Evolutionary Large-Scale Global Optimization
An Introduction

Mohammad Nabi Omidvar\textsuperscript{1}  Xiaodong Li\textsuperscript{2}

\textsuperscript{1}School of Computer Science
University of Birmingham
m.omidvar@cs.bahm.ac.uk

\textsuperscript{2}School of Science
RMIT University, Melbourne, Australia
xiaodong.li@rmit.edu.au
Outline

1. Introduction: Large Scale Global Optimization
2. Approaches to Large-Scale Optimization
3. Variable Interaction: Definitions and Importance
4. Interaction Learning: Exploiting Modularity
5. Conclusion
6. Questions
Optimization

\[
\begin{align*}
\min & \quad f(x), \ x = (x_1, \ldots, x_n) \in \mathbb{R}^n \\
\text{s.t.} & \quad g(x) \leq 0 \\
& \quad h(x) = 0
\end{align*}
\]

(1)

Can be converted to unconstrained optimization using:

- Penalty method;
- Lagrangian;
- Augmented Lagrangian.

Our focus is unconstrained optimization. We must learn how to walk before we can run.
Large Scale Global Optimization (LSGO)

How large is large?

- The notation of large-scale is not fix.
- Changes over time.
- Differs from problem to problem.
- The dimension at which existing methods start to fail.

State-of-the-art (EC)

- Binary: $\approx 1$ billion $[a]$.
- Integer (linear): $\approx 1$ billion $[b], [c]$.
- Real: $\approx 1000-5000$.

---


Large Scale Global Optimization: Applications

Why large-scale optimization is important?

- Growing applications in various fields.
  - Target shape design optimization [a].
  - Satellite layout design [b].
  - Parameter estimation in large scale biological systems [c].
  - Seismic waveform inversion [d].
  - Parameter calibration of water distribution systems [e].
  - Vehicle routing [f].

---


Large Scale Global Optimization: Research
Large Scale Global Optimization: Research

Scopus

Copyright © 2017 Elsevier B.V. All rights reserved. Scopus® is a registered trademark of Elsevier B.V.
The Challenge of Large Scale Optimization

Why is it difficult?

- Exponential growth in the size of search space (curse of dimensionality).

Research Goal

- Improving search quality (get to the optimal point).
- Improving search efficiency (get there fast).
Large Scale Global Optimization: Evolutionary Approaches

1. Initialization
2. Sampling and Variation Operators
3. Approximation and Surrogate Modeling
4. Local Search and Memetic Algorithms
5. Decomposition and Divide-and-Conquer
6. Parallelization (GPU, CPU)
7. Hybridization
Study the importance of initialization methods [1] in large-scale optimization.

Initialization Methods

- Inconclusive evidence for or against initialization methods:
  - Uniform design works worse than RNG, while good-lattice point and opposition-based methods perform better [1].
  - Another study showed that population size has a more significant effect than the initialization [2].
  - Achieving uniformity is difficult in high-dimensional spaces [3].
  - Yet another study suggest comparing average performances may not reveal the effect of initialization [4].

- Shortcomings:
  - It is difficult to isolate the effect of initialization.
  - Different effect on different algorithms (mostly tested on DE).
  - Numerous parameters to study.


Sampling and Variation Operators

- Opposition-based sampling [1]
- Center-based sampling [2].
- Quantum-behaved particle swarm [3].
- Competitive Swarm Optimizer [4].
- Social learning PSO [5].
- Mutation operators [6], [7].


Approximation Methods and Surrogate Modeling

- High-Dimensional Model Representation (HDMR) [1].
- Radial Basis Functions [2].
- Kriging and Gradient-Enhanced Kriging Metamodels [3].
- Piecewise Polynomial (Spline) [4].

---


Local Search and Memetic Algorithms

- Multiple Trajectory Search (MTS) [1].
- Memetic algorithm with local search chaining [2].
  - MA-SW-Chains [3].
  - MA-SSW-Chains [4].
- Multiple offspring sampling (MOS) [5], [6].

---


Parallelization

- Algorithms capable of parallelization [1], [2].
- GPU [3], [4].
- CPU/OpenMP [5].


Hybridization (The best of both worlds)

- **Rationale:** benefiting from unique features of different optimizers.
  - EDA+DE: [1].
  - PSO+ABC: [2].
  - Different DE variants: JADE+SaNSDE [3].
  - PSO+ACO [4].
  - Minimum Population Search+CMA-ES [5].

---


Decomposition Methods

- Divide-and-conquer
- Dimensionality reduction
Variable Interaction, Linkage, Epistasis

What is variable interaction?

- Genetics: two genes are said to interact with each other if they collectively represent a feature at the phenotype level.
- The extent to which the fitness of one gene can be suppressed by another gene.
- The extent to which the value taken by one gene activates or deactivates the effect of another gene.

Why variable interaction?

- The effectiveness of optimization algorithms is affected by how much they respect variable interaction.
- Also applies to classic mathematical programming methods.
Variable Interaction, Linkage, Epistasis

Illustrative Example

- \( f(x, y) = x^2 + \lambda_1 y^2 \)
- \( g(x, y) = x^2 + \lambda_1 y^2 + \lambda_2 xy \)
Definitions

Variable Interaction

A variable $x_i$ is separable or does not interact with any other variable iff:

$$\arg\min_x f(x) = \left( \arg\min_{x_i} f(x), \arg\min_{\forall x_j, j \neq i} f(x) \right),$$

where $x = (x_1, \ldots, x_n)^\top$ is a decision vector of $n$ dimensions.

Partial Separability

A function $f(x)$ is partially separable with $m$ independent subcomponents iff:

$$\arg\min_x f(x) = \left( \arg\min_{x_1} f(x_1, \ldots), \ldots, \arg\min_{x_m} f(\ldots, x_m) \right),$$

$x_1, \ldots, x_m$ are disjoint sub-vectors of $x$, and $2 \leq m \leq n$.

Note: A function is fully separable if sub-vectors $x_1, \ldots, x_m$ are 1-dimensional (i.e., $m = n$).
Definitions

**Full Nonseparability**

A function $f(x)$ is fully non-separable if every pair of its decision variables interact with each other.

**Additive Separability**

A function is *partially additively separable* if it has the following general form:

$$
 f(x) = \sum_{i=1}^{m} f_i(x_i),
$$

where $x_i$ are mutually exclusive decision vectors of $f_i$, $x = (x_1, \ldots, x_n)^\top$ is a global decision vector of $n$ dimensions, and $m$ is the number of independent subcomponents.
Effect of Variable Interaction

Sampling and Variation Operators:

- GAs: inversion operator to promote tight linkage [1].
  - Increasing the likelihood of placing linked genes close to each other to avoid disruption by crossover.
  - Rotation of the landscape has a detrimental effect on GA [2].
- The need for rotationally invariance:
  - Model Building Methods:
    - Estimation of Distribution Algorithms and Evolutionary Strategies: Covariance Matrix Adaptation.
    - Bayesian Optimization: Bayesian Networks.
  - DE’s crossover is not rotationally invariant.
  - PSO is also affected by rotation [3].


Effect of Variable Interaction

1. Approximation and Surrogate Modelling:
   - Should be able to capture variable interaction.
   - Second order terms of HDMR.

2. Local Search and Memetic Algorithms:
   - What subset of variables should be optimized in each iteration of local search?
   - Coordinate-wise search may not be effective. Memetics perform well on separable functions! A coincidence?!

3. Decomposition and Divide-and-Conquer:
   - Interacting variables should be placed in the same component.
Linkage Learning and Exploiting Modularity

Implicit Methods:
- In EC:
  - Estimation of Distribution Algorithms
  - Bayesian Optimization: BOA, hBOA, Linkage Trees
  - Adaptive Encoding, CMA-ES
- Classic Optimization:

Explicit Methods:
- In EC:
  - Random Grouping
  - Statistical Correlation-Based Methods
  - Delta Grouping
  - Meta Modelling
  - Monotonicity Checking
  - Differential Grouping
- Classic Optimization
  - Block Coordinate Descent
  - Adaptive Coordinate Descent
Implicit Methods

- **Scaling Up EDAs:**
  - Model Complexity Control [1].
  - Random Matrix Projection [2].
  - Use of mutual information [3].
  - Cauchy-EDA [4].

- **Scaling up CMA-ES:**
  - CC-CMA-ES [5].
  - LM-CMA [6].

---


Scalability issues of EDAs

- Accurate estimation requires a large sample size which grows exponentially with the dimensionality of the problem [1].
- A small sample results in poor estimation of the eigenvalues [2].
- The cost of sampling from a multi-dimensional Gaussian distribution increases cubically with the problem size [3].

---


Random Projection EDA

![Diagram of random projection EDA](image)
Explicit Methods

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)
- Optimization: Large-scale linear programs (Dantzig)
- Politics: Divide and rule (In *Perpetual Peace* by Immanuel Kant: *Divide et impera* is the third political maxims.)
Decomposition in EAs: Cooperative Co-evolution [1]

CC is a Framework

CC as a scalability agent:

- CC is not an optimizer.
- Requires a component optimizer.
- CC coordinates how the component optimizer is applied to components.
- A scalability agent.
Challenges of CC

Main Questions

1. How to decompose the problem?
2. How to allocated resources?
3. How to coordinate?
The Decomposition Challenge

How to decompose?
- There are many possibilities.
- Which decomposition is the best?

Optimal decomposition
- It is governed by the interaction structure of decision variables.
- An optimal decomposition is the one that minimizes the interaction between components.
Survey of Decomposition Methods

- Uninformed Decomposition [1]
  - $n$ 1-dimensional components (the original CC)
  - $k$ $s$-dimensional components.
- Random Grouping [2]
- Statistical Correlation-Based Methods
- Delta Grouping [3]
- Meta Modelling [4]
- Monotonicity Checking [5]
- Differential Grouping [6]


Illustrative Example (Canonical CC)

Figure: Variable interaction of a hypothetical function.

- \( n \) 1-dimensional components:
  - \( C_1: \{x_1\}, \{x_2\}, \{x_3\}, \{x_4\}, \{x_5\}, \{x_6\}, \{x_7\} \)
  - \( C_2: \{x_1\}, \{x_2\}, \{x_3\}, \{x_4\}, \{x_5\}, \{x_6\}, \{x_7\} \)
  - ... 
  - \( C_c: \{x_1\}, \{x_2\}, \{x_3\}, \{x_4\}, \{x_5\}, \{x_6\}, \{x_7\} \)
Illustrative Example (fixed $k$ $s$-dimensional)

$k$ $s$-dimensional ($k = 2$, $s = 4$):

- $C_1$: \{x_1, x_2, x_3, x_4\}, \{x_5, x_6, x_7\}
- $C_2$: \{x_1, x_2, x_3, x_4\}, \{x_5, x_6, x_7\}
- ...
- $C_c$: \{x_1, x_2, x_3, x_4\}, \{x_5, x_6, x_7\}

Figure: Variable interaction of a hypothetical function.
Illustrative Example (Random Grouping)

Random Grouping \((k=2, s=4)\): 
- \(C_1:\ \{x_2, x_3, x_6, x_5\}, \{x_7, x_1, x_4\}\) 
- \(C_2:\ \{x_3, x_4, x_1, x_2\}, \{x_6, x_7, x_5\}\) 
- \(\ldots\) 
- \(C_c:\ \{x_1, x_5, x_6, x_7\}, \{x_2, x_4, x_3\}\)

Figure: Variable interaction of a hypothetical function.
Theorem

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

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

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

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 \( m = 10 \).
Figure: Increasing $N$, the number of cycle increases the probability of grouping $v$ number of interacting variables in one subcomponent.
Illustrative Example (Informed with Fixed Groups)

Figure: Variable interaction of a hypothetical function.

- Delta Grouping \((k = 2, s = 4)\):
  - \(C_1\): \(\{x_1, x_5, x_2, x_4\}, \{x_3, x_6, x_7\}\)
  - \(C_2\): \(\{x_3, x_5, x_6, x_7\}, \{x_1, x_2, x_4\}\)
  - ... 
  - \(C_c\): \(\{x_3, x_6, x_1, x_4\}, \{x_2, x_5, x_7\}\)
Delta Grouping

Decomposition and CC for LSGO

Mohammad Nabi Omidvar, Xiaodong Li
Infomred Decompositions with Fixed Groups

- Adaptive Variable Partitioning [1].
- Delta Grouping [2].
- Min/Max-Variance Decomposition (MiVD/MaVD) [3].
  - Sorts the dimensions based on the diagonal elements of the covariance matrix in CMA-ES.
- Fitness Difference Partitioning [4], [5], [6].


Infomred Decompositions with Variable Groups

- Multilevel Grouping: MLCC [1], MLSoft [2].
- Adaptive Variable Partitioning 2 [3].
- 4CDE [4].
- Fuzzy Clustering [5].


Illustrative Example (Exact Methods)

Figure: Variable interaction of a hypothetical function.

- **Differential Grouping and Variable Interaction Learning:**
  - $C_1$: $\{x_1, x_2, x_4\}, \{x_3, x_5, x_6, x_7\}$
  - $C_2$: $\{x_1, x_2, x_4\}, \{x_3, x_5, x_6, x_7\}$
  - $C_c$: $\{x_1, x_2, x_4\}, \{x_3, x_5, x_6, x_7\}$
Monotonicity Check

\[ \exists x, x_i', x_j' : f(x_1, ..., x_i, ..., x_j, ..., x_n) < f(x_1, ..., x_i', ..., x_j, ..., x_n) \land \\
 f(x_1, ..., x_i, ..., x_j', ..., x_n) > f(x_1, ..., x_i', ..., x_j', ..., x_n) \]
Monotonicity Check (Algorithms)

- Linkage Identification by Non-Monotonicity Detection [1]
- Adaptive Coevolutionary Learning [2]
- Variable Interaction Learning [3]
- Variable Interdependence Learning [4]
- Fast Variable Interdependence [5]

---


Differential Grouping [1]

Theorem

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

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

(5)

then \( x_p \) and \( x_q \) are non-separable, where

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

(6)

refers to the forward difference of \( f \) with respect to variable \( x_p \) with interval \( \delta \).

---

Separability $\Rightarrow \Delta_1 = \Delta_2$

Assuming:

$$f(x) = \sum_{i=1}^{m} f_i(x_i)$$

We prove that:

Separability $\Rightarrow \Delta_1 = \Delta_2$

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

$\Delta_1 \neq \Delta_2 \Rightarrow$ non-separability

or

$|\Delta_1 - \Delta_2| > \epsilon \Rightarrow$ non-separability
Deductive Reasoning

**Strong Syllogism**

\[ A \implies B \]

*\( A \) is true

\[ \therefore B \] is true

\[ A \implies B \]

*\( B \) is false

\[ \therefore A \] is false

**Weak Syllogism**

\[ A \implies B \]

*\( A \) is false

\[ \therefore B \] is less plausible

\[ A \implies B \]

*\( B \) is true

\[ \therefore A \] is more plausible
Deductive Reasoning - Example

**Strong Syllogism**
- Rain $\Rightarrow$ Cloud
  - It is rainy
    - $\therefore$ It is cloudy

- Rain $\Rightarrow$ Cloud
  - It is not cloudy
    - $\therefore$ It is not rainy

**Weak Syllogism**
- Rain $\Rightarrow$ Cloud
  - It is not rainy
    - $\therefore$ Cloud becomes less likely

- Rain $\Rightarrow$ Cloud
  - It is cloudy
    - $\therefore$ Rain becomes more likely
The Differential Grouping Algorithm

Detecting Non-separable Variables

\[ |\Delta_1 - \Delta_2| > \epsilon \Rightarrow \text{non-separability} \]

Detecting Separable Variables

\[ |\Delta_1 - \Delta_2| \leq \epsilon \Rightarrow \text{Separability (more plausible)} \]
Consider the non-separable objective function \( f(x_1, x_2) = x_1^2 + \lambda x_1 x_2 + x_2^2 \), \( \lambda \neq 0 \).

\[
\frac{\partial f(x_1, x_2)}{\partial x_1} = 2x_1 + \lambda x_2.
\]

This clearly shows that the change in the global objective function with respect to \( x_1 \) is a function of \( x_1 \) and \( x_2 \). By applying the Theorem:

\[
\Delta_{\delta, x_1}[f] = [(x_1 + \delta)^2 + \lambda(x_1 + \delta)x_2 + x_2^2] - [x_1^2 + \lambda x_1 x_2 + x_2^2]
= \delta^2 + 2\delta x_1 + \lambda x_2 \delta.
\]
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 Family of Algorithms

- Linkage Identification by Non-linearity Check (LINC, LINC-R) [1]
- Differential Grouping (DG) [2]
- Global Differential Grouping (GDG) [3]
- Improved Differential Grouping (IDG) [4]
- eXtended Differential Grouping (XDG) [5]
- Graph-based Differential Grouping (gDG) [6]
- Fast Interaction Identification [7]


Shortcomings of Differential Grouping

- Cannot detect the overlapping functions.
- Slow if all interactions are to be checked.
- Requires a threshold parameter ($\epsilon$).
- Can be sensitive to the choice of the threshold parameter ($\epsilon$).
Algorithm 1: DG2

\((\Lambda, F, \tilde{f}, f_{\text{base}}, \Gamma) = \text{ISM}(f, n, \bar{x}, \bar{x})\);
\(\Theta = \text{DSM}(\Lambda, F, \tilde{f}, f_{\text{base}}, n)\);
\((k, y_1, \ldots, y_k) = \text{ConnComp}(\Theta)\);
\(x_{\text{sep}} = \{\}, \ g = 0;\)

\textbf{for} \(i = 1 \rightarrow k \ \textbf{do}\)
\[\begin{align*}
\text{if } |y_i| = 1 \text{ then} & \quad x_{\text{sep}} = x_{\text{sep}} \cup y_i; \\
\text{else} & \quad g = g + 1, \ x_g = y_i;
\end{align*}\]
Differential Grouping 2

Figure: Geometric representation of point generation in DG2 for a 3D function.

\[ x_1 \leftrightarrow x_2 : \Delta^{(1)} = f(a', b, c) - f(a, b, c), \Delta^{(2)} = f(a', b', c) - f(a, b', c) \]

\[ x_1 \leftrightarrow x_3 : \Delta^{(1)} = f(a', b, c) - f(a, b, c), \Delta^{(2)} = f(a', b, c') - f(a, b, c') \]

\[ x_2 \leftrightarrow x_3 : \Delta^{(1)} = f(a, b', c) - f(a, b, c), \Delta^{(2)} = f(a, b', c') - f(a, b, c') \]
Algorithm 2: ISM

$\Lambda = 0_n \times n$;
$F = \text{NaN}_{n \times n}$;  \hspace{1cm} // matrix of all NaNs
$\tilde{f} = \text{NaN}_{n \times 1}$;  \hspace{1cm} // vector of all NaNs
$x = x$, $f_{\text{base}} = f(x)$, $\Gamma = 1$;
$m = \frac{1}{2}(\overline{x} + \underline{x})$;

for $i = 1 \rightarrow n - 1$ do

if \text{isnan}($\tilde{f}_i$) then

$x^{(2)} = x$, $x^{(2)}_i = m$;
$\tilde{f}_i = f(x^{(2)})$, $\Gamma = \Gamma + 1$;

for $j = i + 1 \rightarrow n$ do

if \text{isnan}($\tilde{f}_j$) then

$x^{(3)} = x$, $x^{(3)}_j = m$;
$\tilde{f}_j = f(x^{(3)})$, $\Gamma = \Gamma + 1$;

$x^{(4)} = x$, $x^{(4)}_i = m$, $x^{(4)}_j = m$;
$F_{ij} = f(x^{(4)})$, $\Gamma = \Gamma + 1$;
$\Delta^{(1)} = \tilde{f}_i - f(x^{(1)})$;
$\Delta^{(2)} = F_{ij} - \tilde{f}_j$;
$\Lambda_{ij} = |\Delta^{(1)} - \Delta^{(2)}|$;
DG2: Accuracy

\[ n = \pm s \times \beta^{e-p}, \]

Figure: Non-uniform distribution of floating-point numbers for a hypothetical system \((\beta = 2, e_{\text{min}} = -1, e_{\text{max}} = 3, \text{and } p = 3)\). The vertical bars denote all the representable numbers in this system.

Theorem

If \( x \in \mathbb{R} \) lies in the range of \( F \), then

\[ fl(x) = x(1 + \delta), \quad |\delta| < \mu_M, \]

where \( \mu_M \) is called the unit roundoff, which is equal to \( \frac{1}{2} \beta^{1-p} \).
Theorem

Given a floating-point number system that satisfies IEEE 754 such that $|\delta_i| < \mu_M$. We have:

$$\prod_{i=1}^{k} (1 + \delta_i)^{e_i} = 1 + \theta_k,$$

where

$$|\theta_k| \leq \frac{\mu_M}{1 - n\mu_M} := \gamma_k, \ e_i = \pm 1,$$

provided that $k\mu_M < 1$. 

DG2: Accuracy
DG2: Accuracy

\[ \hat{\Delta}_1 = f(x) \ominus f(x') = (f(x) - f(x'))(1 + \delta_1) = \Delta^{(1)}(1 + \delta_1), \]
\[ \hat{\Delta}_2 = f(y) \ominus f(y') = (f(y) - f(y'))(1 + \delta_2) = \Delta^{(2)}(1 + \delta_2), \]

\[ \hat{\lambda} = |\hat{\Delta}_1 \ominus \hat{\Delta}_2| = |\hat{\Delta}_1 - \hat{\Delta}_2|(1 + \delta_3) \]
\[ = |f(x)(1 + \delta_1)(1 + \delta_3) - f(x')(1 + \delta_1)(1 + \delta_3)\]
\[ - f(y)(1 + \delta_2)(1 + \delta_3) + f(y')(1 + \delta_2)(1 + \delta_3)|. \]
DG2: Accuracy

$$|\lambda - \hat{\lambda}| \leq \gamma_2 \left| (f(x) - f(x')) - (f(y) - f(y')) \right|$$  \(\text{(8)}\)

$$= \gamma_2 \left| (f(x) + f(y')) - (f(y) + f(x')) \right|$$

$$\leq \gamma_2 \cdot \max \left\{ (f(x) + f(y')) , (f(y) + f(x')) \right\} := e_{\text{inf}}.$$  

Equation (8) is based on the assumption that the codomain of $f$ is non-negative, i.e., $f : \mathbb{R} \to \mathbb{R}_0^+$. A more general form for $f : \mathbb{R} \to \mathbb{R}$ is as follows:

$$e_{\text{inf}} = \gamma_2 \left( |f(x)| + |f(y')| + |f(y)| + |f(x')| \right).$$  \(\text{(9)}\)
DG2: Accuracy

\[ |f(\cdot) - \hat{f}(\cdot)| \leq \gamma \sqrt{\phi} f(\cdot) := e_{\text{sup}}. \] (10)

\[ e_{\text{sup}} = \gamma \sqrt{n} \max \{ f(\mathbf{x}), f(\mathbf{x}'), f(\mathbf{y}), f(\mathbf{y}') \} \] (11)

\[ \epsilon = \frac{\eta_0}{\eta_0 + \eta_1} e_{\text{inf}} + \frac{\eta_1}{\eta_0 + \eta_1} e_{\text{sup}}, \] (12)
Algorithm 3: $\Theta = DSM(\Lambda, F, \tilde{f}, f_{\text{base}}, n)$

\begin{align*}
\Theta &= \text{NaN}_{n \times n}; \\
\eta_1 &= \eta_2 = 0; \\
\text{for } i = 1 \rightarrow n - 1 \text{ do} & \\
& \quad \text{for } j = i + 1 \rightarrow n \text{ do} \\
& \quad \quad f_{\text{max}} = \max\{f_{\text{base}}, F_{ij}, \tilde{f}_i, \tilde{f}_j\}; \\
& \quad \quad e_{\text{inf}} = \gamma_2 \cdot \max\{f_{\text{base}} + F_{ij}, \tilde{f}_i + \tilde{f}_j\}; \\
& \quad \quad e_{\text{sup}} = \gamma\sqrt{n} \cdot f_{\text{max}}; \\
& \quad \quad \text{if } \Lambda_{ij} < e_{\text{inf}} \text{ then} \\
& \quad \quad \quad \Theta_{i,j} = 0; \eta_0 = \eta_0 + 1; \\
& \quad \quad \text{else if } \Lambda_{ij} > e_{\text{sup}} \text{ then} \\
& \quad \quad \quad \Theta_{i,j} = 1; \eta_1 = \eta_1 + 1; \\
& \quad \text{for } i = 1 \rightarrow n - 1 \text{ do} \\
& \quad \quad \text{for } j = i + 1 \rightarrow n \text{ do} \\
& \quad \quad \quad f_{\text{max}} = \max\{f_{\text{base}}, F_{ij}, \tilde{f}_i, \tilde{f}_j\}; \\
& \quad \quad \quad e_{\text{inf}} = \gamma_2 \cdot \max\{f_{\text{base}} + F_{ij}, \tilde{f}_i + \tilde{f}_j\}; \\
& \quad \quad \quad e_{\text{sup}} = \gamma\sqrt{n} \cdot f_{\text{max}}; \\
& \quad \quad \quad \text{if } \Theta_{i,j} \neq \text{NaN} \text{ then} \\
& \quad \quad \quad \quad \epsilon = \frac{\eta_0}{\eta_0 + \eta_1} \cdot e_{\text{inf}} + \frac{\eta_1}{\eta_0 + \eta_1} \cdot e_{\text{sup}}; \\
& \quad \quad \quad \quad \text{if } \Lambda_{ij} > \epsilon \text{ then} \\
& \quad \quad \quad \quad \quad \Theta_{i,j} = 1; \\
& \quad \quad \quad \text{else} \\
& \quad \quad \quad \quad \quad \Theta_{i,j} = 0; \\
\end{align*}
Direct/Indirect Interactions

**Indirect Interactions**

In an objective function $f(x)$, decision variables $x_i$ and $x_j$ interact directly (denoted by $x_i \leftrightarrow x_j$) if

$$\exists a : \left. \frac{\partial f}{\partial x_i \partial x_j} \right|_{x=a} \neq 0,$$

decision variables $x_i$ and $x_j$ interact *indirectly* if

$$\frac{\partial f}{\partial x_i \partial x_j} = 0,$$

and there exists a set of decision variables $\{x_{k1}, ..., x_{ks}\}$ such that $x_i \leftrightarrow x_{l1}, ..., x_{ks} \leftrightarrow x_j$. 
Efficiency vs Accuracy

Saving budget at the expense of missing overlaps:

- eXtended Differential Grouping [1].
- Fast Interdependency Identification [2].

**Figure:** The interaction structures represented by the two graphs cannot be distinguished by XDG.

---


Benchmark Suites

- CEC’2005 Benchmark Suite (non-modular)
- CEC’2008 LSGO Benchmark Suite (non-modular)
- CEC’2010 LSGO Benchmark Suite
- CEC’2013 LSGO Benchmark Suite
Challenges of CC

Main Questions

1. How to decompose the problem?
2. How to allocated resources?
3. How to coordinate?
The Imbalance Problem

- Non-uniform contribution of components.

**Imbalanced Functions**

\[
f(x) = \sum_{i=1}^{m} w_i f_i(x_i), \tag{13}
\]

\[
w_i = 10^{sN(0,1)},
\]
The Imbalance Problem (2)
## Contribution-Based Cooperative Co-evolution (CBCC)

### Types of CC
- **CC:** *round-robin* optimization of components.
- **CBCC:** favors components with a higher contribution.
  - Quantifies the contribution of components.
  - Optimizes the one with the highest contribution.

### How to Quantify the Contribution
- For quantification of contributions a relatively accurate decomposition is needed.
- Changes in the objective value while other components are kept constant.
(a) Round-Robin CC

(b) Contribution-Based CC

Mohammad Nabi Omidvar, Xiaodong Li
Decomposition and CC for LSGO
Contribution-Aware Algorithms

- Contribution-Based Cooperative Co-evolution (CBCC) [1], [2].
- Incremental Cooperative Coevolution [3]

---


Large-Scale Multiobjective Optimization

Large-scale multiobjective optimization is growing popularity:

- Development of a benchmark [1].
- Exploiting modularity using CC [2], [3], [4].
- Analysis of the existing benchmarks [5].


Analysis of ZDT

\[
x_1 = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 1 \\ x_2 = 1 & 0 & 1 & 1 & 1 & 1 \\ x_3 = 1 & 1 & 0 & 1 & 1 & 1 \\ x_4 = 1 & 1 & 1 & 0 & 1 & 1 \\ x_5 = 1 & 1 & 1 & 1 & 0 & 1 \\ x_6 = 1 & 1 & 1 & 1 & 1 & 0 \\
\end{pmatrix}
\]

Figure: Variable interaction structures of the $f_2$ function of ZDT test suite.
Analysis of DTLZ1-DTLZ4

![Variable interaction graphs of DTLZ1 to DTLZ4.]

**Figure:** Variable interaction graphs of DTLZ1 to DTLZ4.

**Proposition 1**

For DTLZ1 to DTLZ4, \( \forall f_i, i \in \{1, \cdots, m\} \), we divide the corresponding decision variables into two non-overlapping sets: \( x_I = (x_1, \cdots, x_\ell)^T \), \( \ell = m - 1 \) for \( i \in \{1, 2\} \) while \( \ell = m - i + 1 \) for \( i \in \{3, \cdots, m\} \); and \( x_{II} = (x_m, \cdots, x_n)^T \). All members of \( x_I \) not only interact with each other, but also interact with those of \( x_{II} \); all members of \( x_{II} \) are independent from each other.
Analysis of DTLZ5-DTLZ7

Figure: Variable interaction graphs of DTLZ5 and DTLZ6.

Proposition 2
For DTLZ5 and DTLZ6, \( \forall f_i, i \in \{1, \cdots, m\} \), we divide the corresponding decision variables into two non-overlapping sets: \( x_I = (x_1, \cdots, x_\ell)^T, \ell = m - 1 \) for \( i \in \{1, 2\} \) while \( \ell = m - i + 1 \) for \( i \in \{3, \cdots, m\} \); and \( x_{II} = (x_m, \cdots, x_n)^T \). For \( f_i \), where \( i \in \{1, \cdots, m - 1\} \), all members of \( x_I \) and \( x_{II} \) interact with each other; for \( f_m \), we have the same interaction structure as DTLZ1-DTLZ4.

Proposition 3
All objective functions of DTLZ7 are fully separable.
Some Future Directions (I)

- What if the components have overlap?
- Differential group is off-line and can be time-consuming. Is there a more efficient method?
- Do we need to get 100% accurate grouping? What is the relationship between grouping accuracy and optimality achieved by a CC algorithm?
CC for combinatorial optimization, e.g.,


However, every combinatorial optimization problem has its own characteristics. We need to investigate CC for other combinatorial optimization problems.
Learning variable interdependencies is a strength of estimation of distribution algorithms (EDAs), e.g.,


Interestingly, few work exists on scaling up EDAs.
LSGO Resources

- There is an IEEE Computational Intelligence Society (CIS) Task Force on LSGO:
- Upcoming LSGO Tutorials
  - July 2017 GECCO (Berlin, Germany).
  - November 2017 SEAL (Shenzhen, China).
- LSGO Repository: http://www.cercia.ac.uk/projects/lsgo
Acknowledgement

Thanks goes to

- Professor Xin Yao and EPSRC (grant nos. EP/K001523/1 and EP/J017515/1) for supporting this tutorial.
- Dr. Ata Kaban and Dr. Momodou L. Sanyang for allowing us to use some figures from their publications.
Thanks for your attention!