Decomposition and Cooperative Coevolution Techniques for Large Scale Global Optimization

Xiaodong Li

School of Computer Science and Information Technology
RMIT University, Melbourne, Australia
Email: xiaodong.li@rmit.edu.au

CEC’2015 - 25 May 2015
Outline

1. Large Scale Global Optimization
2. Cooperative Coevolution
3. Decomposition Methods with CC
4. Contribution Based Cooperative Co-evolution (CBCC)
5. Differential Grouping
6. CEC’2013 LSGO Benchmark Test Functions
7. Route Distance Grouping for Capacitated Arc Routing Problems
8. Conclusions
Outline

1. Large Scale Global Optimization
2. Cooperative Coevolution
3. Decomposition Methods with CC
4. Contribution Based Cooperative Co-evolution (CBCC)
5. Differential Grouping
6. CEC’2013 LSGO Benchmark Test Functions
7. Route Distance Grouping for Capacitated Arc Routing Problems
8. Conclusions
Large Scale Global Optimization

**LSGO problem**

\[
\min_{\vec{x} \in \mathbb{R}^n} f(\vec{x}) \tag{1}
\]

where \( f : \mathbb{R}^n \rightarrow \mathbb{R} \) a real-valued objective function, and \( n \) (the number of variables) is large, eg., from several hundreds to thousands. Equation (1) assumes minimization.
Existing meta-heuristic methods are ill-equipped in dealing with LSGO problems, though they may be effective in solving small to medium sized problems.

LSGO problems can be found in many application areas, eg., engineering, computational genetics, natural language processing.

In this research, we focus on single objective, continuous, unconstrained, and black-box LSGO problems.

For LSGO problems with constraints, Augmented Lagrangian methods can be used to transform the original problem into its Lagrangian dual, which is unconstrained.
EAs for LSGO

- Many Evolutionary Algorithms have been developed for global optimization.
- **Curse of dimensionality** - The search space grows exponentially. The performance of EAs deteriorates as the number of variables (dimensions) increases.
- Traditional EAs do not scale well as the dimensionality of the problem increases.
- New techniques are required with better scalability to higher dimensions.
Outline

1. Large Scale Global Optimization
2. Cooperative Coevolution
3. Decomposition Methods with CC
4. Contribution Based Cooperative Co-evolution (CBCC)
5. Differential Grouping
6. CEC’2013 LSGO Benchmark Test Functions
7. Route Distance Grouping for Capacitated Arc Routing Problems
8. Conclusions
Cooperative coevolutionary framework

- First Cooperative Coevolutionary (CC) model was proposed by [Potter and De Jong 1994].
- A “divide-and-conquer” method to decompose a problem into several smaller subcomponents, each of which is evolved using a separate EA.
- Specifically, a $n$-dimensional decision vector is divided into $n$ 1-dimensional subcomponents each of which is optimized using a separate GA in a round-robin fashion.
- CC has been used with various EAs, eg., Evolutionary Programming, Evolutionary Strategies, Particle Swarm Optimization, and Differential Evolution.
Divide-and-conquer

- A large problem can be subdivided into smaller and simpler problems.
- Dates back to René Descartes (*Discourse on Method*).
- Has been widely used in many areas:
  - Computer Science: Sorting algorithms (quick sort, merge sort)
  - Engineering: Discrete Fourier transform (FFTs)
  - Optimization: Large-scale linear programs (Dantzig)
  - Politics: Divide and rule (In *Perpetual Peace* by Immanuel Kant: *Divide et impera* is the third political maxims.)

Individual $x_{1c}$ is assigned with a fitness value by evaluating an $n$-dim vector $(x_{1c}, x_{2b}, \ldots, x_{nb})$, which consists of $x_{1c}$ and the best-fit individuals from all the remaining subcomponents.
Cooperative coevolutionary framework

CC evolutionary algorithms’ performance degrades when applied to non-separable problems. In an ideal setting the interacting variables should be grouped in one subcomponent in order to enhance the performance.

Questions:

- How to group interacting variables into the same subcomponents, so that the inter-dependency between subcomponents is kept at minimum? Can this be learnt?
- How to determine the suitable subcomponent sizes (which may be unequal)?
- What would be a good and competent optimizer for a subcomponent?
Separability and non-separability

**Definition**

A function $f(x_1, \ldots, x_n)$ is separable iff:

\[
\text{arg min}_{(x_1, \ldots, x_n)} f(x_1, \ldots, x_n) = \left( \text{arg min}_{x_1} f(x_1, \ldots), \ldots, \text{arg min}_{x_n} f(\ldots, x_n) \right), \tag{2}
\]

and non-separable otherwise (assuming minimization).

In other words, if it is possible to find the global optimum of a function by optimizing one dimension at a time independently from other dimensions, then the function is said to be separable (otherwise non-separable) [Auger, et al. 2007].
Non-separability (epistasis)

- Non-separability means variable interaction (or linkage, epistasis).

  \[ f(x, y) = x^2 + \lambda_1 y^2 \]
  \[ g(x, y) = x^2 + \lambda_1 y^2 + \lambda_2 xy \]

- An optimization algorithm may perform poorly because the inter-dependencies among different variables could not be captured well enough by the algorithm.

Partially (or additively) Separable Functions

\[ f(\vec{x}) = \sum_{i=1}^{m} f_i(\vec{x}_i) \]  \hspace{1cm} (3)

where \( \vec{x}_i \) are mutually exclusive decision vectors of \( f_i \), and \( \vec{x} = \langle x_1, \ldots, x_n \rangle \) is the global decision vector of \( n \) dimensions and \( m \) is the number of independent subcomponents in the global objective function \( f \). This information can be exploited.
Observation

The fitness landscape of a separable function can be rotated to produce a non-separable function, with only the orientation of the landscape being changed.
Identifying Interacting Variables

- **Binary-Coded EAs:**
  - LLGA (Linkage Learning GA) [Harik et. al. 1996];
  - LINC (Linkage Identification by Nonlinearity Check) [Munetomo and Goldberg 1999];
  - BOA (Bayesian Optimization Algorithm) [Pelikan et. al. 1999];
  - ...

- **Real-coded EAs:**
  - LINC-R (Linkage Identification by Nonlinearity Check for Real-Coded GAs) [Tezuka, et al. 2004].
  - Used for LSGO:
    - FEPCC - Fast Evolutionary Programming with CC [Liu, et al. 2001];
    - Random grouping [Yang, et al. 2008];
    - Delta grouping [Omidvar, et al. 2010a];
Outline

1. Large Scale Global Optimization
2. Cooperative Coevolution
3. Decomposition Methods with CC
4. Contribution Based Cooperative Co-evolution (CBCC)
5. Differential Grouping
6. CEC’2013 LSGO Benchmark Test Functions
7. Route Distance Grouping for Capacitated Arc Routing Problems
8. Conclusions
### Classes of variable grouping methods

#### Fixed grouping
Variable grouping is fixed throughout the optimization run, including methods such as the original CC method [Potter and De Jong 1994], FEPCC [Liu, et al. 2001], Splitting-in-Half method [Shi, et al. 2005]; and CPSO [Van den Bergh and Engelbrecht 2004].

#### Random grouping
Variable grouping is changed during the optimization run, e.g., random grouping [Yang, et al. 2008], and CCPSO2 [Li and Yao 2012].

#### Learnt Grouping
Variable grouping is learnt either before or during the optimization run, e.g., the CC technique for identifying interacting variables [Weicker and Weicker 1999], Delta Grouping [Omidvar, et al. 2010a], CCVIL [Chen, et al. 2010], and Differential Grouping [Omidvar, et al. 2014a].
Fixed grouping - CPSO

- Unlike CCEA by [Potter and De Jong 1994], in Cooperative Particle Swarm Optimization (CPSO) [Van den Bergh and Engelbrecht 2004] a $n$-dimensional problem is decomposed into $m$ $s$-dimensional subcomponents, where $s$ is the number of variables in a subcomponent.

- Once the decomposition of variables is decided at the beginning, this grouping remains fixed, which means that the arrangement of variables is not changed during the optimization.

- Interacting variables that happen to be placed in different subcomponents will remain so. This is against the idea to keep the interdependency between subcomponents to minimum.
Cooperative PSO

Figure: Concatenation of all the personal bests (from swarm $P_1$ to $P_K$) $P_1 \cdot \hat{y}, P_2 \cdot \hat{y}, \ldots, P_K \cdot \hat{y}$ constitutes the context vector $\hat{y}$. 
Cooperative PSO

Algorithm 1: The pseudocode of the CPSO algorithm.

Create and initialize $K$ swarms, each with $s$ dimensions (where $n = K \times s$); The $j$-th swarm is denoted as $P_j, j \in [1..K]$;

repeat
  for each swarm $j \in [1..K]$ do
    for each particle $i \in [1..\text{swarmSize}]$ do
      if $f(b(j, P_j.x_i)) < f(b(j, P_j.y_i))$ then $P_j.y_i \leftarrow P_j.x_i$;
      if $f(b(j, P_j.y_i)) < f(b(j, P_j.\hat{y}))$ then $P_j.\hat{y} \leftarrow P_j.y_i$;
    end
  Perform velocity and position updates for each particle in $P_j$;
end
until termination criterion is met;
Random Grouping

Motivation

Instead of using a fixed grouping for variables, it is possible to dynamically regroup the variables iteratively by randomly decomposing variables into different subcomponents.

One such method is Random Grouping by [Yang, et al. 2008, Omidvar, et al. 2010b, Li and Yao 2012], where decision variables are shuffled in each co-evolutionary cycle so that the probability of two interacting variables being placed in the same subcomponent is increased;
Random Grouping

Theorem

Given $N$ cycles, the probability of assigning $v$ interacting variables $x_1, x_2, \ldots, x_v$ into one subcomponent for at least $k$ cycles is:

$$P(X \geq k) = \sum_{r=k}^{N} \binom{N}{r} \left( \frac{1}{m^{v-1}} \right)^r \left( 1 - \frac{1}{m^{v-1}} \right)^{N-r}$$  \hspace{1cm} (4)

where $N$ is the number of cycles, $v$ is the total number of interacting variables, $m$ is the number of subcomponents, and the random variable $X$ is the number of times that $v$ interacting variables are grouped in one subcomponent.
Random Grouping

Lemma

A variable can be assigned to a subcomponent in \( m \) different ways, and since there are \( v \) interacting variables, the probability of assigning all of the interacting variables into one subcomponent would be:

\[
p_{sub} = \frac{1}{m} \times \ldots \times \frac{1}{m} = \frac{1}{m^v}
\]

Since there are \( m \) different subcomponents, the probability of assigning all \( v \) variables to any of the subcomponents would be:

\[
p = m \times p_{sub} = \frac{m}{m^v} = \frac{1}{m^{v-1}}
\]
Random Grouping

Proof.

There are a total of $N$ independent random decompositions of variables into $m$ subcomponents, so using a binomial distribution the probability of assigning $v$ interacting variables into one subcomponent for exactly $r$ times would be:

$$P(X = r) = \binom{N}{r} p^r (1 - p)^{N-r}$$

$$= \binom{N}{r} \left(\frac{1}{m^{v-1}}\right)^r \left(1 - \frac{1}{m^{v-1}}\right)^{N-r}$$

Thus,

$$P(X \geq k) = \sum_{r=k}^{N} \binom{N}{r} \left(\frac{1}{m^{v-1}}\right)^r \left(1 - \frac{1}{m^{v-1}}\right)^{N-r}$$
Random Grouping

Example

Given \( n = 1000 \), \( m = 10 \), \( N = 50 \) and \( v = 4 \), we have:

\[
P(X \geq 1) = 1 - P(X = 0) = 1 - \left(1 - \frac{1}{10^3}\right)^{50} = 0.0488
\]

which means that over 50 cycles, the probability of assigning 4 interacting variables into one subcomponent for at least 1 cycle is only 0.0488. As we can see this probability is very small, and it will be even less if there are more interacting variables.
Figure: Increasing $v$, the number of interacting variables will significantly decrease the probability of grouping them in one subcomponent, given $n = 1000$ and $m = 10$. 
Increasing the number of cycles $N$

Figure: Increasing $N$, the number of cycle increases the probability of grouping $v$ number of interacting variables in one subcomponent.
Our results in [Omidvar, et al. 2010a] suggested that:

- More frequent random grouping result in faster convergence without sacrificing solution quality, due to increased probability in grouping interacting variables in a subcomponent.
- More frequent random grouping also increases the efficiency in dealing with problems with two interacting variables, but as the number of interacting variables increases (e.g., 6 or 7), the probability of these interacting variables being placed in the same subcomponent drops very rapidly.
- For problems with a large number of interacting variables, random grouping will not be very helpful.

**Question:** Can we do better than just random grouping, which relies on shuffling variables in order to increase the probability of placing interacting variables together? Can this be learnt?
CC Variable Interaction Learning

Key ideas

- [Chen, et al. 2010] improved the technique by [Weicker and Weicker 1999] and proposed CC variable interaction learning (CCVIL) and applied it to LSGO.

- CCVIL exploits the knowledge about partially (or additively) separable functions.

- If a function $f$ is separable, then its global optimum can be reached by successive line search along the axes. If $f$ is not separable, then there must be interactions between at least two variables in the decision vector.
CC Variable Interaction Learning

**CCVIL**

\[ \exists \vec{x}, x'_i, x'_j : \]
\[ f(x_1, ..., x_i, ..., x_j, ..., x_n) < f(x_1, ..., x'_i, ..., x_j, ..., x_n) \land \]
\[ f(x_1, ..., x_i, ..., x'_j, ..., x_n) > f(x_1, ..., x'_i, ..., x'_j, ..., x_n) \]  

(5)

where \( \vec{x} \) is a candidate decision vector and \( x'_i, x'_j \) are two values to be replaced by the \( i \)th and \( j \)th decision variables respectively.

**Learning stage:**

In the **variable interaction learning** stage, initially each variable is placed in a separate subcomponent. Then, with repeated applications of the above equation to any two dimensions \( i \) and \( j \), the interacting dimensions are merged until the termination criteria is met. CCVIL is still based on a cooperative coevolutionary (CC) framework.
Outline

1. Large Scale Global Optimization
2. Cooperative Coevolution
3. Decomposition Methods with CC
4. Contribution Based Cooperative Co-evolution (CBCC)
5. Differential Grouping
6. CEC’2013 LSGO Benchmark Test Functions
7. Route Distance Grouping for Capacitated Arc Routing Problems
8. Conclusions
Motivation

- When dealing with non-separable problems, there is usually an imbalance between the contribution of various subcomponents to the global fitness;
- In cooperative coevolution, a round-robin method is employed to optimize all of the subcomponents in an iterative manner. This switching strategy splits the computational budget equally between all subcomponents.
- In the presence of imbalance between subcomponents, the round-robin method is not computationally efficient.
- More computational budget should be spent on the subcomponents with the greatest contribution to the improvement of global fitness.
The Imbalanced Problems

CEC’2010 $f_4$

\[ f_4(\vec{x}) = 10^6 \times f_{\text{elliptic}}(R[x_{p_1}, \ldots, x_{p_m}]^T) + f_{\text{elliptic}}([x_{p_{m+1}}, \ldots, x_{p_n}]^T), \]

![Graph showing the comparison of DECC-I-Nonsep and DECC-I-Sep methods over iterations.](image)
CEC’2010 \( f_{14} \) : How Realistic It Is?

**CEC’2010 \( f_{14} \)**

\[
f_{14} = \sum_{k=1}^{\frac{D}{m}} f_{\text{elliptic}}(R_k \vec{x}_k), \quad \vec{x}_k \in \mathbb{R}^m
\]

**A more realistic situation**

\[
f = \sum_{i=1}^{m} w_i \times f_i(\vec{x}_i)
\]  \hspace{1cm} (6)

where \( \vec{x}_i \) are mutually exclusive decision vectors of functions \( f_i \), and \( m \) is the number independent subcomponents in the global fitness function \( f \). \( w_i \) is the coefficient.
How to achieve CBCC?

- Firstly, subcomponents with minimum interdependency are identified. This requires an effective variable grouping method.
- Secondly, contributions to the global fitness from different subcomponents need to be properly measured. Note that in a real-world scenario we often do not have prior knowledge to the fitness of subcomponents. The global fitness is all we have.

Method

Given an ideal decomposition of decision variables, if we optimize one subcomponent at a time, the changes in the global fitness function is the reflection of changes in the subcomponent that undergoes optimization.

- The changes in the global fitness function under a near optimum decomposition could be served as measure for the contribution of various subcomponents.
The CBCC Algorithm

Algorithm 2: CBCC(FEs)

1. \( \text{pop}[1 : N_{\text{popsize}}, 1 : n] \leftarrow \text{random population} \)
2. evaluate the \( \text{pop} \) using an EA
3. initialize prev_best, cur_best
4. \( \Delta F \leftarrow \text{zeros}(1, \text{num}\_\text{groups}) \)
5. while termination criteria is not reached do
6. for \( i \leftarrow 1 \) to \( \text{num}\_\text{groups} \) do
7. optimize and evaluate the \( i^{th} \) subpopulation using an EA
8. update prev_best, cur_best
9. \( \Delta F[i] \leftarrow \Delta F[i] + |\text{prev_best} - \text{cur_best}| \)
10. end for
11. \( \delta \leftarrow 1 \)
12. while \( \delta \neq 0 \) do
13. optimize the subpopulation with the highest contribution using an EA
14. update prev_best, cur_best
15. \( \delta \leftarrow |\text{prev_best} - \text{cur_best}| \)
16. \( \Delta F[\text{max}\_\text{contrib}] \leftarrow \Delta F[\text{max}\_\text{contrib}] + \delta \)
17. end while
18. end while

Further information on CBCC can be found from [Omidvar, et al. 2011].
Outline

1. Large Scale Global Optimization
2. Cooperative Coevolution
3. Decomposition Methods with CC
4. Contribution Based Cooperative Co-evolution (CBCC)
5. Differential Grouping
6. CEC’2013 LSGO Benchmark Test Functions
7. Route Distance Grouping for Capacitated Arc Routing Problems
8. Conclusions
Differential Grouping

Motivation

- The structure of a problem is not always known in advance, hence manual decomposition is not always straightforward;
- New systematic procedures are needed to learn the underlying structure of the problem.
- **Differential Grouping** [Omidvar, et al. 2014a] is developed to discover interacting variables so that they can be placed in the same subcomponents.
- This facilitates automatic decomposition (or grouping) of decision variables, so that the interdependency between subproblems is kept at minimum.
Additively or partially separable functions

Definition

A function is said to be *additively or partially separable* if it has the following general form:

\[ f(\tilde{x}) = \sum_{i=1}^{m} f_i(\tilde{x}_i) , \]  

where \( \tilde{x}_i \) are mutually exclusive decision vectors of \( f_i \), and \( \tilde{x} = \langle x_1, \ldots, x_n \rangle \) is the global decision vector of \( n \) dimensions and \( m \) is the number of independent subcomponents in the global objective function \( f \).
Differential Grouping

**Theorem**

Let $f(\vec{x})$ be an additively separable function. $\forall a, b_1 \neq b_2, \delta \in \mathbb{R}, \delta \neq 0$, if the following condition holds

$$
\Delta_{\delta, x_p} [f](\vec{x}) |_{x_p=a, x_q=b_1} \neq \Delta_{\delta, x_p} [f](\vec{x}) |_{x_p=a, x_q=b_2},
$$

(8)

then $x_p$ and $x_q$ are non-separable, where

$$
\Delta_{\delta, x_p} [f](\vec{x}) = f(\ldots, x_p + \delta, \ldots) - f(\ldots, x_p, \ldots),
$$

(9)

refers to the forward difference of $f$ with respect to variable $x_p$ with interval $\delta$. 
Differential Grouping

Illustration

We calculate the $\Delta_{x_i}[f]$ twice in the following way:

$$\Delta 1_{x_i}[f] = f(\ldots, a_i + \delta, b_j, \ldots) - f(\ldots, a_i, b_j, \ldots).$$

$$\Delta 2_{x_i}[f] = f(\ldots, a_i + \delta, c_j, \ldots) - f(\ldots, a_i, c_j, \ldots).$$

where $a_i$ is an arbitrarily chosen value for the $i$-th variable, and $b_j$ and $c_j$ are two arbitrarily chosen different values for the $j$-th variable. If $\Delta 1_{a_i}[f] = \Delta 2_{a_i}[f]$, then it can be said that $i$-th and $j$-th variables are independent with each other. Otherwise, they are interacting with each other.
Separability \Rightarrow \Delta_1 = \Delta_2

Assuming:

\[ f(\bar{x}) = \sum_{i=1}^{m} f_i(\bar{x}_i) \]

We prove that:

Separability \Rightarrow \Delta_1 = \Delta_2

By contraposition (\(P \Rightarrow Q \equiv \neg Q \Rightarrow \neg P\)):

\[ \Delta_1 \neq \Delta_2 \Rightarrow \text{non-separability} \]

or

\[ |\Delta_1 - \Delta_2| > \varepsilon \Rightarrow \text{non-separability} \]
### Differential Grouping

<table>
<thead>
<tr>
<th>Detecting Non-separable Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Detecting Separable Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
</tr>
</tbody>
</table>
Example (Partially Separable Function)

Given an objective function \( f(x, y) = x^2 + y^2 \), we have:

\[
\frac{\partial f(x, y)}{\partial x} = 2x.
\]

This clearly shows that the changes in the global objective function caused by modifications to \( x \) are independent of \( y \). Now by applying Equation (9) we have:

\[
\Delta_x[f] = [(x + \delta)^2 + y^2] - [x^2 + y^2] = \delta^2 + 2\delta x.
\]

It can be seen that \( \Delta_x[f] \) does not depend on \( y \). Therefore, we conclude that \( x \) and \( y \) are independent.
Example (Non-separable Function)

Given an objective function \( f(x, y) = x^2 + \lambda xy + y^2, \lambda \neq 0 \), we have:

\[
\frac{\partial f(x, y)}{\partial x} = 2x + \lambda y.
\]

The changes in the global objective function is a function of \( x \) and \( y \). Now by applying Equation (9) we have:

\[
\Delta_x[f] = [(x + \delta)^2 + \lambda(x + \delta)y + y^2] - [x^2 + \lambda xy + y^2] = \delta^2 + 2\delta x + \lambda y \delta.
\]

\( \Delta_x[f] \) depends on both \( x \) and \( y \), and evaluating the difference equation for two different values of \( y \) does not give the same answer. Hence we conclude that \( x \) and \( y \) are interacting with each other.
CC framework with DG

Algorithm 4: CC(func, lbounds, ubounds, n)

1. `groups ← grouping(func, lbounds, ubounds, n)`  // grouping stage.
2. `pop ← rand(popsize, n)`  // optimization stage.
3. `(best, best_val) ← min(func(pop))`
4. `for i ← 1 to cycles do`
5. `for j ← 1 to size(groups) do`
6. `indices ← groups[j]`
7. `subpop ← pop[:, indices]`
8. `subpop ← optimizer(best, subpop, FEs)`
9. `pop[:, indices] ← subpop`
10. `(best, best_val) ← min(func(pop))`
11. `end for`
12. `end for`

Two separate stages:

- **Grouping stage**: it can be any off-line grouping procedure, e.g., *differential grouping*.
- **Optimization stage**: The identified subcomponents are optimized in a round-robin fashion as in CC. The subcomponent optimizer can be any optimization method.
**Algorithm 3: allgroups ← differential_grouping(func, lbounds, ubounds, n)**

1. \( \text{dims} \leftarrow \{1, 2, \ldots, n\} \)
2. \( \text{seps} \leftarrow \{\} \)
3. \( \text{allgroups} \leftarrow \{\} \)  // contains a set of all identified groups.
4. for \( i \in \text{dims} \) do
5.      \( \text{group} \leftarrow \{i\} \)
6.   for \( j \in \text{dims} \land i \neq j \) do
7.      \( \vec{p}_1 \leftarrow \text{lbound} \times \text{ones}(1, n) \)
8.      \( \vec{p}_2 \leftarrow \vec{p}_1 \)
9.      \( \vec{p}_2(i) \leftarrow \text{ubound} \)
10. \( \Delta_1 \leftarrow \text{func}(\vec{p}_1) - \text{func}(\vec{p}_2) \)
11. \( \vec{p}_1(j) \leftarrow 0 \)
12. \( \vec{p}_2(j) \leftarrow 0 \)
13. \( \Delta_2 \leftarrow \text{func}(\vec{p}_1) - \text{func}(\vec{p}_2) \)
14. if \( |\Delta_1 - \Delta_2| > \epsilon \) then
15.      \( \text{group} \leftarrow \text{group} \cup j \)
16. end if
17. end for
18. \( \text{dims} \leftarrow \text{dims} - \text{group} \)
19. if \( \text{length}(	ext{group}) = 1 \) then
20.      \( \text{seps} \leftarrow \text{seps} \cup \text{group} \)
21. else
22.      \( \text{allgroups} \leftarrow \text{allgroups} \cup \{\text{group}\} \)
23. end if
24. end for
25. \( \text{allgroups} \leftarrow \text{allgroups} \cup \{\text{seps}\} \)
Differential Grouping vs CCVIL

Figure: Detection of interacting variables using differential grouping and CCVIL on different regions of a 2D Schwefel Problem 1.2.
Differential Grouping vs CCVIL

Referring to the figure in the previous slide.

- Differential grouping and CCVIL behave differently depending on the positions of the chosen sample points, three regions (A, B, and C) are marked on a two-dimensional version of the Schwefel’s Problem 1.2, where both variables are interacting with each other.

- The condition given in Equation (5) which is used in CCVIL is only satisfied in region A, but not for points in regions B or C. CCVIL will need to continue its search with more effort.

- Unlike CCVIL, which directly compares the fitness of the sample points, differential grouping compares the difference between the elevation of the two points connected in a dashed line (as shown in the Figure), (|f(x_1, x_2) − f(x_1 + δ, x_2)| and |f(x_1, x'_2) − f(x_1 + δ, x'_2)|). If this difference in elevation of the two pairs is different, it is inferred that the corresponding dimensions are non-separable.
### Benchmark Functions

1. **Separable Functions** \((f_1-f_3)\)
2. **Single-group** \(m\)-**nonseparable Functions** \((f_4-f_8)\)
   - \(f_4\): Single-group Shifted and \(m\)-rotated Elliptic Function.
   - \(f_5\): Single-group Shifted and \(m\)-rotated Rastrigin’s Function.
   - \(f_6\): Single-group Shifted and \(m\)-rotated Ackley’s Function.
   - \(f_7\): Single-group Shifted and \(m\)-rotated Schwefel’s Problem 1.2.
   - \(f_8\): Single-group Shifted and \(m\)-rotated Rosenbrock’s Function.
3. **\(\frac{n}{2m}\)-group** \(m\)-**nonseparable Functions** \((f_9-f_{13})\)
4. **\(\frac{n}{m}\)-group** \(m\)-**nonseparable Functions** \((f_{14}-f_{18})\)
5. **Nonseparable Functions** \((f_{19}-f_{20})\)

### Experiment Setup

- **Dimensionality of the functions**: 1000; \(m = 50\).
- **Number of fitness evaluations**: \(3 \times 10^6\)
- **Population Size**: 50; Subcomponent Optimizer: SaNSDE.
- **\(\epsilon\)** (see Algorithm 3) is set \(10^{-3}\).
- The average, mean and standard deviation are recorded over 25 independent runs.
### Differential Grouping ($\epsilon = 10^{-3}$) / CCVIL

<table>
<thead>
<tr>
<th>Function</th>
<th>Non-sep Vars</th>
<th>Non-sep Groups</th>
<th># Misplaced Vars</th>
<th># FE Vars</th>
<th>Grouping Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_1$</td>
<td>0</td>
<td>0</td>
<td>0 / 0</td>
<td>1001000 / 69990</td>
<td>100% / 100%</td>
</tr>
<tr>
<td>$f_2$</td>
<td>0</td>
<td>0</td>
<td>0 / 0</td>
<td>1001000 / 69990</td>
<td>100% / 100%</td>
</tr>
<tr>
<td>$f_3$</td>
<td>0</td>
<td>0</td>
<td>0 / 31</td>
<td>1001000 / 1798666</td>
<td>100% / 93.8%</td>
</tr>
<tr>
<td>$f_4$</td>
<td>50</td>
<td>1</td>
<td>0 / 7</td>
<td>14564 / 1797614</td>
<td>100% / 86%</td>
</tr>
<tr>
<td>$f_5$</td>
<td>50</td>
<td>1</td>
<td>0 / 0</td>
<td>905450 / 1795705</td>
<td>100% / 100%</td>
</tr>
<tr>
<td>$f_6$</td>
<td>50</td>
<td>1</td>
<td>0 / 3</td>
<td>906332 / 1796370</td>
<td>100% / 94%</td>
</tr>
<tr>
<td>$f_7$</td>
<td>50</td>
<td>1</td>
<td>16 / 1</td>
<td>67250 / 1796475</td>
<td>69% / 98%</td>
</tr>
<tr>
<td>$f_8$</td>
<td>50</td>
<td>1</td>
<td>4 / 50</td>
<td>23608 / 69842</td>
<td>92% / 0%</td>
</tr>
<tr>
<td>$f_9$</td>
<td>500</td>
<td>10</td>
<td>0 / 0</td>
<td>270802 / 1792212</td>
<td>100% / 67.4%</td>
</tr>
<tr>
<td>$f_{10}$</td>
<td>500</td>
<td>10</td>
<td>0 / 8</td>
<td>272958 / 1774642</td>
<td>100% / 98.4%</td>
</tr>
<tr>
<td>$f_{11}$</td>
<td>500</td>
<td>10</td>
<td>1 / 9</td>
<td>270640 / 1774565</td>
<td>99.2% / 98.2%</td>
</tr>
<tr>
<td>$f_{12}$</td>
<td>500</td>
<td>10</td>
<td>0 / 65</td>
<td>271390 / 1777344</td>
<td>100% / 87%</td>
</tr>
<tr>
<td>$f_{13}$</td>
<td>500</td>
<td>10</td>
<td>374 / 500</td>
<td>49470 / 69990</td>
<td>25.2% / 0%</td>
</tr>
<tr>
<td>$f_{14}$</td>
<td>1000</td>
<td>20</td>
<td>0 / 281</td>
<td>21000 / 1785975</td>
<td>100% / 71.9%</td>
</tr>
<tr>
<td>$f_{15}$</td>
<td>1000</td>
<td>20</td>
<td>0 / 18</td>
<td>21000 / 1751241</td>
<td>100% / 98.2%</td>
</tr>
<tr>
<td>$f_{16}$</td>
<td>1000</td>
<td>20</td>
<td>4 / 11</td>
<td>21128 / 1751647</td>
<td>99.6% / 98.9%</td>
</tr>
<tr>
<td>$f_{17}$</td>
<td>1000</td>
<td>20</td>
<td>0 / 25</td>
<td>21000 / 1752340</td>
<td>100% / 97.5%</td>
</tr>
<tr>
<td>$f_{18}$</td>
<td>1000</td>
<td>20</td>
<td>827 / 1000</td>
<td>34230 / 69990</td>
<td>17.3% / 0%</td>
</tr>
<tr>
<td>$f_{19}$</td>
<td>1000</td>
<td>1</td>
<td>0 / 0</td>
<td>2000 / 48212</td>
<td>100% / 100%</td>
</tr>
<tr>
<td>$f_{20}$</td>
<td>1000</td>
<td>1</td>
<td>918 / 980</td>
<td>22206 / 1798708</td>
<td>8.2% / 2%</td>
</tr>
</tbody>
</table>
DG compares with CCVIL’s grouping performance

- Differential grouping algorithm performs a more accurate grouping with considerably fewer fitness evaluations on most of the functions except for $f_1$, $f_2$, and $f_7$.
- CCVIL performs even worse than differential grouping on all instances of the Rosenbrock function.
- An advantage of CCVIL is its ability to quickly detect fully separable variables with a relatively low number of fitness evaluations.
<table>
<thead>
<tr>
<th>Functions</th>
<th>DECC-DG</th>
<th>MLCC</th>
<th>DECC-D</th>
<th>DECC-DML</th>
<th>DECC-I</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_1$</td>
<td>5.47e+03</td>
<td>1.53e-27</td>
<td>1.01e-24</td>
<td>1.93e-25</td>
<td>1.73e+00</td>
</tr>
<tr>
<td>$f_2$</td>
<td>4.39e+03</td>
<td>5.57e-01</td>
<td>2.99e+02</td>
<td><strong>2.17e+02</strong></td>
<td>4.40e+03</td>
</tr>
<tr>
<td>$f_3$</td>
<td>1.67e+01</td>
<td>9.88e-13</td>
<td><strong>1.81e-13</strong></td>
<td><strong>1.18e-13</strong></td>
<td>1.67e+01</td>
</tr>
<tr>
<td>$f_4$</td>
<td>4.79e+12</td>
<td>9.61e+12</td>
<td><strong>3.99e+12</strong></td>
<td><strong>3.58e+12</strong></td>
<td>6.13e+11</td>
</tr>
<tr>
<td>$f_5$</td>
<td><strong>1.55e+08</strong></td>
<td>3.84e+08</td>
<td>4.16e+08</td>
<td>2.98e+08</td>
<td>1.34e+08</td>
</tr>
<tr>
<td>$f_6$</td>
<td><strong>1.64e+01</strong></td>
<td>1.62e+07</td>
<td>1.36e+07</td>
<td>7.93e+05</td>
<td>1.64e+01</td>
</tr>
<tr>
<td>$f_7$</td>
<td><strong>1.16e+04</strong></td>
<td>6.89e+05</td>
<td>6.58e+07</td>
<td>1.39e+08</td>
<td>2.97e+01</td>
</tr>
<tr>
<td>$f_8$</td>
<td><strong>3.04e+07</strong></td>
<td>4.38e+07</td>
<td>5.39e+07</td>
<td><strong>3.46e+07</strong></td>
<td>3.19e+05</td>
</tr>
<tr>
<td>$f_9$</td>
<td>5.96e+07</td>
<td>1.23e+08</td>
<td><strong>6.19e+07</strong></td>
<td><strong>5.92e+07</strong></td>
<td>4.84e+07</td>
</tr>
<tr>
<td>$f_{10}$</td>
<td><strong>4.52e+03</strong></td>
<td><strong>3.43e+03</strong></td>
<td>1.16e+04</td>
<td>1.25e+04</td>
<td>4.34e+03</td>
</tr>
<tr>
<td>$f_{11}$</td>
<td>1.03e+01</td>
<td>1.98e+02</td>
<td>4.76e+01</td>
<td><strong>1.80e-13</strong></td>
<td>1.02e+01</td>
</tr>
<tr>
<td>$f_{12}$</td>
<td><strong>2.52e+03</strong></td>
<td>3.49e+04</td>
<td>1.53e+05</td>
<td>3.79e+06</td>
<td>1.47e+03</td>
</tr>
<tr>
<td>$f_{13}$</td>
<td>4.54e+06</td>
<td>2.08e+03</td>
<td><strong>9.87e+02</strong></td>
<td><strong>1.14e+03</strong></td>
<td>7.51e+02</td>
</tr>
<tr>
<td>$f_{14}$</td>
<td>3.41e+08</td>
<td>3.16e+08</td>
<td>1.98e+08</td>
<td><strong>1.89e+08</strong></td>
<td>3.38e+08</td>
</tr>
<tr>
<td>$f_{15}$</td>
<td><strong>5.88e+03</strong></td>
<td>7.11e+03</td>
<td>1.53e+04</td>
<td>1.54e+04</td>
<td>5.87e+03</td>
</tr>
<tr>
<td>$f_{16}$</td>
<td><strong>7.39e-13</strong></td>
<td>3.76e+02</td>
<td>1.88e+02</td>
<td>5.08e-02</td>
<td>2.47e-13</td>
</tr>
<tr>
<td>$f_{17}$</td>
<td><strong>4.01e+04</strong></td>
<td>1.59e+05</td>
<td>9.03e+05</td>
<td>6.54e+06</td>
<td>3.91e+04</td>
</tr>
<tr>
<td>$f_{18}$</td>
<td>1.11e+10</td>
<td>7.09e+03</td>
<td><strong>2.12e+03</strong></td>
<td><strong>2.47e+03</strong></td>
<td>1.17e+03</td>
</tr>
<tr>
<td>$f_{19}$</td>
<td><strong>1.74e+06</strong></td>
<td>1.36e+06</td>
<td>1.33e+07</td>
<td>1.59e+07</td>
<td>1.74e+06</td>
</tr>
<tr>
<td>$f_{20}$</td>
<td>4.87e+07</td>
<td>2.05e+03</td>
<td><strong>9.91e+02</strong></td>
<td><strong>9.91e+02</strong></td>
<td>4.14e+03</td>
</tr>
</tbody>
</table>
DECC-DG outperforms other algorithms on non-separable functions, whereas DECC-DG is outperformed by DECC-DML on separable functions $f_1$, $f_2$, and $f_3$.

DECC-DG outperforms DECC-DML when grouping accuracy is high. DECC-DG performs poorly on instances of the Rosenbrock functions ($f_8$, $f_{13}$, $f_{18}$, and $f_{20}$), where low grouping accuracy is obtained.

Comparing with DECC-I (where ideal grouping is used), the results show in most cases, DECC-DG benefits from utilizing grouping information.
Convergence plots on $f_6$ and $f_9$

Observation

Although a certain number of fitness evaluations have been used to discover the ideal grouping structure, this effort is compensated for in the optimization phase due to optimum grouping structures.
Results on DG with contribution-based CC

It is arguable that in most real-world problems, some imbalance exists between various subcomponents. Hence we modified CEC’2010 benchmark functions $f_9$ to $f_{13}$ (category 3) $f_{14}$ to $f_{18}$ (category 4) to allow imbalance to be considered.

**Imbalanced functions**

$$F_{cat3} = \sum_{i=0}^{\frac{n}{2m}-1} 10^{2(i-9)} \times F_{\text{nonsep}} + F_{\text{sep}}$$

$$F_{cat4} = \sum_{i=0}^{\frac{n}{m}-1} 10^{(i-9)} \times F_{\text{nonsep}} + F_{\text{sep}}.$$
Convergence plots on $f'_{11}$ and $f'_{12}$

Observation

For MA-SW-Chains, there is initially a drastic improvement in the fitness value, and thereafter it becomes stagnant. This is largely due to MA-SW-Chains’ strong local search ability (it is actually a memetic algorithm).
CBCC-DG vs DECC-DG and MA-SW-CHAINS

Table: CBCC-DG’s number of wins, loses and ties against DECC-DG and MA-SW-Chains before and after inclusion of imbalance in benchmark functions ($f_4$-$f_8$ and $f'_9$-$f'_{18}$)

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Balanced</th>
<th></th>
<th></th>
<th>Imbalanced</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Wins</td>
<td>Loses</td>
<td>Ties</td>
<td>Wins</td>
<td>Loses</td>
<td>Ties</td>
<td></td>
</tr>
<tr>
<td>DECC-DG</td>
<td>7</td>
<td>5</td>
<td>3</td>
<td>9</td>
<td>2</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>MA-SW-Chains</td>
<td>5</td>
<td>10</td>
<td>0</td>
<td>6</td>
<td>7</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

Observation

Contribution-based CC is beneficial especially for dealing with imbalanced problems. CBCC is just one simple scheme. More effective contribution-based CC schemes are possible.
Outline

1. Large Scale Global Optimization
2. Cooperative Coevolution
3. Decomposition Methods with CC
4. Contribution Based Cooperative Co-evolution (CBCC)
5. Differential Grouping
6. CEC’2013 LSGO Benchmark Test Functions
7. Route Distance Grouping for Capacitated Arc Routing Problems
8. Conclusions
CEC’2013 LSGO benchmark test functions

Designed to challenge large-scale black-box optimization algorithms, especially their ability to decompose large-scale problems. This was built on the success of CEC’2008, CEC’2010, and CEC’2012 Special Session and Competition on Large Scale Global Optimization.

CEC’2013 LSGO benchmark

[Li, et al. 2013, Omidvar, et al. 2015]

- 15 large-scale benchmark test functions, an extension to the CEC’2010 benchmark functions;
- Facilitate comparative studies between various evolutionary algorithms for large-scale global optimization;
- Introducing imbalance between various subcomponents;
- Subcomponents with nonuniform sizes;
- Conforming and conflicting overlapping functions.
- New transformations to the base functions: ill-conditioning, symmetry breaking, and irregularities.
Outline

1. Large Scale Global Optimization
2. Cooperative Coevolution
3. Decomposition Methods with CC
4. Contribution Based Cooperative Co-evolution (CBCC)
5. Differential Grouping
6. CEC’2013 LSGO Benchmark Test Functions
7. Route Distance Grouping for Capacitated Arc Routing Problems
8. Conclusions

- 3000 gritting routes
- 120,000 km or 30% of the entire road network
- Millions of pounds each year
- Large Scale Combinatorial Optimization!

**Figure:** Black ice hazard
An example of South Gloucester, UK

Figure: Temperature distribution

Figure: A routing plan
Decomposition: Route Distance Grouping

Figure: Grouping routes that are close to each other.

Figure: Optimizing the grouped routes.

Further information can be found in [Mei, et al. 2014].
IEEE CIS Taskforce on Large Scale Global Optimization

- Promote research for large scale optimization problems;
- Facilitate the knowledge sharing and collaboration between researchers in the related areas;
- Exchange experience and promote discussion and contacts between researchers, industrialists and practitioners.

Further information:

In 2015, the taskforce is involved in the following:

- CEC’2015 Special session/competition on Large Scale Global Optimization.
- CEC’2015 tutorial on “Decomposition and Cooperative Coevolution Techniques for Large Scale Global Optimization”.

Outline

1. Large Scale Global Optimization
2. Cooperative Coevolution
3. Decomposition Methods with CC
4. Contribution Based Cooperative Co-evolution (CBCC)
5. Differential Grouping
6. CEC’2013 LSGO Benchmark Test Functions
7. Route Distance Grouping for Capacitated Arc Routing Problems
8. Conclusions
Conclusions

- Various decomposition methods using cooperative coevolution (CC) techniques have been developed over the years, and a few recent methods have shown particularly promising results for LSGO problems.

- A related issue to decomposition, in the presence of imbalanced problems, is how to best spend computational budget on the subproblems which contributes the most to the global fitness. We have shown that a contribution based CC method can improve over the traditional CC.

Future works

- A more accurate contribution assessment scheme that can quickly respond to the changes.
- What is the optimal decomposition for totally separable problems, given a fixed computational budget? See a recent work on this [Omidvar, et al. 2014b].
- Effects of different population sizes in different subcomponents.
- What would be the competent optimizer for a subcomponent in CC, given the optimal (or close to optimal) grouping of variables discovered?
- How to better deal with the overlapping functions as presented in the technical report of CEC’2013 LSGO benchmark functions?
Acknowledgement

Many thanks to following people:

- Mohammad Nabi Omidvar for providing many slides and figures;
- Xin Yao for an EPSRC grant to start our collaboration on LSGO;
- Ke Tang and Zhenyu Yang for interesting discussions and valuable feedback over the past few years;
- Michael Kirley (Melbourne University), Yi Mei, Kai Qin, and others in RMIT ECML group for useful feedback.
- Support from an ARC Discovery Grant (DP120102205) and an EPSRC Grant.


Bibliography II


Questions