scholar citation: 35919 (till 2010.10.18)
EOA: Applications

- **Computer vision**
  - Image enhancement
  - Image reconstruction
  - Image registration
  - Image segmentation
  - Image classification
  - Texture synthesis
  - Calibration
  - Tracking

- **Pattern Recognition**
  - Evolutionary clustering
  - Evolutionary classification
    - Evolutionary parameter estimation
    - Evolving neural networks
    - Learning classifier systems

- **Statistical modeling**
  - Evolutionary parameter estimation

---

- **Bioinformatics**
  - Alignment and comparison of DNA, RNA, and protein sequences
  - Gene mapping on chromosomes
  - Gene finding and promoter identification from DNA sequences
  - Gene selection from microarray gene expression data
  - Gene regulatory network identification
  - Construction of phylogenetic trees for studying evolutionary relationship
  - DNA/RNA structure prediction
  - Protein structure prediction and classification
  - Molecular design and molecular docking
Outline

Part I: Introduction of EOAs
- Basic ideas
- Generic paradigm and core components
- Historical perspective
- Categorization
- Canonical EOAs
- State-of-the-art EOAs
- Benchmarking testbeds
- Challenges in practice

Part II: Two advanced EOAs: CLPSO and SaDE
- Particle swarm optimization (PSO)
- My previous work: comprehensive learning PSO (CLPSO)
- Differential evolution (DE)
- My previous work: self-adaptive DE (SaDE)

Part III: Research plan discussion
- Statistical comparison and interactive selection of EOAs
- Development of novel EOAs
- Applications
Part I: General Introduction of EOAs

- Basic ideas
- Generic paradigm and core components
- Historical perspective
- Categorization
- Canonical EOAs
- State-of-the-art EOAs
- Benchmarking testbeds
- Challenges in practice
Optimization

- **Optimization** seeks to improve performance towards some optimal point(s).
  - Destination (theoretical viewpoint)
  - Improvement (practical viewpoint)

- **Optimization problem** aims at minimizing (or maximizing) a real-valued objective function by choosing the values of decision variables from within an allowed set.

\[
\text{Maximize (or minimize) } f(x) = (f_1(x), \ldots, f_M(x)) \quad \text{w.r.t. } x = (x_1, \ldots, x_D), \quad f : \Omega^D \rightarrow \mathbb{R}^M
\]

\[
\text{s.t. } x \in [x_{\text{min}}, x_{\text{max}}]; \quad g_j(x) \leq 0, \quad j = 1, \ldots, J; \quad h_k(x) = 0, \quad k = 1, \ldots, K
\]

- A large number of scientific and engineering problems can be formulated as optimization problems, and solved by some computational methods called **optimization algorithms**.

- Categorization of optimization algorithms
  - As per problem properties
    - Decision variables: discrete vs. continuous vs. mixed; constrained vs. unconstrained
    - Objective function: single-objective vs. multi-objective
    - Fitness landscape: unimodal vs. multimodal; noise-free vs. noisy; static vs. dynamic
  - As per algorithm characteristics
    - Deterministic vs. stochastic
EOA: Basic Ideas

- Evolutionary optimization algorithms (EOAs) are computational methods inspired by natural evolution process characterized by Darwin’s *survival of the fittest* principle.

- The key idea of EOAs is to identify and capture computationally useful aspects of natural evolution processes instead of faithfully (or even plausibly) modeling of biological processes.

- EOAs are meta-heuristic, stochastic search techniques, effective for solving complex optimization problems with multi-modal, static/dynamic multi-dimensional search space in the black-box manner, i.e., the landscape model of the search space is not required in advance.

- EOAs iteratively evolve populations of candidate solutions using parallel guided random search to achieve the desired end.

- Major characteristics of an EOA
  - Populations of candidate solutions
  - Fitness-guided random population changing during the course of search
    - Variation with inheritance
    - Replacement
EOA: Generic Paradigm

Begin

Solution representation

Initialization

Problem

Objective Function Evaluation

Population $P_1$ of candidate solutions with known quality

Reproduction

Population $P_2$ of candidate solutions with known quality

Selection

Terminate?

End

Note:
- Problem related modules
- Algorithm related modules

no

yes
EOA: Core Components

Problem aspects:

- Solution representation
  - Data structures that computer uses to represent solutions
  - Genotype vs. phenotype
  - Batch vs. sequential
  - Individual vs. collective

- Objective function evaluation
  - Known vs. unknown function
  - Time-expensive evaluation

Algorithm aspects:

- Initialization
  - Known vs. unknown search range
  - Prior knowledge
  - Sampling method

- Reproduction
  - Parent selection: sexual vs. asexual
  - Reproduction operators: recombination vs. mutation

- Selection
  - Evolving population over generations (time)
  - Darwinian survival of the fittest principle
  - Selection pressure
EOA: Historical Perspective

- **1930s~1950s: early algorithmic views**
  - Viewing an evolutionary system as an optimization process
  - Viewing an evolutionary system as a complex adaptive controller

- **1960s: three canonical EOAs developed independently**
  - Genetic Algorithm (GA) - Holland, USA, 1962, 1967
  - Evolutionary Strategy (ES) - Rechenberg and Schwefel, Germany, 1965
  - Evolutionary Programming (EP) - Fogel, USA, 1966

- **1970s~1980s: exploitation and exploration**

- **1990s: unified field as evolutionary computation**

- **1990s~2000s: boosting**
  - Swarm intelligence
  - Differential evolution
  - Estimation of distribution
  - Harmonic search
  …
EOA: Categorization

- As per problem properties
  - Discrete vs. continuous vs. mixed vs. structural
  - Constrained vs. unconstrained
  - Single-objective vs. multi-objective
  - Noiseless vs. noisy
  - Static vs. dynamic

- As per algorithmic operators
  - Genetic algorithms
  - Genetic programming
  - Evolutionary strategies/programming
  - Swarm intelligence
  - Estimation of distribution algorithms
  - Differential evolution
  - Artificial immune system
  - Harmonic search
  - Culture algorithm
  - Memetic algorithm
  ...

-11-
Basic concepts
- Chromosome
- Gene, locus and allele
- Genotype and phenotype

A simple GA
- Encoding
- Initialization
- Reproduction
  - Parents selection
  - Crossover (recombination)
  - Mutation
- Selection of survivals

Implicit parallelism explained by schema theory (a specified string of length \( l \) contains \( 2^l \) schema)
011: 011, *11, 0*1, 01*, **1, *1*, 0**, ***

Advanced GA
- Real-parameter GA
- Arithmetic crossover and mutation
- Probabilistic model building GA
- Niching based GA
Evolutionary strategy vs. evolutionary programming

A simple ES
- Initialization
- Reproduction
  - Parents selection
  - Mutation
- Selection of survivals

Mutation strategies
- \((u, v)\) and \((u+v)\)
- Gaussian mutation

Relation to GA
- No need to encoding
- No crossover

Advanced ES
- Crossover added
- Fast evolutionary programming
- Covariance matrix adaptation ES (CMA-ES)
State-of-the-art EOAs

- **Advanced GA**
  - Real-parameter GA
  - Probabilistic model building GA

- **Advanced ES**
  - Fast evolutionary programming
  - Covariance matrix adaptation ES

- **Genetic programming**

- **Swarm intelligence**
  - Particle swarm optimization
  - Ant colony optimization
  - Bee colony optimization
  - Cuckoo search
  - Bat algorithm

- **Differential evolution**

- **Estimation of distribution algorithms**
  - Population-based incremental learning
  - Estimation of multivariate normal algorithm
  - Bayesian optimization algorithm

- **Artificial immune system**

- **Harmonic search**

- **Memetic algorithm**

- **Culture algorithm**

...
EOA: Benchmarking Testbeds

- Benchmark test problems
  - Numerical functions
    - Discrete vs. continuous vs. mixed vs. structural
    - Constrained vs. unconstrained (CEC’06, CEC’10)
    - Single-objective (CEC’05, GECCO’09, GECCO’10) vs. multi-objective (CEC’07, CEC’09)
    - Noiseless vs. noisy (GECCO’09, GECCO’10)
    - Static vs. dynamic (CEC’09)
  - (Simulated) real problems
  - Problem scale (CEC’08, CEC’10)

- Performance measurement
  - Convergence map
  - Average objective function values
  - Success rate with the average number of objective function evaluations
  - Expected running time
  - Bootstrapping based dispersion measure
  - Empirical cumulative distribution
  - Time complexity

- Comparison methods
  - Numerical values
  - Hypothesis testing
EOA: Challenges in Practice

- **Problem formulation**
  - Solution representation
  - Objective function

- **Algorithm comparison**
  - Which performance measures to choose?
  - Whether algorithms are statistically significantly different?
  - How to make comparison conclusions?

- **Algorithm selection**
  - Problem-dependent
  - Requirement-dependent
  - User-dependent
  - Statistical data mining is needed!

- **Parameter and operator selection**
  - Many parameters: are they equally sensitive?
  - Many operators: are they significantly influential?

- **Computational cost**
  - Expensive objective function evaluation
  - Sequential algorithmic structure

**Applications**

**Statistics**

**Parallel Computing**
Part II: Two Advanced EOAs

- Particle swarm optimization (PSO)
- My previous work: comprehensive learning PSO (CLPSO)

- Differential evolution (DE)
- My previous work: self-adaptive DE (SaDE)

Particle swarm optimization (PSO)

- Emulate social behavior of bird flocking or fish schooling to solve optimization problems.
- Each potential solution is represented as a particle.
- Each particle is associated with the objective function value and the velocity.
Notations in PSO

- $x_i = (x_i^1, x_i^2, ..., x_i^D)$ represents the position of the $i^{th}$ particle.
- $v_i = (v_i^1, v_i^2, ..., v_i^D)$ represents the position changing rate (velocity) of the $i^{th}$ particle.
- $pbest_i = (pbest_i^1, pbest_i^2, ..., pbest_i^D)$ represents the best previous position (the position associated with the best objective function value) of the $i^{th}$ particle.
- $gbest = (gbest^1, gbest^2, ..., gbest^D)$ represents the best previous position of the whole swarm.
- $lbest_i = (lbest_i^1, lbest_i^2, ..., lbest_i^D)$ represents the best previous position achieved by those particles within the neighborhood of the $i^{th}$ particle.
Basic PSO formula

- Two basic versions of PSO
  - Global version: each particle learns from its own previous best position (pbest) and the best position found by the whole swarm (gbest).

\[
v_i^d \leftarrow v_i^d + c_1 \times \text{rand1}_i^d \times (\text{pbest}_i^d - x_i^d) + c_2 \times \text{rand2}_i^d \times (\text{gbest}^d - x_i^d)
\]

\[
x_i^d \leftarrow x_i^d + v_i^d
\]

- Local version: each particle learns from its pbest and the best position found by particles within its certain neighborhood (lbest).

\[
v_i^d \leftarrow v_i^d + c_1 \times \text{rand1}_i^d \times (\text{pbest}_i^d - x_i^d) + c_2 \times \text{rand2}_i^d \times (\text{lbest}_i^d - x_i^d)
\]

\[
x_i^d \leftarrow x_i^d + v_i^d
\]

- The random numbers (rand1_i^d and rand2_i^d) should be generated for each dimension of each particle in every iteration.
PSO variants

- Controlling velocities
  - Inertia weight $\omega$
  - Constriction coefficient
  - Time varying acceleration coefficients
  - Linearly decreasing the upper bound of velocity

- Introducing neighborhood topologies
  - Extensive experimental studies
  - Dynamic neighborhood
  - Combine the global version and the local version, named as unified PSO (UPSO)
  - Fully informed PSO (FIPS)

- Hybrid PSO
  - PSO + selection operator
  - PSO + crossover operator
  - PSO + mutation operator
  - PSO + co-operative approach

- Others
  - Binary optimization
  - Constrained optimization
  - Multi-objective optimization
  - Dynamic optimization
Comprehensive learning PSO (CLPSO)

- **Motivation**
  - Premature convergence due to the social learning aspect.
  - Every particle’s \texttt{pbest} may involve some beneficial components deserved to be learn by other particles.

- **Comprehensive learning strategy in CLPSO:**
  \[
  \begin{align*}
  v_i^d &\leftarrow w \times v_i^d + c \times rand_i^d \times \left( p\texttt{best}_{f_i(d)}^d - x_i^d \right) \\
  x_i^d &\leftarrow x_i^d + v_i^d
  \end{align*}
  \]

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

Flow chart: the procedure of obtaining $p_{best}^i$ for the $i^{th}$ particle in the swarm with respect to certain generation
A particle learns from its comprehensive exemplar until stopping to improve for a certain number of generations, called the refreshing gap $m$. After that, the comprehensive exemplar is re-chosen.

CLPSO vs. conventional PSO

- Instead of using particle’s pbest and gbest as the exemplars, all particles’ pbests can be used to guide a particle’s flying direction.

- Instead of learning from the same exemplar for all dimensions, different dimensions of a particle may learn from different exemplars. In other words, at one iteration, different dimensions of a particle may learn from different particle’s pbests at the corresponding dimension.

- Instead of learning from two exemplars (pbest and gbest) in every generation, each dimension of a particle in CLPSO learns from just one comprehensive exemplar.

Search behavior: diversity is much increased!
Experiments

- Parameter sensitivity analysis:
  - Learning probability $P_{C_i}$
    
    $$P_{C_i} = 0.05 + 0.45 \times \left( \frac{\exp\left(\frac{10(i-1)}{ps-1}\right) - 1}{\exp(10) - 1} \right)$$
  - Refreshing gap $m$

- Searching boundary problem
  No re-initialization, not fixed at boundary point, but NOT evaluate!
Experiments

- Performance comparison
  - 9 methods in comparison
    - PSO with inertia weight (PSO-w)
    - PSO with constriction factor (PSO-cf)
    - Local version of PSO with inertia weight (PSO-w-local)
    - Local version of PSO with constriction factor (PSO-cf-local)
    - UPSO
    - Fully informed particle swarm (FIPS)
    - FDR-PSO
    - CPSO-H
    - CLPSO
  - 16 numerical test functions of 10D and 30D in 4 groups with different complexity
  - 30 independent runs for each algorithm
  - Computational budgets: 30,000 FEs for 10D; 200,000 FEs for 30D
  - Performance measures
    - Table: means and stds of objective functions values over 30 runs for each individual algorithm
    - Convergence map: median objective function value within 30 runs vs. FEs
## Experiments

### Table

<table>
<thead>
<tr>
<th>Func</th>
<th>Group C</th>
<th>Group C</th>
<th>Group D</th>
<th>Group D</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
</tr>
<tr>
<td>PSO-w</td>
<td>1.02e+001 ± 3.58e+000</td>
<td>5.69e+002 ± 2.16e+002</td>
<td>1.20e+002 ± 8.94e+001</td>
<td>1.38e+002 ± 1.80e+002</td>
</tr>
<tr>
<td>PSO-cf</td>
<td>1.53e+001 ± 6.38e+000</td>
<td>1.19e+003 ± 4.23e+002</td>
<td>1.60e+002 ± 1.64e+002</td>
<td>2.31e+002 ± 1.93e+002</td>
</tr>
<tr>
<td>PSO-w-local</td>
<td>1.09e+001 ± 4.08e+000</td>
<td>4.72e+002 ± 3.07e+002</td>
<td>4.00e+001 ± 5.98e+001</td>
<td>1.53e+002 ± 1.53e+002</td>
</tr>
<tr>
<td>PSO-cf-local</td>
<td>1.07e+001 ± 2.81e+000</td>
<td>9.09e+002 ± 3.25e+002</td>
<td>9.00e+001 ± 8.52e+001</td>
<td>1.34e+002 ± 1.71e+002</td>
</tr>
<tr>
<td>UPSO</td>
<td>1.47e+001 ± 6.53e+000</td>
<td>1.27e+003 ± 2.29e+002</td>
<td>8.00e+001 ± 8.34e+001</td>
<td>1.79e+002 ± 1.56e+002</td>
</tr>
<tr>
<td>FDR</td>
<td>1.07e+001 ± 3.86e+000</td>
<td>1.07e+003 ± 2.23e+002</td>
<td>1.00e+002 ± 9.73e+001</td>
<td>1.53e+002 ± 2.01e+002</td>
</tr>
<tr>
<td>FIPs</td>
<td>8.84e+000 ± 3.27e+000</td>
<td>2.89e+002 ± 2.00e+002</td>
<td>6.00e+001 ± 5.16e+001</td>
<td>4.21e+001 ± 6.37e+001</td>
</tr>
<tr>
<td>CPSO-H</td>
<td>1.90e+001 ± 9.05e+000</td>
<td>9.67e+002 ± 3.67e+002</td>
<td>1.65e+002 ± 1.42e+002</td>
<td>2.46e+002 ± 2.18e+002</td>
</tr>
<tr>
<td>CLPSO</td>
<td>5.44e+000 ± 1.39e+000</td>
<td>1.14e+002 ± 1.28e+002</td>
<td>1.64e+001 ± 3.63e+001</td>
<td>1.98e+001 ± 2.93e+001</td>
</tr>
</tbody>
</table>

### Diagrams

- **Left Diagram**: Graph showing Best Function Value vs. FEs
- **Right Diagram**: Graph showing Best Function Value vs. FEs
Experiments

General conclusions

- CLPSO shows slower convergence speed on unimodal and simple multimodal problems.
- CLPSO achieves much better optimization performance on multimodal problems, especially on the most complex composition functions.
- CLPSO is more effective in solving problems with less linkage due to the dimension-wise updating rule.
Differential evolution (DE)

- DE, proposed by Price and Storn in 1995, was motivated by an attempt to using Genetic Annealing to solve the Chebychev polynomial fitting problem.

- Genetic annealing is a population-based, combinatorial optimization algorithm.

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

- DE is closely related to many other multi-point derivative free search methods.
DE at a glance

- Characteristics
  - Population-based stochastic direct search
  - Self-referential mutation
  - Simple but powerful
  - Reliable, robust and efficient
  - Easy parallelization
  - Floating-point encoding

- Basic components
  - Initialization
  - Trial vector generation
    - Mutation
    - Recombination
  - Replacement

- Feats
  - DE demonstrated promising performance in 1st and 2nd ICEO
Initialization

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

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

Trial vector generation

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

Differential mutation: with respect to each vector $x_{i,g}$ in the current population, called target vector, a mutant vector $v_{i,g}$ is generated by adding a scaled, randomly sampled, vector difference to a basis vector randomly selected from the current population.
**Insight into classic DE (DE/rand/1/bin)**

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

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

*Example:* at the $g$th generation, the $i$th trial vector $u_{i,g}$ with respect to $i$th target vector $x_{i,g}$ in the current population is generated by:

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

**Replacement**

If the trial vector $u_{i,g}$ has the better objective function value than that of its corresponding target vector $x_{i,g}$, it replaces the target vector in the $(g+1)^{th}$ generation; otherwise the target vector remains in the $(g+1)^{th}$ generation.
Illustration of classic DE
Illustration of classic DE
Illustration of classic DE

Four operating vectors in 2D continuous space
Illustration of classic DE

Mutation
Illustration of classic DE

Crossover

\[ F \cdot (x_{r1,g} - x_{r2,g}) \]
Illustration of classic DE

Replacement

-38-
Differentiation

Intelligent use of differences between individuals resulted in a simple linear operator, so-called differentiation, makes differential evolution unique.

Most important characteristics of DE: self-referential mutation!

Step size is adapted intrinsically along evolution process: large step size at beginning, small step size when converging.

Randomness of search direction and base vector retards convergence.
DE variants

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

**Initialization**

Uniform distribution, Gaussian distribution

**Opposition-based initialization** (Hamid R. Tizhoosh)

\[
OP_{x,0}(j) = b_{j,U} + b_{j,L} - P_{x,0}(j), \quad j = 1, \ldots, D
\]

**Trial vector generation**

- Base vector selection
  - Random selection without replacement: \( r_0 = \text{ceil}(\text{rand} \cdot [0,1] \cdot Np) \)
  - Permutation selection: \( r_0 = \text{permute}[i] \)
  - Random offset selection: \( r_0 = (i + r_g) \% Np \) (e.g. \( r_g = 2 \))
  - Biased selection: global best, local best and better
DE variants

- Differential mutation
  - One difference vector: $F \cdot (x_{r_1} - x_{r_2})$
  - Two difference vector: $F \cdot (x_{r_1} - x_{r_2}) + F \cdot (x_{r_3} - x_{r_4})$
  - Mutation scale factor $F$
    - Crucial role: balance exploration and exploitation
    - Dimension dependence: jitter (rotation variant) and dither (rotation invariant)
    - Randomization: different distributions of $F$

DE/rand/1: $V_{i,G} = X_{r_1,G} + F \cdot (X_{r_2,G} - X_{r_3,G})$

DE/best/1: $V_{i,G} = X_{best,G} + F \cdot (X_{r_1,G} - X_{r_2,G})$

DE/current-to-best/1: $V_{i,G} = X_{i,G} + F \cdot (X_{best,G} - X_{i,G}) + F \cdot (X_{r_1,G} - X_{r_2,G})$

DE/rand/2: $V_{i,G} = X_{r_1,G} + F \cdot (X_{r_2,G} - X_{r_3,G} + X_{r_4,G} - X_{r_5,G})$

DE/best/2: $V_{i,G} = X_{best,G} + F \cdot (X_{r_1,G} - X_{r_2,G} + X_{r_3,G} - X_{r_4,G})$
DE variants

- Recombination
  - Discrete recombination (crossover) (rotation variant)
    - One point and multi-point
    - Exponential
    - Binominal (uniform)
  - Arithmetic recombination
    - Line recombination (rotation invariant)
    - Intermediate recombination (rotation variant)
    - Extended intermediate recombination (rotation variant)
DE variants

- Crossover rate \( CR \in [0,1] \)
  Decomposable (small \( CR \)) and indecomposable functions (large \( CR \))

- Degenerate cases in the trial vector generation
  For example, in classic DE, \( r_1 = r_2, r_0 = r_1, r_0 = r_2, i = r_0, i = r_1, i = r_2 \)
  Better to generate mutually exclusive indices for target vector, base vector and vectors constituting the difference vector

  **Replacement**

  - One-to-one replacement
  - Neighborhood replacement
Motivation for self-adaptation in DE

The performance of DE on different problems mainly depends on:

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

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

- Population size usually depends on the problem scale and complexity
- During evolution, **different strategies** coupled with specific **parameter settings** may favor different search stages
- Replacement schemes influence the population diversity
- Trial and error scheme may take excessive computational time & resources

**Automatically adapt the configuration in DE so as to generate promising trial vectors during evolution**
Related works

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

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

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

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

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

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

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

DE with strategy and parameter self-adaptation

**Strategy adaptation:** select one strategy from a pool of candidate strategies with the probability proportional to its previous successful experience to generate promising trial vectors within a certain learning period

**Steps:**

- Initialize selection probability $p_i=1/\text{num}_{\text{st}}, i=1,...,\text{num}_{\text{st}}$ for every strategy
- According to the current probabilities, we employ *stochastic universal selection* to assign one strategy to each target vector in the current population
- For each strategy, we define $n_{si,g}$ and $n_{fi,g}, i=1,...,\text{num}_{\text{st}}$ to store the number of trial vectors successfully entering the next generation or discarded by applying this strategy, respectively, at the generation $g$. 
Success and failure memories are created to store those numbers within a specified number of previous generations, called “learning period (LP)”. When the memories overflow, the first record of the earliest generation will be removed and the latest record will enter the memory.

The selection probability $p_i$ is updated by:

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

$$S_{k,G} = \frac{\sum_{g=G-LP}^{G-1} n_{s_{k,g}}}{\sum_{g=G-LP}^{G-1} n_{s_{k,g}} + \sum_{g=G-LP}^{G-1} n_{f_{k,g}}} + \varepsilon \quad (k=1,2,...,num\_st; G>LP)$$
Self-adaptive DE (SaDE)

- 4 strategies involved in the candidate pool:

DE/rand/1/bin:
\[ V_{i,G} = X_{i,G} + F \cdot (X_{r_2,G} - X_{r_3,G}) \]

DE/rand/2/bin:
\[ V_{i,G} = X_{r_1,G} + F \cdot (X_{r_2,G} - X_{r_3,G} + X_{r_4,G} - X_{r_5,G}) \]

DE/current-to-best/2/bin:
\[ V_{i,G} = X_{i,G} + F \cdot (X_{\text{best}G} - X_{i,G}) + F \cdot (X_{r_1,G} - X_{r_2,G} + X_{r_3,G} - X_{r_4,G}) \]

DE/current-to-rand/1:
\[ V_{i,G} = X_{i,G} + \text{rand}(0,1) \cdot (X_{r_1,G} - X_{i,G}) + F \cdot (X_{r_2,G} - X_{r_3,G}) \]
Self-adaptive DE (SaDE)

Parameter adaptation

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

Crossover rate ($CR_j$): when applying strategy $j$ to a target vector, the corresponding $CR_j$ value is generated according to an assumed distribution. We hereby assume that each $CR_j$, $j=1,...,num_st$ is normally distributed with its mean and standard deviation initialized to 0.5 and 0.1, respectively. Those $CR_j$ values that had generated trial vectors successfully entering the next generation over previous generations are stored in a memory of size $LP$. The mean of $CR_j$ normal distribution is updated at every generation after initial $LP$ generations by using the median value in the memory.
Experiments

- Parameter sensitivity analysis
  - Learning period (LP)
    - 5 different LP values
      - (20, 30, 40, 50, and 60)
    - Not Sensitive

- Analysis of self-adaptation property
  - Self-adaptation of crossover probability
Experiments

- Self-adaptation of trial vector generation strategy

Griewank

Rotated Griewank
Experiments

- **Performance comparison**
  - 10 methods in comparison
    - DE/rand/1/bin, F=0.9, CR=0.1
    - DE/rand/1/bin, F=0.9, CR=0.9
    - DE/rand/1/bin, F=0.5, CR=0.3
    - DE/rand-to-best/1/bin F=0.5, CR=0.3
    - DE/rand-to-best/2/ bin with F=0.5, CR=0.3
    - Adaptive DE
    - SDE
    - jDE
    - FADE
    - SaDE
  - 26 numerical test functions: \( f_1 \sim f_{14} \) (10D and 30D), \( f_{15} \sim f_{26} \) (specified)
  - 30 independent runs for each algorithm
  - Computational budgets: \( f_1 \sim f_{14} \) (10D: 100,000 FEs; 30D: 300,000 FEs); \( f_{15} \sim f_{26} \) (500,000 FEs)

- **Performance measures**
  - Table: means & stds of objective functions values as well as the successful rate for \( f_1 \sim f_{14} \), means of FEs for \( f_{15} \sim f_{26} \) over 30 runs
  - Convergence map: median objective function value within 30 runs vs. FEs
  - Empirical distribution of normalized success performance
Experiments

- General conclusions
  - In comparison with conventional DE, SaDE achieves smaller mean function values and higher success rates for all problems, while the convergence speed is faster for most problems.
  - In comparison with other adaptive DE variants, SaDE compares favorably with others.
  - Overall comparison: SaDE outperforms other algorithms in terms of empirical distribution of normalized success performance.
Part III: Research Plan Discussion

- Statistical comparison and interactive selection of EOAs
- Development of novel EOAs
- Applications
  - Computer vision and pattern recognition
  - Statistical modeling
  - Bioinformatics
  ...
Statistical Comparison of EOAs

- **Motivations**
  - Many algorithms
  - Many operators
  - Many parameters

- **Objectives:** statistical comparison, grouping and ranking of EOAs

- **Challenges**
  - Problem dependent
  - Choose suitable performance measures
  - Choose statistically sound comparison methods

- **Pre-requisites**
  - Survey of existing benchmarking test problems
  - Survey of existing performance measures
  - Survey of existing comparison method

- **Research directions**
  - Survey of existing benchmarking testbeds and comparison methods
  - Investigation of inferential statistical tests
    - Symmetric hypothesis test
    - Random process
    - Factorial analysis of variance
  - Statistical comparison, grouping and ranking state-of-the-art PSO and DE
Interactive Selection of EOAs

- **Motivations**
  - Problem dependent
  - Requirement dependent
  - User dependent

- **Objectives:** user-friendly interactive selection software

- **Challenges**
  - Less known problem properties in practice

- **Pre-requisite**
  - Statistical comparison, grouping and ranking of EOAs

- **Research plans**
  - Iterative selection pipeline
  - Test on real applications
  - Online selection website
Development of Novel EOAs

- Novel ensemble EOAs framework
- GMM, extreme value theory and copula theory with EOAs
- High-dimensional EOAs
- Common PC based parallel EOAs
Applications

- Computer vision and pattern recognition
- Statistical modeling
  - Any helps on model parameter estimation?
- Bioinformatics
  - Gene regulatory network model inference?
  - Gene selection?
  - Any others?
- Other potential applications
Thanks