Variable Interaction Learning in Cooperative Coevolution

Wenxiang Chen

Nature Inspired Computation and Applications Laboratory (NICAL)
School of Computer Science and Technology
University of Science and Technology of China (USTC)

August 14, 2010

Outline

Warming-Up: A Fabricated Problem

Problem

Suppose that we are faced with:

- a fully-decomposable problem
- its dimension is D
- domains for each dimension are the same, say $[1, \ldots, N]$

Significantly Reduce The Hardness of Optimizing Decomposable Problems by . . .

Divide-and-Conquer

- For conventional Approach:
  - The size of search space grows exponentially with D increasing
  - Comp(N) $= N^D$
- For Divide-and-Conquer Approach:
  - The size of search space grows linearly with D increasing
  - Comp(N) $= N \times D$

What If No Prior Informations Are Given?

Black-Box optimization

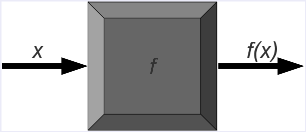


Figure: Illustration of BlackBox

What If No Prior Informations Are Given?

Black-Box optimization

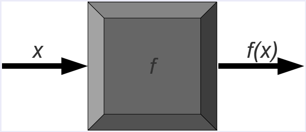


Figure: Illustration of BlackBox

Guess by Learning Interdependence

In order to apply Divide-and-Conquer approach in such case, we need the Interdependence Learning, which:

- reduces the complexity of optimization
- doesn't affect the quality of solution greatly

Interdependency from three different but related views

Interdependence in Nature and EC

Interdependence		
Biology	Evolutionary Computation	
	Binary Representation	Real-Value Representation
Epistasis[6][4]	Linkage[5][3][2]	Variable Interaction[1][11]

NOTE:

- Numerous studies have been devoted to study on Linkage.
- The red part is what we are focus on.

What is CC?

Cooperative Coevolution[7][9][8]

- divide the decision vector into groups of variables
- evolve each sub-population by groups
- combine the best representatives from all other dimensions to compose the vector for fitness evaluation

Why is CC?

Problem Domain

We focus on tackling Large-Scale Numerical Optimization(LNGO), in which:

- landscapes of fitness functions become more complex
- search space and computational effort grow exponentially

Why is CC?

Problem Domain

We focus on tackling Large-Scale Numerical Optimization(LNGO), in which:

- landscapes of fitness functions become more complex
- search space and computational effort grow exponentially

Promising Solution:Cooperative Coevolution

- reduce the computational complexity significantly
- speed up the convergence of the search procedure
- easy to maintain the quality for highly separable problems

Drawbacks of the Current CC-based Algorithms

Drawbacks

- do not consider the separability
- easy to trap in local optimum
- seriously affects the quality of the solution

which make it unpractical to apply CC-based algorithms in the complex, non-separable problems.

A Way Out: Learning the Separability of Problem

Assumption

We start by assuming all pair of variables are independent.

A Way Out: Learning the Separability of Problem

Assumption

We start by assuming all pair of variables are independent.

Our Target

1. provide a fine-grain learning mechanism on the separability
2. every variable interaction learned by the mechanism should be correct
3. make use of the separability information gathered by learning

Theoretical Base For Our Learning Mechanism

Definition

A function is separable, if it satisfies the Equation 1. [10]

$$\arg\min_{(x_1,\ldots,x_N)} f(x_1,\ldots,x_N) = \left(\arg\min_{(x_1)} f(x_1,\cdots),\ldots,\arg\min_{(x_N)} f(\cdots,x_N)\right) \tag{1}$$

A separable function can be optimized dimension by dimension.

Theoretical Base For Our Learning Mechanism

Definition 1

Two decision variables i and j are interacting if there is a decision vector $\vec{x}$ whose i^{th} and j^{th} variable can be substituted with values x_i' and x_j' so that Equation holds.[11]

$$\exists \vec{x}, x_i', x_j': \quad f(x_1,\ldots, x_i, \quad, x_j, \ldots, x_n) < f(x_1,\ldots, x_i', \quad, x_j, \ldots, x_n) \wedge$$
$$f(x_1,\ldots, x_i, \quad, x_j', \ldots, x_n) > f(x_1,\ldots, x_i', \quad, x_j', \ldots, x_n)$$

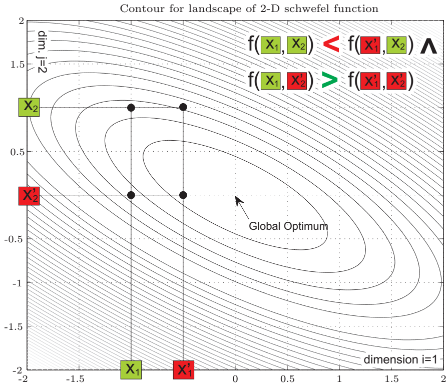
Any interacting variables should be put together in the same sub-population.

Example of Variable Interaction Learning

Example

As shown in Figure. 5,

1. suppose we start with (x_1, x_2) and (x_1', x_2), the Global Optimum is located in (0,0), and it is a minimization problem

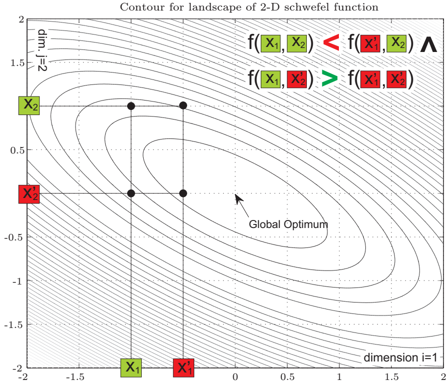


Example of Variable Interaction Learning

Example

As shown in Figure. 5,

1. suppose we start with (x_1, x_2) and (x_1', x_2), the Global Optimum is located in (0,0), and it is a minimization problem
2. $f(x_1, x_2) \leq f(x_1', x_2)$

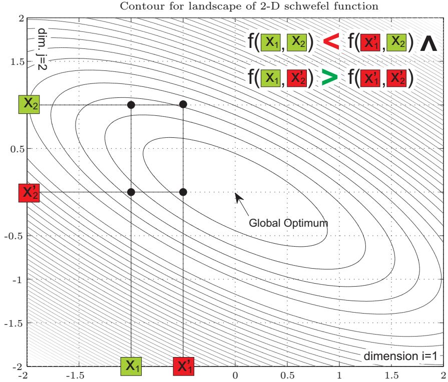


Example of Variable Interaction Learning

Example

As shown in Figure. 5,

1. suppose we start with (x_1, x_2) and (x_1', x_2), the Global Optimum is located in (0,0), and it is a minimization problem
2. $f(x_1, x_2) \leq f(x_1', x_2)$
3. move both the point along one axis towards the global optimum

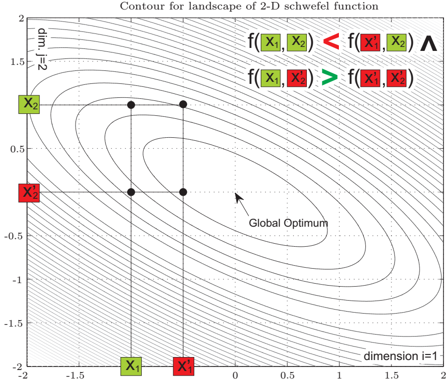


Example of Variable Interaction Learning

Example

As shown in Figure. 5,

1. suppose we start with (x_1, x_2) and (x_1', x_2), the Global Optimum is located in (0,0), and it is a minimization problem
2. $f(x_1, x_2) \leq f(x_1', x_2)$
3. move both the point along one axis towards the global optimum
4. the relation changes, $f(x_1, x_2') \geq f(x_1', x_2')$

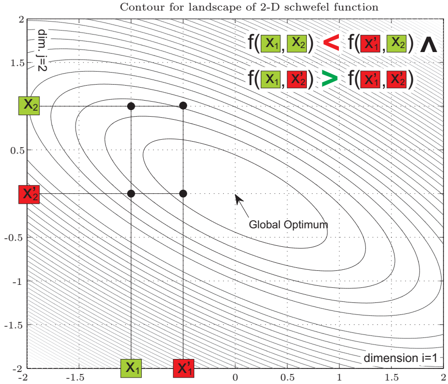


Example of Variable Interaction Learning

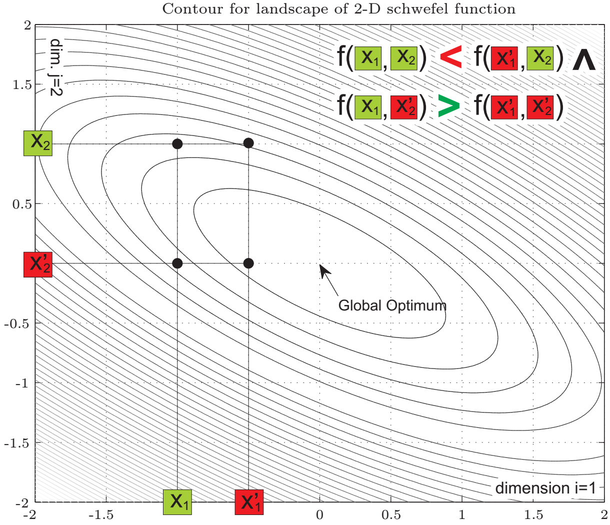
Example

As shown in Figure. 5,

1. suppose we start with (x_1, x_2) and (x_1', x_2), the Global Optimum is located in (0,0), and it is a minimization problem
2. $f(x_1, x_2) \le f(x_1', x_2)$
3. move both the point along one axis towards the global optimum
4. the relation changes, $f(x_1, x_2') \ge f(x_1', x_2')$
5. Intuitively, there are variable interactions in 2-D Schewfel Function



Example of Variable Interaction Learning (Zoom In)

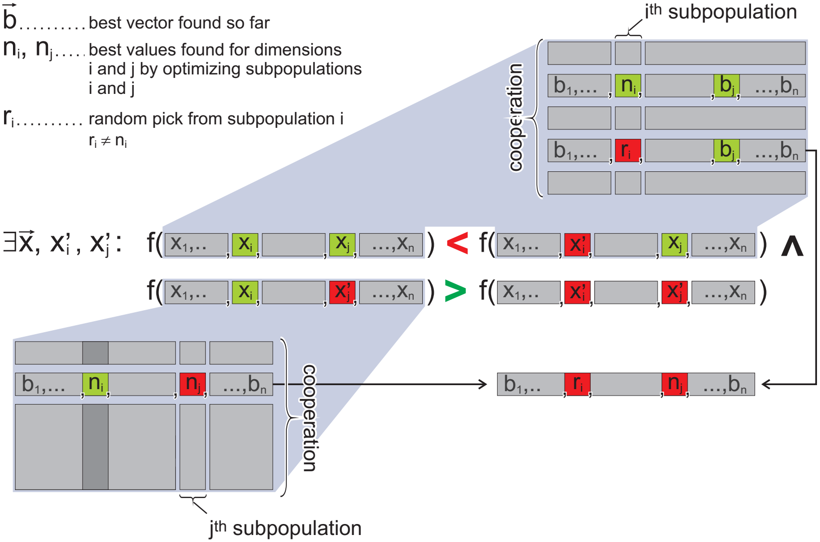


How Does Learning Mechanism Works?

$\vec{b}$ .... best vector found so far

n_i, n_j ..... best values found for dimensions i and j by optimizing subpopulations i and j

r_i .... random pick from subpopulation i
$r_i \neq n_i$



Our Contribution: Avoid Type II Error Completely

In The Work by Weicker [11]

- examine dimension i and dimension j, which are selected totally randomly
- it can violates the condition of Definition 1
- may lead to detection of non-existing variable interaction

Our Contribution: Avoid Type II Error Completely

In The Work by Weicker [11]

- examine dimension i and dimension j, which are selected totally randomly
- it can violates the condition of Definition 1
- may lead to detection of non-existing variable interaction

Our Contribution

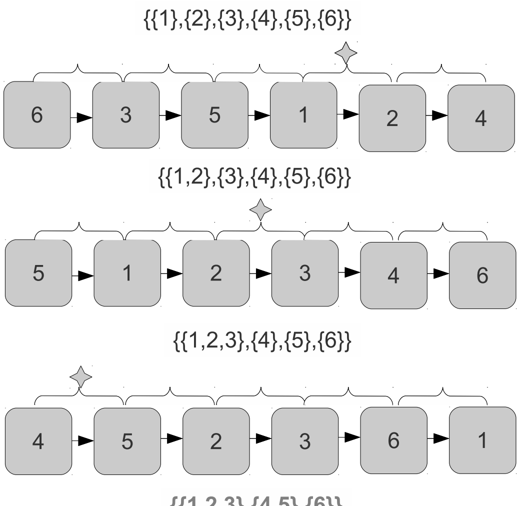
- only tests the currently and the previously optimized dimension for interaction
- The condition of Definition 1 can never be violated and only becomes true for real interactions
- the flaw of detecting non-existent interactions(the type II error) is avoided

CCVIL: A Two Stage Approaches

1. Learning Stage
    - population size $= 3$, generation limit for each subcomponent $= 1 \rightarrow$ reduce the overhead of learning
    - population re-initialization in each cycle $\rightarrow$ avoid possible premature convergence
    - the sequence of the dimensions to optimize is randomly permuted in each cycle $\rightarrow$ every pair of variables shares equal chance to examine interaction
    - store the learned interaction knowledge in *groupInfo*
2. Optimization Stage
    - apply a certain EA as the sub-optimizer
        - in my case, it is JADE [14]
    - evolve the sub-population in CC with respect to *groupInfo*

Example of Learning Procedure in CCVIL

Real Interaction: $\{\{1, 2, 3\}, \{4, 5\}, \{6\}\}$



How Learning Works: Efficiency of Learning

Assume that we are faced with an N-dimension problem:

1. probability of placing dimensions i and j adjacently in one random permutation is $2/N$
2. probability $p_{capt}(K)$ that 1 happens in at least once in K learning cycles then is given in Equation 2.

$$p_{capt}(K) = 1 - (1 - 2/N)^K \tag{2}$$

3. $p_{capt}(500)$=0.6325, $p_{capt}(800)$=0.7984
   - from learning perspective, it is always better to have more cycles

No Free Lunch: The Learning Overhead

Appropriate setting for learning cycle can deal with both separable functions and non-separable functions:

For separable functions

- set up a lower bound $\check{K}$ for cycle of learning
  - if no interactions were learned by then, we treat it as separable function, and switch to optimization stage at once

No Free Lunch: The Learning Overhead

Appropriate setting for learning cycle can deal with both separable functions and non-separable functions:

For separable functions

- set up a lower bound $\check{K}$ for cycle of learning
  - if no interactions were learned by then, we treat it as separable function, and switch to optimization stage at once

For non-separable functions

If any interaction has been learned before reaching the $\check{K}$ cycles, we treat it as a non-separable function. In this case, learning stage only stops if:

- all N dimensions have been combined into one group
- 60% of fitness evaluation has been consumed in learning stage

Benchmark Function

From CEC'10 Speicial Session of large-scale global optimization [10]. The major features of them can be concluded as follow:

	Function	Sep	Multi modal	Groups real	Groups found
f_1	Shifted Elliptic Function	Yes	No	1000	1000
f_2	Shifted Rastrigin's Function	Yes	Yes	1000	1000
f_3	Shifted Ackley's Function	Yes	Yes	1000	969
f_4	Single-group Shifted 50-rotated Elliptic Function	No	No	951	963
f_5	Single-group Shifted 50-rotated Rastrigin's Function	No	Yes	951	952
f_6	Single-group Shifted 50-rotated Ackley's Function	No	Yes	951	921
f_7	Single-group Shifted 50-dimensional Schwefel's	No	No	951	952
f_8	Single-group Shifted 50-dimensional Rosenbrock's	No	Yes	951	1000
f_9	10-group Shifted 50-rotated Elliptic Function	No	No	510	627
f_{10}	10-group Shifted 50-rotated Rastrigin Function	No	Yes	510	516
f_{11}	10-group Shifted 50-rotated Ackley Function	No	Yes	510	501
f_{12}	10-group Shifted 50-dimensional Schwefel's	No	No	510	522
f_{13}	10-group Shifted 50-dimensional Rosenbrock's	No	Yes	510	1000
f_{14}	20-group Shifted 50-rotated Elliptic Function	No	No	20	232
f_{15}	20-group Shifted 50-rotated Rastrigin's Function	No	Yes	20	37
f_{16}	20-group Shifted 50-rotated Ackley Function	No	Yes	20	39
f_{17}	20-group Shifted 50-rotated Schwefel's Function	No	No	20	42
f_{18}	20-group Shifted 50-rotated Rosenbrock's Function	No	Yes	20	1000
f_{19}	Shifted Schwefel's Function 1.2	No	No	1	1
f_{20}	Shifted Rosenbrock's Function	No	Yes	1	1000

Experimental Result

TABLE: Comparison with other CC-based algorithms and plain JADE.

	CCVIL		DECC-G		MLCC		R_1	Naive JADE		R_2
	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev		Mean	Std Dev	
f_1	1.55e-17	7.75e-17	2.93e-07	8.62e-08	1.53e-27	7.66e-27	–	1.57e+04	1.38e+04	W
f_2	6.71e-09	2.31e-08	1.31e+03	3.24e+01	5.55e-01	2.20e+00	W	7.66e+03	9.67e+01	W
f_3	7.52e-11	6.58e-11	1.39e+00	9.59e-02	9.86e-13	3.69e-12	L	4.52e+00	2.41e-01	W
f_4	9.62e+12	3.43e+12	5.00e+12	3.38e+12	1.70e+13	5.38e+12	W	6.14e+09	3.81e+09	L
f_5	1.76e+08	6.47e+07	2.63e+08	8.44e+07	3.84e+08	6.93e+07	W	1.35e+08	1.21e+07	L
f_6	2.94e+05	6.09e+05	4.96e+06	8.02e+05	1.62e+07	4.97e+06	W	1.94e+01	1.79e-02	–
f_7	8.00e+08	2.48e+09	1.63e+08	1.38e+08	6.89e+05	7.36e+05	–	2.99e+01	3.30e+01	–
f_8	6.50e+07	3.07e+07	6.44e+07	2.89e+07	4.38e+07	3.45e+07	–	1.19e+04	4.92e+03	L
f_9	6.66e+07	1.60e+07	3.21e+08	3.39e+07	1.23e+08	1.33e+07	W	2.70e+07	2.08e+06	L
f_{10}	1.28e+03	7.95e+01	1.06e+04	2.93e+02	3.43e+03	8.72e+02	W	8.50e+03	2.30e+02	W
f_{11}	3.48e+00	1.91e+00	2.34e+01	1.79e+00	1.98e+02	6.45e-01	W	9.29e+01	9.66e+00	W
f_{12}	8.95e+03	5.39e+03	8.93e+04	6.90e+03	3.48e+04	4.91e+03	W	6.21e+03	1.34e+03	–
f_{13}	5.72e+02	2.55e+02	5.12e+03	3.95e+03	2.08e+03	7.26e+02	W	1.87e+03	1.11e+03	W
f_{14}	1.74e+08	2.68e+07	8.08e+08	6.06e+07	3.16e+08	2.78e+07	W	1.00e+08	8.84e+06	L
f_{15}	2.65e+03	9.34e+01	1.22e+04	9.10e+02	7.10e+03	1.34e+03	W	3.65e+03	1.09e+03	W
f_{16}	7.18e+00	2.23e+00	7.66e+01	8.14e+00	3.77e+02	4.71e+01	W	2.09e+02	2.01e+01	W
f_{17}	2.13e+04	9.16e+03	2.87e+05	1.97e+04	1.59e+05	1.43e+04	W	7.78e+04	5.87e+03	W
f_{18}	1.33e+04	1.00e+04	2.46e+04	1.05e+04	7.09e+03	4.77e+03	–	3.71e+03	9.58e+02	L
f_{19}	3.52e+05	2.04e+04	1.11e+06	5.00e+04	1.36e+06	7.31e+04	W	3.48e+05	1.67e+04	–
f_{20}	1.11e+03	3.04e+02	4.06e+03	3.66e+02	2.05e+03	1.79e+02	W	2.06e+03	2.01e+02	W

Speed Up Learning Stage

Weaknesses of Current Learning Algorithm

For a N-dimension function, the overhead for checking each pair of variables once is $\frac{N(N-1)}{2}$.

- it is hard to decide when to switch stage
- even given sufficienctly long time for learning stage, still can't ensure learning all existing interaction

accelerate the learning stage by introducing heuristic approximation [2]

Generalize The Benchmark Function

Drawbacks of Benchmarks from CEC'10 LSGO

- only treat existence of interaction as true or false
- do not distinguish the strength of interaction

Consider the Strength of Interaction

- introduce the concept of strength of interaction in the benchmark
- characterize the strength of interaction as a real value between 0 to 1
- design an new algorithm that can tackle the new set of benchmark functions

Wenxiang Chen, Thomas Weise, Zhenyu Yang, and Ke Tang.
Large-scale global optimization using cooperative coevolution with variable interaction learning.
In Robert Schaefer, Carlos Cotta, Joanna Kolodziej, and Günter Rudolph, editors, *Parallel Problem Solving from Nature, PPSN XI*, LNCS 6239, pages 300–309. Springer-Verlag Berlin Heidelberg, 2010.

Georges Harik.
Linkage learning via probabilistic modeling in the ECGA.
Technical report, Illinois Genetic Algorithms Laboratory, 1999.

Yuan-Wei Huang and Ying-ping Chen.
On the detection of general problem structures by using inductive linkage identification.
In *GECCO '09: Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 1853–1854, New York, NY, USA, 2009. ACM.

B. Naudts, A. Verschoren, and B. Antwerp.
Epistasis on finite and infinite spaces.

In *8th Int. Conf. on Systems Research, Informatics and Cybernetics*, pages 19–23, 1996.

Martin Pelikan, David E. Goldberg, and Erick Cantú-Paz.
Linkage problem, distribution estimation, and Bayesian networks.
Evolutionary Computation, 8(3):340, 2000.

Patrick C. Phillips.
The language of gene interaction.
Genetics, 149(3):1167, 1998.

Mitchell A. Potter.
The Design and Analysis of a Computational Model of Cooperative Coevolution.
PhD thesis, George Mason University, 1997.

Mitchell A. Potter and Kenneth Alan De Jong.
A cooperative coevolutionary approach to function optimization.
In *3rd Conf. on Parallel Problem Solving from Nature*, volume 2, pages 249–257, 1994.

Mitchell A. Potter and Kenneth Alan De Jong.
Cooperative coevolution: architecture for evolving coadapted subcomponents.
Evolutionary Computation, 8(1):1–29, 2000.

Ke Tang, Xiaodong Li, Ponnuthurai Nagaratnam Suganthan, Zhenyu Yang, and Thomas Weise.
Benchmark functions for the CEC'2010 special session and competition on large scale global optimization.
TR, NICAL, USTC, Hefei, Anhui, China, 2009.
http://nical.ustc.edu.cn/cec10ss.php.

Karsten Weicker and Nicole Weicker.
On the improvement of coevolutionary optimizers by learning variable interdependencies.
In *Proc. 1999 Congress on Evolutionary Computation (CEC'99). IEEE Press*, pages 1627–1632, 1999.

Zhenyu Yang, Ke Tang, and Xin Yao.
Large scale evolutionary optimization using cooperative coevolution.

Information Sciences, 178(15):2985–2999, 2008.

Zhenyu Yang, Ke Tang, and Xin Yao.
Multilevel cooperative coevolution for large scale optimization.
In *IEEE Congress on Evolutionary Computation*, pages 1663–1670. IEEE Press, 2008.

Jingqiao Zhang and Arthur C. Sanderson.
JADE: adaptive differential evolution with optional external archive.
IEEE Transactions on Evolutionary Computation, 13(5):945–958, 2009.

Algorithm 1: *groupInfo* $\longleftarrow$ Learning Stage of CCVIL

$K \longleftarrow 0$;
groupInfo $\longleftarrow \{\{1\}, \{2\}, \ldots, \{N\}\}$ `// initially assume full separability`
repeat `// start a new learning cycle`
  $\Pi \longleftarrow$ random permutation of dimension indices $\{1, 2, \ldots, N\}$;
  $K \longleftarrow K + 1$;
  lastIndex $\longleftarrow 0$;
  $(subpop, pop, \vec{best}) \longleftarrow$ **initializeCC**$(3, 3, \ldots, 3)$ `// use` $NP_i = 3 \; \forall i \in 1..N$
  for $i = 1$ **to** N **do** `// start a new learning phase, i.e., tackle next dimension`
    if *lastIndex* $\neq 0$ **then**
      $G_1 \longleftarrow$ **find**$(groupInfo, \Pi_i)$ `// find the group containing` Π_i
      $G_2 \longleftarrow$ **find**$(groupInfo, lastIndex)$ `// find group of last optimized variable`
    if $i = 1 \vee (G_1 \neq G_2)$ **then**
      for $j = 1$ **to** NP **do**
        $popcc_j \longleftarrow (best_1, \ldots, best_{\Pi_i - 1}, subpop_{\Pi_i, j}, best_{\Pi_i + 1}, \ldots)$
      $(popcc, \vec{new}) \longleftarrow$ **optimizer**$(popcc, \Pi_i)$ `// any optimizer, we used JADE`
      $subpop_{\Pi_i} \longleftarrow popcc_{\Pi_i}$;
      $best_{\Pi_i} \leftarrow new_{\Pi_i}$;
      if *lastIndex* $\neq 0$ **then**
        Compose $\vec{x}$ and $\vec{x'}$ according to Equations 3 and 4;
        if $f(\vec{x}) < f(\vec{x'})$ **then** `// interaction between dim.` i `and` *lastIndex*`?`
          groupInfo $\longleftarrow ((groupInfo \setminus \{G_1\}) \setminus \{G_2\}) \cup (\{G_1 \cup G_2\})$;
      lastIndex $\longleftarrow \Pi_i$ `// only test successively optimized dimensions`
until $(|groupInfo| = 1) \vee [(K > \check{K}) \wedge (|groupInfo| = N)] \vee (K > \hat{K})$;
return *groupInfo* `// return the set of mutually separable groups of interacting variables`

Experimental Setup

Parameter Configuration

The dimension of problem is 1000.

- generation limit of optimizing a certain subcomponent: $Gen_i = \min\{|G_i| + 5, 500\}$
- population size of optimizing a certain subcomponent: $NP_i = |G_i| + 10$

Comparison

We compare the performance of DECC-G[12] MLCC[13] and JADE[14] on the benchmark, with the parameter setting recommended by the original literature.

- DECC-G and MLCC is state-of-the-art CC-based algorithms.
- JADE is the sub-optimizer applied in our algorithm. The population size of JADE is set to 1000 for comparison.

How Does Learning Mechanism Works?

The Criterion for Judging Variable Interaction

- $\vec{best}$ is the vector of the best values for each decision variable discovered so far
- after optimizing on dimension i, $\vec{best}_i$ is updated by $\vec{new}_i$
- $\vec{rand}_i$ is a random candidate from global (with $\vec{new} \neq \vec{rand} \neq \vec{best}$)

$$x_j = \begin{cases} new_i & \text{if } j = i \\ best_j & \text{otherwise} \end{cases} \quad (3) \qquad x'_j = \begin{cases} new_i & \text{if } j = i \\ rand_k & \text{if } j = k \\ best_j & \text{otherwise} \end{cases} \quad (4)$$

If $f(\vec{x}') < f(\vec{x})$, there is likely an interaction between dimension i and k. [11]

However, there are flaws in the method from [11], which lead it to commit type II error in some cases.