On the Analysis of Interaction between Decision Variables

Yuan Sun
ORCID 0000-0003-2911-0070

Submitted in total fulfilment of the requirements of the degree of

Doctor of Philosophy

Department of Mechanical Engineering
THE UNIVERSITY OF MELBOURNE

September 2017

Abstract

MANY real-world design and decision-making problems are characterized by the interactions between decision variables. In engineering *optimization* problems, the effects of one input variable on the performance measure may be influenced by one or several other input variables. In *classification* problems, one feature by itself may be irrelevant to the class label, however when combined with one or many feature(s), it may become highly relevant to the class label.

Identifying variable interactions is important yet non-trivial. In this thesis, the overarching goals are (1) to identify and quantify variable interactions in a given *optimization* or *classification* problem and (2) to use this information to effectively solve the problem. The methodologies used to meet these goals are a combination of *theoretical inquiry*, *computational modelling* and *experimental validation* of the proposed methods.

To identify and quantify variable interactions in a 'Black-box' Continuous Optimization Problem (BCOP), a novel Exploratory Landscape Analysis (ELA) measure is proposed based on the Maximal Information Coefficient (MIC). MIC can identify a wide range of functional relationships with high levels of accuracy. Then the proposed ELA measure is embedded into an algorithm design framework to effectively solve a BCOP. The experimental results confirm the effectiveness of the proposed ELA measure.

The high computational cost is the main limitation of the proposed ELA measure, especially in large-scale BCOPs. Therefore I propose an eXtended Differential Grouping (XDG) method, which can be used to identify variable interactions based on non-linearity detection. The XDG method decomposes a large-scale BCOP into several sub-problems considering both direct and indirect variable interactions. When XDG is embedded into a Cooperative Co-evolution framework to solve large-scale BCOPs, it generates high quality solution.

To further improve the efficiency of problem decomposition, a Recursive Differential Grouping (RDG) method is proposed, which avoids the need to check the pairwise interactions between decision variables. RDG recursively examines the interaction between a selected decision variable and the remaining variables, placing all interacting decision variables into the same group. The efficiency of RDG is shown both theoretically and empirically.

In the final stage of this thesis, I shift the focus from *optimization* problems to the closely related *classification* (more specifically feature selection) problems, by investigating the interactions between features to improve classification accuracy. I relax the assumptions made on the distribution of features and class labels, and propose a novel feature selection method which considers the Mutual Information (MI) between three features. To reduce the computational cost, the MI between three features is estimated from the pairwise MI between features. The experimental results confirm the effectiveness of the proposed method.

In summary, the interactions between decision variables have been investigated in the optimization and classification domains. Novel methods have been proposed to improve the accuracy and efficiency of identifying variable interactions in a BCOP, large-scale BCOP or feature selection problem. I then have shown that this information can be used to guide the design of search techniques to effectively solve the problem.

Declaration

This is to certify that

1. the thesis comprises only my original work towards the PhD,
2. due acknowledgement has been made in the text to all other material used,
3. the thesis is less than 100,000 words in length, exclusive of tables, maps, bibliographies and appendices.

Yuan Sun, September 2017

Acknowledgements

First, I would like to express my deepest gratitude to my supervisor Michael Kirley for his advice, guidance, encouragement, support and trust during my Ph.D candidature. Without him, this thesis would not have been completed.

I am also very grateful to my supervisor Saman K. Halgamuge and the chair of committee Adrian Pearce. Their advice, support and guidance have made this project move along much more smoothly.

I much appreciate Mario A. Muñoz from Monash University, Xiaodong Li from RMIT University and Marcus Gallagher from The University of Queensland for their expert knowledge and helpful comments.

I also want to thank my research group colleagues Wei Wang, Khalid Abdulla, Yahui Sun, Dulini Mendis, Chalini Wijetunge, Hansani Weeratunge, Pathima Hameed, Damayanthi Herath and for their comments on my research.

My thanks go to Jan May, Rajes Moodley, Kim Robinson and Carolyn Barrie for their help on the administrative matters through out my Ph.D study.

I appreciate The University of Melbourne for providing scholarships to financially support my Ph.D study, and the Melbourne School of Engineering as well as the ACM association for financially supporting my conference travel.

I would like to thank my parents Yongmei and Shiping for their endless love; my siblings Weiwei and Xiaofang for their support; as well as my son Max for brightening my day when I feel tired of writing.

Most importantly, I would like to sincerely thank my wife Jingjing for her love, support and company. It's great to share the Ph.D adventure with her.

This page intentionally left blank.

To my wife, Jingjing, and my parents, Yongmei and Shiping.

This page intentionally left blank.

Contents

This page intentionally left blank.

List of Figures

List of Tables

Nomenclature

Mathematical Notations

α	significance level of statistical tests
$\arg\max$	arguments of the maximum
$\arg\min$	arguments of the minimum
$\bar{R}_l^2$	adjusted coefficient of determination of a linear regression model
$\bar{R}_q^2$	adjusted coefficient of determination of a quadratic regression model
β	parameter used to remove noise or computational errors
$\cap$	intersection of two vectors or sets
$\cup$	combination of two vectors or sets
δx	parameter used to calculate partial derivatives
δx_i	perturbation in decision variable x_i
δ	difference between two numbers
$\emptyset$	empty set
ϵ	threshold
$\hat{\partial}$	partial sub-derivative
$\in$	belongs to
$\inf$	infimum of a set
∞	infinity
$\int$	integral
λ	non-linearity term
$\lim$	limit
$\liminf$	lower limits

log	logarithm
$\bar{\mathbb{R}}$	output space
$\mathbb{R}$	real number set
$\mathbb{R}^n$	decision space
$\mathbf{lb}$	lower bound
$\mathbf{S}$	selected feature subset
$\mathbf{ub}$	upper bound
$\mathbf{u}$,$\mathbf{u}_1$,$\mathbf{u}_2$	unit vector in the decision space
$\mathbf{X}$	decision variable or feature set
$\mathbf{x}$	decision vector in the decision space
$\mathbf{x}^*$	candidate solution in the decision space
$\mathbf{x}_1$	sub-vector of decision vector
$\mathbf{X}_1$,$\mathbf{X}_2$	decision variable subset
$\mathcal{D}$	directional derivative
$\mathcal{O}$	big O notation
$\vert\,\vert$	absolute value of a number or cardinality of a set
min	minimum of a set of values
$\notin$	not belongs to
∂	partial derivative
$\prod$	product
$\rightarrow$	mapping (sets) or approaching (numbers)
$\subset$	subset
$\sum$	summation
Θ	big Θ notation
ξ	measure of decision variable interactions
A	array
B	mathematical ball excluding the center
C	class label or co-variance matrix

D	array
f	objective function
g	objective function
H	entropy
I	mutual information
IM_d	interaction matrix only with direct interactions
J	evaluation criterion of features
L	curve in the decision space
l, l_1, l_2	arc length of a curve or real numbers
m	sample size
n	dimensionality of a problem
p	probability density function
p_r	activation probability of R operator
p_s	activation probability of S operator
R	rotation operator
r_m	Maximal Information Coefficient
r_p	Pearson Correlation Coefficient
r_s	Spearman's Rank Correlation Coefficient
U_X	set of unit vectors in the decision space of X
U_{X_1}, U_{X_2}	subset of unit vectors in the decision space
v	first-order partial sub-derivative
x_i	decision variable
X_m	candidate feature
y	fitness value or output

Acronyms / Abbreviations

ACF	Auto-Correlation Function
BE	Bit-wise Epistasis

BMDA	Bivariate Marginal Distribution Algorithm
BOA	Bayesian Optimization Algorithm
CBAVP2	Improved Correlation-Based Adaptive Variable Partitioning
CBAVP	Correlation-Based Adaptive Variable Partitioning
CC	Cooperative Co-evolution
CCVIL	Cooperative Co-evolution with Variable Interaction Learning
CEC	Congress on Evolutionary Computation
CIFE	Conditional Informative Feature Extraction
CMAESCC	A CC framework using CMA-ES as the component optimizer
CMA-ES	Covariance Matrix Adaptation - Evolutionary Strategy
CMIM	Conditional Mutual Information Maximization
D^5	Dependency Detection for Distribution Derived from Df
D	Delta Grouping
DDVI	Degree of Direct Variable Interaction
DECC	A CC framework using SaNSDE as the component optimizer
DG2	Improved Differential Grouping
DG	Differential Grouping
DIVI	Degree of Indirect Variable Interaction
DM-HDMR	A decomposition method
DVI	Degree of Variable Interaction
EC	Evolutionary Computation
ECorr	Epistasis Correlation
EDA	Estimation of Distribution Algorithm
EE	Entropic Epistasis
ELA	Exploratory Landscape Analysis
EV	Epistasis Variance
FDA	Factorized Distribution Algorithm
FII	Fast Interdependency Identification

fmGA	fast messy GA
FVIS	Fast Variable Interdependence Searching
GA	Genetic Algorithm
GDG	Global Differential Grouping
gemGA	gene expression messy GA
graphDG	graph based DG
IM	Interaction Matrix
IPOP-CMA-ES	A restart CMA-ES
jDElscop	A variant of Differential Evolution for large-scale optimization
JMI	Joint Mutual Information
KNN	K Nearest Neighbor
LIEM	Linkage Identification based on Epistasis Measures
LIMD	Linkage Identification by Non-monotonicity Detection
LINC	Linkage Identification by Non-linearity Check
LINC-R	Real coded Linkage Identification by Non-linearity Check
LLGA	Linkage-Learning Genetic Algorithm
LS-CMA-ES	Least-Square Co-variance Matrix Adaptation – Evolutionary Strategy
MA-SW-Chain	A Memetic optimizer for large-scale optimization
mDE-bES	A variant of Differential Evolution for large-scale optimization
MEE	Maximum Entropic Epistasis
mGA	messy GA
MIC	Maximal Information Coefficient
MIFS	Mutual Information Feature Selection
MI	Mutual Information
MIMIC	Mutual Information Maximization for Input Clustering
MIM	Mutual Information Maximization
MOS	Multiple Offspring Sampling
MRMR	Minimum-Redundancy Maximum-Relevance

ODT	Optimal Dependency Trees
PBIL	Population-Based Incremental Learning
PCC	Pearson Correlation Coefficient
PDF	Probability Density Function
RDG	Recursive Differential Grouping
RelaxMRMR	Relaxed Minimum-Redundancy Maximum-Relevance
RG	Random Grouping
R	The Rosenbrock local search algorithm
SPAM0.5	A version of SPAM with equal activation probabilities of S and R
SPAM_MEE	A version of SPAM with MEE measure
SPAM	Separability Prototype for Automatic Memes
SRCC	Spearman's Rank Correlation Coefficient
S	A local search method
SVIL	Statistical Variable Interdependence Learning
UMDA	Uni-variate Marginal Distribution Algorithm
XDG	eXtended Differential Grouping

Chapter 1

Introduction

MANY real-world design and decision-making problems are characterized by the interactions between decision variables [175, 90, 24, 16, 174, 44]. In engineering *optimization* problems, the objective is to improve a measure of performance – the output variable – by adjusting the values of input variables. The effects of one input variable on the performance measure may be influenced by one or several other input variables [1, 134], which makes a given problem difficult to solve [62, 191, 121, 200].

Investigating the effects of interactions between decision variables – epistasis or linkage – in Evolutionary Computation (EC) has a long history (see [56, 38, 39]). The original motivation behind much of this work, was to improve the design and effectiveness of the genetic operators. For example, specific crossover operators allow a set of interacting binary variables to be inherited together in the evolutionary process, such as the model used in the messy Genetic Algorithm (GA) [38], fast messy GA [37], and the gene expression messy GA [63]. Significantly, there is work describing the level of epistasis and problem difficulty. For example, papers describing Epistasis Variance [25] and Epistasis Correlation [144] have been used to quantify variable interactions and problem difficulty.

In the continuous optimization domain, many studies have examined the effects of variable interactions and used this to guide the evolution trajectory. For example, the class of Estimation of Distribution Algorithms (EDAs) [105] builds a probabilistic model based on promising candidate solutions and updates the model during each generation. The population for the next generation is sampled from the probabilistic model. The Cooperative Co-evolution (CC) [133] framework typically divides a large-scale optimization problem into a number of sub-problems based on the underlying variable interaction structures [120], and uses an EA to solve each sub-problem cooperatively.

In another real-world decision making problem – *classification*, the goal is to predict the class label of an object based on its features. A classification problem may be challenging to solve as (1) the number of features may be large [12], and (2) feature interactions may exist: one feature by itself may be irrelevant to the class label, however when combined with other feature(s), it may become highly relevant to the class label [200, 150]. Feature interactions have been used to guide the selection of a subset of informative features, in order to improve the classification accuracy [200, 171, 184]. For example the Joint Mutual Information (JMI) [195] method uses information theory [157] to quantify feature interactions, and selects a subset of features based on their relevance, redundancy and complementarity [184, 171]. It is important to note that feature selection problems are highly relevant to optimization problems, and many Evolutionary Algorithms (EAs) have been employed directly to solve feature selection problems [194, 193].

1.1 Research Questions

Identifying the interactions between decision variables is important yet non-trivial. The overarching goals of this thesis are to identify and quantify variable interactions in a given *optimization* or *classification* problem, and to use such information to effectively solve the problem. Two research questions guide the investigation of this thesis:

$\mathfrak{RQ}$1. **How can variable interactions be accurately and efficiently identified in a given problem?**

$\mathfrak{RQ}$2. **How can variable interactions be used to effectively solve a given problem?**

In particular, I explore the two research questions in the *continuous optimization*, *large-scale optimization* and *classification* (feature selection) domains.

1.1.1 Continuous Optimization

Continuous optimization problems are ubiquitous in the real-world, for example, engineering control problems; maximizing annual revenue; or building a parameterized model of a physical phenomenon. Two decision variables interact if they can not be optimized independently when attempting to find the optimal solution (see Section 2.1.1 for a formal definition). However, in many practical applications the level of variable interactions is unknown. These problems are typically named 'Black-box' Continuous Optimization Problems (BCOPs)[1] [116].

Exploratory Landscape Analysis (ELA) [95, 94, 96] is a technique that can be used to explore the fitness landscape [162] of an optimization problem, capturing certain characteristics (e.g. the level of variable interactions) of the problem. Once characteristics are identified, they can be used to predict problem difficulty [60, 86] or to select an appropriate algorithm to use when solving the optimization problem [142, 114, 115, 116]. A summary of the well-known ELA measures for BCOPs can be found in [114, 116]. However, despite many recent efforts, there is a lack of research quantifying the level of interactions between continuous variables and the effects such interactions have on problem difficulty. Therefore, there is a need to design an ELA measure that explores variable interactions in a given BCOP.

1.1.2 Large-scale Optimization

Large-scale (high-dimensional) BCOPs occur in domains spanning the sciences, engineering, and multidisciplinary design problems [11, 156, 80]. Such problems cannot be solved when using traditional mathematical approaches in many cases, and are very difficult to solve when using EAs. This in part may be attributed to the fact that (a) the search space of an optimization problem grows exponentially as the dimensionality increases [121]; (b) the complexity of an optimization problem usually grows as the dimensionality increases [191]; and (c) the computational cost of using some EAs (e.g., EDA) when solving high-dimensional problems is extremely high [29].

The CC framework has been used with some success when 'scaling up' EAs to tackle very high dimensional search and optimization problems. For example, CC has been applied to large-scale continuous [120], combinatorial [92], constrained [152], multi-objective

[1]In a black-box optimization problem, the analytic form of the objective function is not known. What is known is only the output given an input.

[170] and dynamic [36] optimization problems. The CC framework divides a large-scale optimization problem into a number of sub-problems, and typically uses one or several EA(s) to solve each sub-problem cooperatively [133, 119]. When optimizing each sub-problem, representatives (typically the best sub-solutions found) from the other sub-problems are combined with individuals in the optimized sub-problem, to form complete candidate solutions that can be evaluated.

A number of studies have shown that the strategy used to decompose the problem into sub-problems has a significant impact on the performance of a CC framework (e.g., [120, 19, 78]). Ideally, the interacting decision variables should be assigned to the same sub-problem, the interaction between different sub-problems should be kept to minimal. However, the existing decomposition methods are either computationally expensive (e.g., Differential Grouping [120]) or do not fully consider variable interactions (e.g., Random Grouping [197]). Therefore, there is a need to further improve the accuracy and efficiency of the decomposition methods.

1.1.3 Classification

In a *classification* problem, the goal is to predict the class label of a given object based on its features. Many real-world classification problems (e.g., micro-array data sets [12, 28, 59]) consist of hundreds and thousands of features, most of which are irrelevant and redundant to the class labels. Therefore it is essential to select a subset of informative features in order to improve the classification accuracy. Unfortunately feature selection is a difficult task partially due to feature interactions – an irrelevant feature may become highly relevant to the class label when combined with other feature(s) [200, 150].

The existing feature evaluation techniques include information theory (e.g., JMI [195]), statistical tests (e.g., Chi_Square [79]), and sparse learning (e.g., reliefF [143]). Among these techniques, information theory is a widely used and well-performed measure. See [55] for a performance comparison and [184] for a comprehensive review. However, the existing information theoretic feature selection methods mainly consider the low-order interaction between features – typically the Mutual Information (MI) between two features. A recent proposed method – Relaxed Minimum-Redundancy Maximum-Relevance

(RelaxMRMR) [185] considers the MI between three features. However, the computational cost of the RelaxRMRM method is extremely high [185]. Therefore, there is a need to improve the efficiency of feature selection methods that consider high-order interaction between features.

1.2 Methodology

The methodologies used to investigate the research questions are a combination of *theoretical inquiry*, *computational modelling* and *experimental validation*.

1. *Theoretical Inquiry:* The theoretical foundations of the proposed methods are established. I propose and rigorously prove the theorems from which the proposed methods are derived, under certain assumptions made for a given optimization or classification problem.

2. *Computational Modelling:* Computational models are designed based on the proposed theorems. The problem solving procedures (algorithms) are described in detail. The complexity of the proposed algorithms are analyzed.

3. *Experimental Validation:* The proposed algorithms are tested on benchmark / real-world optimization / classification problems. The performances of the proposed algorithms are compared with other state-of-the-art algorithms. The experimental results are analyzed statistically.

1.3 Thesis Structure

Chapter 2 presents a literature review related to the interaction between decision variables in the *optimization* domain. This chapter starts by describing fitness landscapes and problem characteristics (e.g., variable interactions). The ELA measures that can be used to explore a fitness landscape and quantify the level of variable interactions are reviewed. Then the algorithms (e.g., GAs and EDAs) that use variable interactions to guide the search are described. Finally, a taxonomy for the methods that can be used to identify and quantify variable interactions is presented, and the merits as well as the drawbacks of the methods in each category are discussed.

In Chapter 3, I propose a novel ELA measure, named *Maximum Entropic Epistasis* (MEE), based on the Maximal Information Coefficient (MIC) [141]. The MIC can accurately identify a wide range of functional relationships given a limited sample size. The proposed MEE measure can be used to identify pairwise interactions between decision variables or to quantify the level of variable interactions in a given BCOP. The efficacy of MEE is evaluated using a suite of 24 benchmark BCOPs with different levels of variable interactions. The experimental results show that MEE can accurately identify and quantify variable interactions. I then show that there is a correlation between the MEE values and the solution quality found by an EA. Finally, I present the results from simulation experiments illustrating that when MEE is embedded into an algorithm design framework, the enhanced algorithm achieves equal or better results on the benchmark problems.

The high computational cost is the main limitation of the proposed MEE measure, especially in large-scale BCOPs. In Chapter 4, I adopt the technique used in the Differential Grouping (DG) [120] method, which identifies variable interactions based on non-linearity detection [176]. I show that the DG method can only identify decision variables that interact directly. Subsequently, I propose an eXtended Differential Grouping (XDG) method that is able to correctly identify decision variables that also interact indirectly. Empirical studies show that the XDG method achieves great decomposition accuracy on all of the large-scale benchmark BCOPs investigated. Significantly, when the proposed XDG method is embedded in a CC framework to solve the large-scale BCOPs, it achieves comparable or better solution quality than the DG method.

The computational cost of the XDG method is still high. In Chapter 5, I address this limitation by proposing a Recursive Differential Grouping (RDG) method, which avoids the need to check pairwise interactions between decision variables, leading to greatly improved efficiency. RDG explicitly considers the interaction between decision variables based on non-linearity detection. It recursively examines the interaction between a selected decision variable and the remaining variables, placing all interacting decision variables into the same sub-problem. The efficacy of the RDG method is evaluated using large-scale benchmark BCOPs. Numerical simulation experiments show that RDG greatly improved the efficiency of problem decomposition in terms of time complexity. Significantly, when RDG is embedded in a CC framework, the optimization results are better than results from seven other decomposition methods.

In the final stage of this thesis, I shift the focus from *optimization* to the closely related *classification* domain, by investigating feature interactions to improve the classification accuracy. In Chapter 6, I first briefly describe the existing assumptions and information theoretic feature selection methods. Then I relax the assumptions made on the distribution of features and class labels, and derive a novel feature evaluation criterion which considers the class-conditional MI between three features. To reduce the computational cost, the three-way (class-conditional) MI is estimated from the two-way (class-conditional) MI between features. The proposed feature selection method is evaluated on a suite of 16 real-world data sets using a widely used classification algorithm. It achieves statistically significantly better accuracy when compared against four other information theoretic feature selection methods in most cases.

Chapter 7 concludes this thesis and identifies possible future directions. The main findings of each chapter are briefly summarized, and possible applications and extensions of the proposed techniques are suggested.

1.4 Contributions

In this thesis, I have proposed novel techniques which can improve both the accuracy and efficiency of identifying variable interactions in a given optimization or classification problem. Significantly, I have shown how variable interactions can be exploited in the problem solving procedures. The detailed contributions are listed as follows:

1. Two types of variable interactions in a BCOP are formally defined in Chapter 3.

2. A robust method is proposed to identify pairwise interactions between decision variables in a BCOP based on MIC, which is described in Chapter 3.

3. A novel ELA measure is proposed to quantify the level of variable interactions in a

BCOP, which is described in Chapter 3.

4. The XDG method is proposed to decompose a large-scale BCOP considering both direct and indirect variable interactions, which is presented in Chapter 4.

5. The RDG method is proposed to greatly improve the efficiency of problem decomposition, which is presented in Chapter 5.

6. A novel feature selection criterion which considers the class-conditional MI between three features is derived in Chapter 6.

7. The three-order MI is approximated from the pairwise MI between features to reduce computational cost, which is described in Chapter 6.

1.5 Related Publications

The elements of research conducted in this thesis which have been published elsewhere or are currently under review are listed as follows:

1. **Under Review**

    - Sun Y, Kirley M, Halgamuge S K. Relaxed Joint Mutual Information Criterion for Feature Selection. *Submitted to IEEE Transactions on Knowledge and Data Engineering*.

      This paper contributes to Chapter 6.

2. **Journal Publications**

    - Sun Y, Kirley M, Halgamuge S K. A Recursive Differential Grouping for Large Scale Continuous Optimization. *IEEE Transactions on Evolutionary Computation*, accepted November 2017.

      This paper contributes to Chapter 5.

    - Sun Y, Kirley M, Halgamuge S K. Quantifying Variable Interactions in Continuous Optimization Problems. *IEEE Transactions on Evolutionary Computation*, 2017, 21(2): 249-264.

This paper contributes to Chapter 3.

- Muñoz M. A, Sun Y, Kirley M, Halgamuge S K. Algorithm Selection for Black-box Continuous Optimization Problems: A Survey on Methods and Challenges. *Information Sciences*, 2015, 317, 224-245.

  This paper partially contributes to Chapter 2.

3. **Conference Publications**

   - Sun Y, Omidvar M N, Kirley M, Li X. Adaptive Threshold Parameter Estimation with Recursive Differential Grouping for Problem Decomposition. In *Proceedings of the 2018 Annual Conference on Genetic and Evolutionary Computation*. ACM, accepted March 2018.

     This paper is related to Chapter 4 and Chapter 5.

   - Sun Y, Kirley M, Li X. Cooperative Co-evolution with Online Optimizer Selection for Large-Scale Optimization. In *Proceedings of the 2018 Annual Conference on Genetic and Evolutionary Computation*. ACM, accepted March 2018.

     This paper is related to Chapter 4 and Chapter 5.

   - Sun Y, Kirley M, Halgamuge S K. A Memetic Cooperative Co-evolution Model for Large Scale Optimization Problems. *Australasian Conference on Artificial Life and Computational Intelligence.* Springer, Cham, 2017: 291-300.

     This paper is related to Chapter 4 and Chapter 5.

   - Sun Y, Kirley M, Halgamuge S K. Extended Differential Grouping for Large Scale Global Optimization with Direct and Indirect Variable Interactions. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*. ACM, 2015: 313-320.

     This paper contributes to Chapter 4.

   - Sun Y, Kirley M, Halgamuge S K. On the Selection of Decomposition Methods for Large Scale Fully Non-separable Problems. In *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*. ACM, 2015: 1213-1216.

     This paper is related to Chapter 4 and Chapter 5.

- Sun Y, Kirley M, Halgamuge S K, Muñoz M A. On the Selection of Fitness Landscape Analysis Metrics for Continuous Optimization Problems. In *Proceedings of the 7th International Conference on Information and Automation for Sustainability.* IEEE, 2014: 1-6.

  This paper partially contributes to Chapter 2.

Chapter 2

Background and Related Work

EPISTASIS, originally defined by William Bateson [9], is used to describe "a phenomenon whereby the effects of a given gene on a biological trait are masked or enhanced by one or more other genes" [101]. Such phenomenon is due to the interaction between genes, leading to a non-linear effect on the phenotype. Table 2.1 shows an example of gene interaction, where two genes 'co-express' to determine the coat color of Labrador retrievers [169]. Epistasis or gene interaction has a huge impact on the structure and evolution of genetic systems [190, 132, 130].

Epistasis has been generalized from genetics to the Evolutionary Computation (EC) domain, where it is used to describe the non-linear effects between decision variables. The presence of epistasis may dramatically complicate the shape of fitness landscapes, making an optimization problem challenging to solve [26, 144]. There have been some attempts to quantify the level of epistasis, which is then used as an indicator of "problem difficulty" (see Section 2.1 for a detailed description).

Variable interaction (epistasis) has been used to guide the design of effective search and optimization algorithms. An application is to generate tight building blocks for Genetic Algorithms (GAs) – placing interacting genes (decision variables) close on a chromosome. The model building algorithms encapsulate variable interactions in a probability model and update this model to guide the search process e.g., Estimation of Distribution Algorithms (EDAs) [7, 107, 105, 104] and Co-variance Matrix Adaptation – Evolutionary Strategy (CMA-ES) [46, 47, 45]. Recently, there has been significant interest in using variable interactions to decompose a large-scale optimization problem into components that are solved individually by Cooperative Co-evolution (CC) algorithms [120].

Table 2.1: An example of epistasis in genetics: the coat-color inheritance in Labrador retrievers. At the pigment gene, the two alleles 'A' and 'a' determine the black and brown colors respectively. The allele 'B' from the other gene allows color deposition in the coat, while 'b/b' prevents color deposition, generating the gold coat color [169].

Genes	AB	aB	Ab	ab
AB	black	black	black	black
aB	black	brown	black	brown
Ab	black	black	gold	gold
ab	black	brown	gold	gold

In this chapter, I present a comprehensive literature review of the methods that can be used to explore variable interactions across the EC domain. Although there have been some reviews in each individual field (e.g., a survey of methods for GAs [21] or for EDAs [52]), there is a paucity of research in the comparison between the methods in different fields. This chapter fills this gap by describing the similarities and differences between these methods. A taxonomy of these methods is suggested, and merits as well as drawbacks of the methods in each category are discussed.

The remainder of this chapter is organized as follows. In Section 2.1, before describing variable interactions in detail, I take a step back and introduce fitness landscapes, which encapsulate all of problem characteristics. The Exploratory Landscape Analysis (ELA) measures that can be used to explore fitness landscapes and quantify the level of variable interactions are also reviewed in Section 2.1. Then the algorithms that utilize variable interactions to guide the search process are presented in Section 2.2. Finally, I conclude this chapter by presenting a taxonomy for the methods that exploring variable interactions in Section 2.3.

It is important to note that, in this chapter, I only present the background and related work of variable interactions in the optimization domain. In each of the remaining chapters (excluding Chapter 7), a review of the most relevant work will be presented. In Chapter 3, I introduce the Maximal Information Coefficient [141]. A detailed description of the Differential Grouping (DG) [120] method is presented in Chapter 4. In Chapter 5, I briefly review the algorithms for large-scale optimization excluding CC. In Chapter 6, representative information theoretic feature selection methods are described.

2.1 Fitness Landscape Analysis

This section describes fitness landscapes and ELA measures that can be used to explore a fitness landscape and quantify variable interactions.

2.1.1 Fitness Landscape

A fitness landscape [192, 91] is the structure of mapping from input values (decision variables) to output values (fitness). Taking the two dimensional optimization problem $f(\mathbf{x}) = sin(x_1)sin(x_2) + sin(x_1)/5$ as an example, the fitness landscape is a three dimensional surface consisting of ridges, valleys and basins as illustrated in Figure 2.1. The basins of the fitness landscape are shown in Figure 2.2. This fitness landscape consists of two peaks and two valleys. The global minimum lies in bottom right valley, while the global maximum is on the bottom left peak in Figure 2.2. Formally, the fitness landscape is defined as follows [162]:

Definition 2.1. *The fitness landscape L of an objective function f: $\mathbb{X} \rightarrow \mathbb{R}$ is defined as the tuple $L = (\mathbb{X}; f; d)$, where $\mathbb{X}$ is the decision space and d is a distance metric that quantifies the similarity between candidates.*

In continuous optimization, d is often the Euclidean distance. The fitness landscape resulting from one coding strategy / distance metric (d) may be significantly different from the one using another coding strategy [131].

In a recent survey paper that I co-authored, a detailed analysis of landscape features (e.g. separability, modality, basins of attraction, ruggedness, smoothness, neutrality) related to problem difficulty in continuous optimization problems is presented [116]. Among these landscape features, separability and modality are of key importance. Separability refers to variable interactions, while modality refers to global or local optima in an optimization problem. In the following, I briefly introduce modality before describing separability in detail.

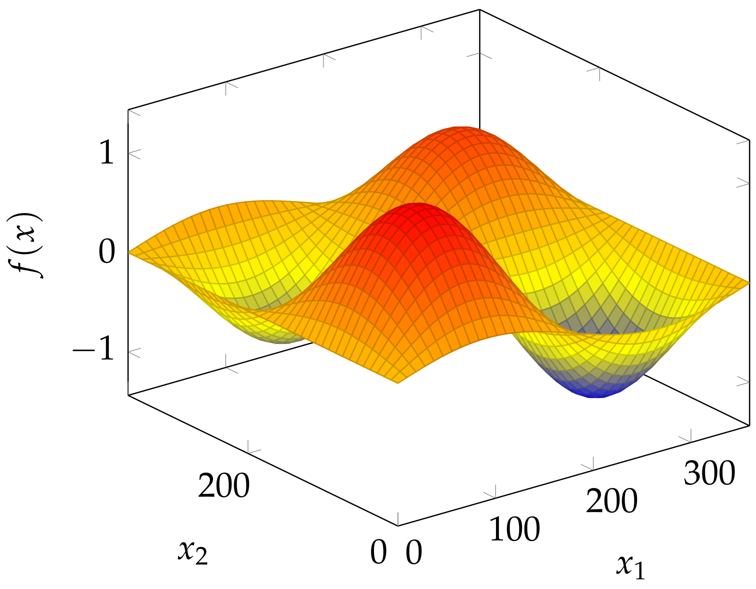


Figure 2.1: The fitness landscape (surface plot) of the objective function: $f(\mathbf{x}) = sin(x_1)sin(x_2) + sin(x_1)/5$. The fitness landscape is a mapping from the input values (decision variables x_1 and x_2) to the fitness values (output variable $f(\mathbf{x})$).

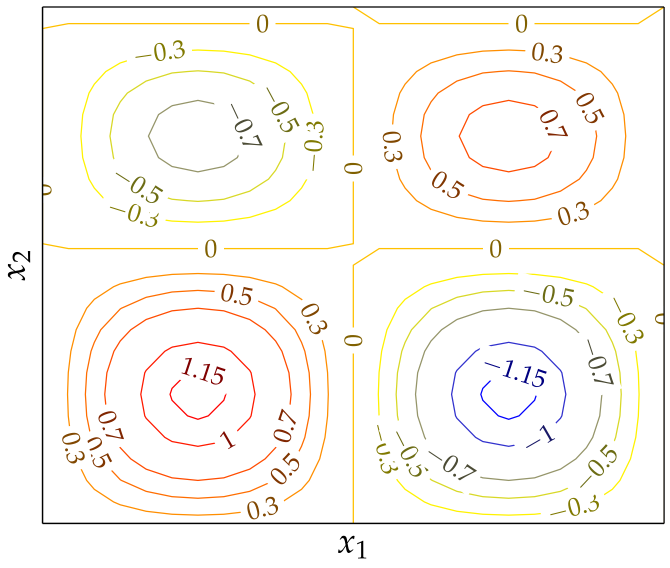


Figure 2.2: The basins of attractions (contour plot) of the objective function $f(\mathbf{x}) = sin(x_1)sin(x_2) + sin(x_1)/5$. This fitness landscape consists of two peaks and two valleys. The global minimum lies in bottom right valley, while the global maximum is on the bottom left peak.

Modality

Without loss of generality, only minimization problems are considered in this chapter. The definitions of global and local optima are described as follows [116]:

Definition 2.2. *In an objective function f: $\mathbb{R}^n \to \mathbb{R}$, a candidate solution $\mathbf{x}_o$ is a global optimum if for all $\mathbf{x} \in \mathbb{R}^n$, $f(\mathbf{x}_o) \leq f(\mathbf{x})$. A candidate solution $\mathbf{x}_l$ is a local optimum if there exists a $\delta > 0$ such that for all $x \in B(\mathbf{x}_l, \delta)$, $f(\mathbf{x}_l) \leq f(\mathbf{x})$, where $B(\mathbf{x}_l, \delta)$ is the ball of center $\mathbf{x}_l$ and radius δ (excluding $\mathbf{x}_l$); If $f(\mathbf{x}_l) < f(\mathbf{x})$, for all $x \in B(\mathbf{x}_l, \delta)$, $\mathbf{x}_l$ becomes a strict local optimum.*

An optimization problem is unimodal if it only has one optimum, and is multi-modal if it has more than one optimum. Several landscape features related to modality may influence the search difficulty, e.g. the number of local optima, the size of the optimum attraction basin, the average shortest path between local optima, ruggedness, smoothness and neutrality [116, 183, 17, 88, 53].

Separability

An objective function is defined as separable if the global optimum can be obtained by optimizing each decision variable individually [191, 121]:

Definition 2.3. *An objective function f: $\mathbb{R}^n \to \mathbb{R}$ is fully separable if*

$$\arg\min_{\mathbf{x}} f(\mathbf{x}) = \Big(\arg\min_{x_1} f(\mathbf{x}), \arg\min_{x_2} f(\mathbf{x}), \cdots, \arg\min_{x_n} f(\mathbf{x})\Big), \tag{2.1}$$

where $\mathbf{x} = (x_1, x_2, \cdots, x_n)$ is the decision vector of n dimensions.

A decision variable is defined as separable (not interacting with any other decision variables) if it can be optimized separately from other decision variables to obtain the global optimum [121].

Definition 2.4. *In an objective function f: $\mathbb{R}^n \to \mathbb{R}$, a decision variable x_i is separable if*

$$\arg\min_{\mathbf{x}} f(\mathbf{x}) = \arg\min_{x_i} f(\mathbf{x}) \cup \arg\min_{\forall x_j \in \mathbf{x}, j \neq i} f(\mathbf{x}), \tag{2.2}$$

where $\cup$ represents the operator to combine the two sub-vectors as a whole decision vector $\mathbf{x}$.

Definition 2.5. *An objective function f: $\mathbb{R}^n \to \mathbb{R}$ is partially separable with m independent components if*

$$\arg\min_{\mathbf{x}} f(\mathbf{x}) = \Big(\arg\min_{\mathbf{x}_1} f(\mathbf{x}), \arg\min_{\mathbf{x}_2} f(\mathbf{x}), \cdots, \arg\min_{\mathbf{x}_m} f(\mathbf{x})\Big), \tag{2.3}$$

where $\mathbf{x}_1$, $\mathbf{x}_2$, $\cdots$, $\mathbf{x}_m$ are mutually exclusive sub-vectors of $\mathbf{x}$, $1 < m < n$.

It is important to note that Definition 2.5 is a more general definition of separability. If $m = n$, Definition 2.5 will decay to Definition 2.3, and the objective function becomes fully separable. When $m = 1$, all decision variables interact, therefore the objective function is fully non-separable.

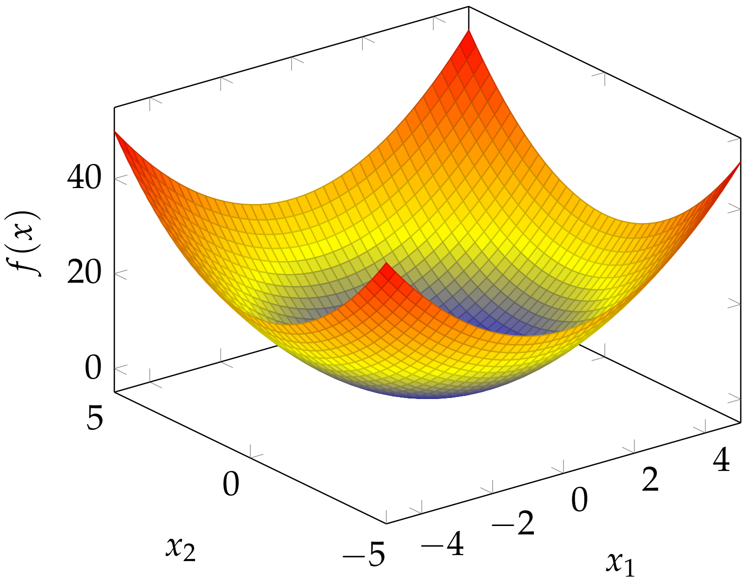
Definition 2.6. *An objective function f: $\mathbb{R}^n \to \mathbb{R}$ is partially additively separable if it can be expressed as the following general form:*

$$f(\mathbf{x}) = \sum_{i=1}^{m} f_i(\mathbf{x}_i), \tag{2.4}$$

where $\mathbf{x}_1$, $\mathbf{x}_2$, $\cdots$, $\mathbf{x}_m$ (mutually exclusive sub-vectors of $\mathbf{x}$) are decision vectors of function f_i respectively, and m is the number of components.

Figure 2.3 shows the surface and contour plots of a separable objective function $f(\mathbf{x}) := x_1^2 + x_2^2$. The effects of x_2 on the fitness values are not influenced by the value of x_1. When we fix the value of x_2 and randomly select two points A and B with different x_1 values, the fitness changes are the same if we search in the same direction of x_2, as shown in Figure 2.3. Therefore, it is possible to optimize the decision variables x_1 and x_2 individually to obtain the global optimum. However in a non-separable objective function (e.g., $f(\mathbf{x}) := 5(\frac{x_1}{\sqrt{2}} - \frac{x_2}{\sqrt{2}})^2 + (\frac{x_1}{\sqrt{2}} + \frac{x_2}{\sqrt{2}})^2$), the decision variables x_1 and x_2 must be optimized simultaneously in order to obtain the global optimum. The reason for this is that at different points A and B with the same x_2 value but different x_1 values, the fitness changes can be much different if we search in the same direction of x_2, as shown in Figure 2.4.

It is important to note that the landscape features may influence each other. Take the NK model [64] as an example, where N is the number of decision variables and K



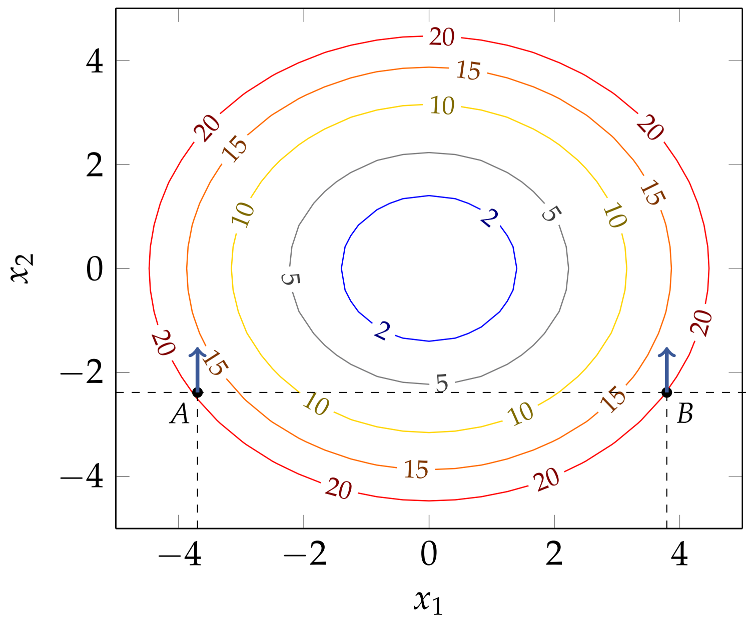
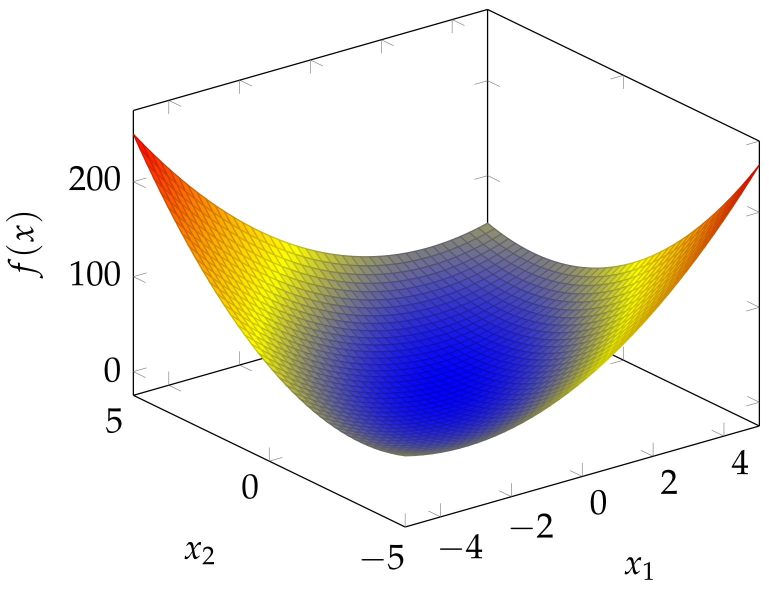


Figure 2.3: The surface (top figure) and contour (bottom figure) plots of the separable objective function $f(\mathbf{x}) := x_1^2 + x_2^2$. In the top figure, the horizontal axis represents the decision variables x_1 and x_2, while the vertical axis represents the fitness value. In the bottom figure, the candidate solutions on the same curve have equal fitness value. When we fix the value of x_2, and randomly select two points (e.g., A and B), the fitness value will decrease if we increase the value of x_2 at both points.



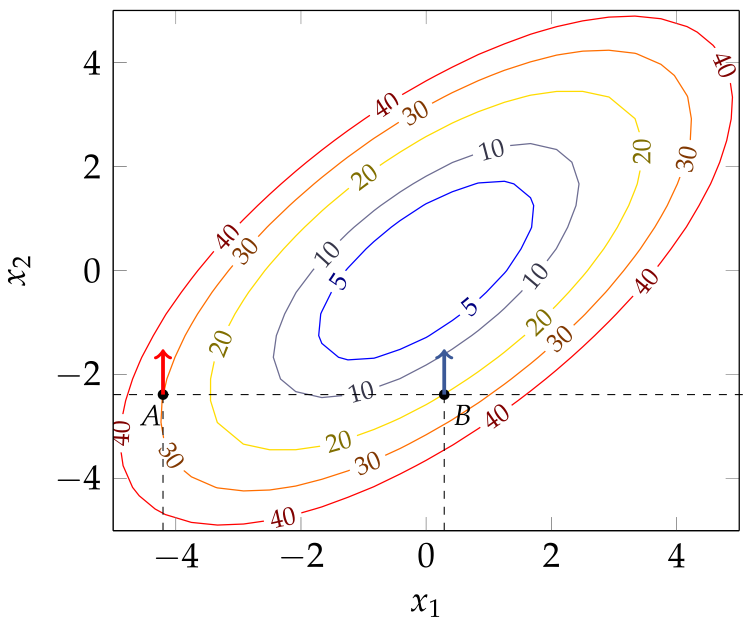


Figure 2.4: The surface (top figure) and contour (bottom figure) plots of the non-separable objective function $f(\mathbf{x}) := 5(\frac{x_1}{\sqrt{2}} - \frac{x_2}{\sqrt{2}})^2 + (\frac{x_1}{\sqrt{2}} + \frac{x_2}{\sqrt{2}})^2$. In the top figure, the horizontal axis represents the decision variables x_1 and x_2, while the vertical axis represents the fitness value. In the bottom figure, the candidate solutions on the same curve have equal fitness value. When we fix the value of x_2, and select two points A and B, the fitness value will increase if we increase the value of x_2 at point A, while decrease at point B.

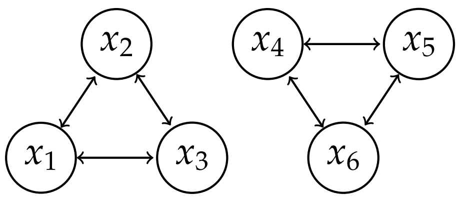


Figure 2.5: An example of the *NK* model. There are in total six decision variables ($N = 6$); each decision variable interact with two other decision variables ($K = 2$).

is the level of variable interactions or epistasis (Figure 2.5).[1] The modality of the *NK* fitness landscape can be tuned by the level of variable interactions. As K increases from 0 to $N - 1$, the fitness landscape becomes more rugged [64, 189]. Searching for a good candidate solution in a highly rugged fitness landscape is difficult, as there is a high probability that the search algorithm is stuck in a local optimum with low fitness [4].

2.1.2 Measures Quantifying Variable Interactions

ELA measures are analytical, approximated and non-predictive methods designed to explore fitness landscapes and to describe certain problem characteristics [145, 54, 86, 117, 95, 94, 96]. In related work, I have attempted to map the ELA measures used to capture the problem characteristics [166]. However, other ELA measures such as Fitness Distance Correlation [60] and Dispersion Metric [82] should be examined to paint a more coherent and complete picture. In the following I describe the ELA measures related to variable interactions in detail, while some of the other ELA measures are briefly summarized in Table 2.2. Note that the review in Table 2.2 is by no means complete. My goal is not to present a comprehensive review of ELA measures. Therefore, I only select some well-known ELA measures and describe them briefly.

The Auto-Correlation Function (ACF) [188] is one of the earliest ELA measures. It calculates the auto-correlation of a series of fitnesses obtained from a random walk over a fitness landscape:

$$ACF_\tau = \frac{1}{\hat{\sigma}_y^2 (m - \tau)} \sum_{i=1}^{m-\tau} (y_i - \bar{y})(y_{i+\tau} - \bar{y}), \tag{2.5}$$

[1]The *NK* model was originally proposed for binary decision variables. The reason I use the *NK* model as an example is that the level of epistasis can be tuned in the *NK* model.

Table 2.2: A summary of the selected ELA measures. My goal is not to give a comprehensive review of ELA measures. Therefore, I only select some well-known ELA measures and briefly describe them.

ELA measures	Description
Fitness Distance Correlation [60]	Measures the Pearson correlation between the location and fitness of the candidate solutions.
Distribution of Fitness [148]	Measures the status of probability density function of fitness values e.g., skewness, kurtosis and the number of peaks.
Length Scale [103]	Measures the ratio of fitness changes to the difference between candidate solutions.
Fitness Cloud [182]	Measures the graph of the fitness values of parents verses the fitness values of their offsprings.
Negative Slope Coefficient [180]	Measures the summation of negative slopes in a partitioned fitness cloud.
Dispersion metrics [82]	Measures the average distance between pairwise high quality candidate solutions.
Information Characteristics [181]	Measures the structure of fitness landscapes based on information theory, e.g., Information Content, Partial Information Content, and Information Stability.

where τ is the number of delays over which the auto-correlation is calculated, $\bar{y}$ is the mean fitness, and $\hat{\sigma}_y^2$ is the fitness variance of the samples. For continuous optimization problems, the ACF has been defined as [89]:

$$ACF\left(r\right) = \frac{1}{\sigma_y^2} \frac{1}{|\mathcal{N}_r|} \sum_{\left(\mathbf{x}_i, \mathbf{x}_j\right) \in \mathcal{N}_r} \left(y_i y_j - \left(y_i + y_j\right) \bar{y} + \bar{y}^2\right), \tag{2.6}$$

where $\mathcal{N}_r\left(\mathbf{x}\right) = \{\mathbf{x}_i : \|\mathbf{x}_i - \mathbf{x}\| = r\}$. However, it has been shown that the ACF measure is not always reliable when used to predict problem difficulty, as ACF only provides a local view of fitness landscapes [131].

The Epistasis Variance (EV) [26] was originally proposed to measure variable interactions in binary strings (binary decision variables) by approximating a linear model of a fitness function. It can be calculated as follows [139]:

$$\eta = \frac{\sum\limits_{\mathbf{x} \in Pop} \left(f(\mathbf{x}) - \xi(\mathbf{x})\right)^2}{\sum\limits_{\mathbf{x} \in Pop} \left(f(\mathbf{x}) - \bar{f}\right)^2}, \tag{2.7}$$

where ξ is the linear approximation of f, and $\bar{f} = \sum\limits_{\mathbf{x} \in Pop} f(\mathbf{x})$ is the mean fitness value. The EV measure is sensitive to the magnitude of fitness values. Subsequently, Manela and Campbell normalize it by the variance of fitness values, making it possible to compare the difficulty of different problems [87].

The Epistasis Correlation (ECorr) [144] measure uses Walsh sums to quantify the level of epistasis in a binary string (binary decision variables). It is defined as [139]

$$\kappa = \frac{\sum\limits_{\mathbf{x} \in Pop} \left(f(\mathbf{x}) - \bar{f}\right)\left(\xi(\mathbf{x}) - \bar{\xi}\right)}{\sqrt{\sum\limits_{\mathbf{x} \in Pop} \left(f(\mathbf{x}) - \bar{f}\right)^2 \sum\limits_{\mathbf{x} \in Pop} \left(\xi(\mathbf{x}) - \bar{\xi}\right)^2}}, \tag{2.8}$$

where ξ is the linear approximation of f, and $\bar{f}$ and $\bar{\xi}$ are the mean values of f and ξ over all populations. However, the ECorr and EV measures have been criticized for only measuring the absence of epistasis [117, 138, 139]. Furthermore, a large sample size is required for problems with high epistasis.

The Bit-wise Epistasis (BE) [32] is a more comprehensive technique, investigating the independence of each binary decision variable. The BE of decision variable x_i is defined as the variance of fitness differences at x_i:

$$\sigma_i^2 = \frac{1}{2^{n-1}} \sum_{\mathbf{x} \in Pop} (\Delta\bar{f}_i - \Delta f_i(\mathbf{x}))^2, \tag{2.9}$$

where $\Delta f_i(\mathbf{x}) = f(\cdots, x_i, \cdots) - f(\cdots, 1 - x_i, \cdots)$ is the fitness difference and $\Delta\bar{f}_i$ is the mean of fitness differences at x_i. The BE can be normalized by the fitness variance of the samples in order to compare between different problems. Unfortunately, this technique can only identify independent decision variables, rather than identifying pair-wise interactions between decision variables.

A more recently proposed measure, Entropic Epistasis (EE) [154], identifies the interaction between decision variables based on Mutual Information (MI). The EE ($\mathcal{E}(V)$) of a non-empty decision variable subset V is defined as follows:

$$\mathcal{E}(V) = \frac{I(X_V;Y) - \sum\limits_{v \in V} I(X_v;Y)}{I(X_V;Y)}, \tag{2.10}$$

where X denotes the decision variable(s), Y denotes the objective function value, and $I(X;Y)$ is the MI between X and Y. It is important to note that EE was originally proposed for combinatorial optimization problems. The *k-d* partitioning method can be used to estimate the entropy for multi-dimensional continuous variables [164]. However, the computational complexity of EE grows exponentially with the dimensionality n, as the sample size used to estimate n-dimensional entropy should be greater than 2^n [164].

The Meta-Model [94] method builds a linear or quadratic regression model with or without variable interactions based on m samples $\{(\mathbf{x}_i, y_i), 1 \leq i \leq m\}$. The adjusted coefficient of determination ($\bar{R}^2$) is employed as an indicator for model accuracy, which is defined as:

$$\bar{R}^2 = 1 - \frac{(m-1)\sum\limits_{i=1}^{m}(\hat{y}_i - \bar{y})^2}{(m-p-1)\sum\limits_{i=1}^{m}(y_i - \bar{y})^2}, \tag{2.11}$$

where $\hat{y}_i$ is the corresponding estimation made by the regression model: $\hat{y}_i = f(\mathbf{x}_i)$; $\bar{y}$ is the mean of y_i; p is the degree of the polynomial regression model (linear: $p = 1$, quadratic: $p = 2$). Green [42] suggested that the sample size m should be greater than $50 + 8n$, where n is the number of decision variables.

In genetics, magnitude epistasis and sign epistasis are two measures of gene interactions in empirical fitness landscapes [190, 132]. For magnitude epistasis, the fitness change induced by a mutation of a gene differs in magnitude but not in sign for different genetic background (different values of other genes). While for sign epistasis, the fitness effect changes in sign. Therefore, finding a global optimum in the fitness landscapes exhibiting sign epistasis is difficult. This in part is attributed to the fact that natural selection will discard some evolutionary paths even if they eventually lead to a global optimum.

2.2 Algorithms Utilizing Variable Interactions

This section presents a review of algorithms that utilize variable interactions to guide the search process.

2.2.1 Genetic Algorithms

In GA research, the decision variables are usually encoded on a chromosome, and each chromosome represents a candidate solution. During the evolutionary process, genetic operators (e.g., crossover and mutation) are used to manipulate the chromosomes to generate new (hopefully better) candidate solutions. The foundation of GA is based on the building blocks hypothesis; that is, optimal candidate solution can emerge by juxtaposing the relatively short, high-performance schemata (building blocks) [33, 98].

The original motivation of investigating gene linkage or epistasis is to achieve a good coding scheme for GAs [56, 38, 39, 40, 21]. That is to place tightly linked genes (strongly interacting decision variables) close in a chromosome so that the linked genes can be inherited together by offsprings in the evolutionary process. Taking an optimization problem with six decision variables as an example, where decision variables x_1, x_2, x_3 interact, and x_4, x_5, x_6 interact, three possible different coding schemes are shown in Figure 2.6. When applying the one-point crossover on chromosomes, there is a high probability that the interacting decision variables encoded by scheme 1 will be inherited together. For coding scheme 2, the interacting decision variables $\{x_4, x_5, x_6\}$ are likely to be inherited together, while $\{x_1, x_2, x_3\}$ can be easily split. For coding scheme 3, the interacting decision variables will always be disconnected.

There has been some experimental evidence suggesting that the success of GAs relies on tight coding of decision variables on chromosomes [38, 39, 40, 21]. When interacting decision variables are correctly placed into the same building block, GAs can perform well even for hard optimization problems. For example, in the experiments conducted by Goldberg *et. al.* [38], the simple GA with three coding schemes: tightly ordering, loosely ordering, and randomly ordering, are tested on a benchmark problem. The experimental results show that the simple GA with tight coding performs well, while the simple GA

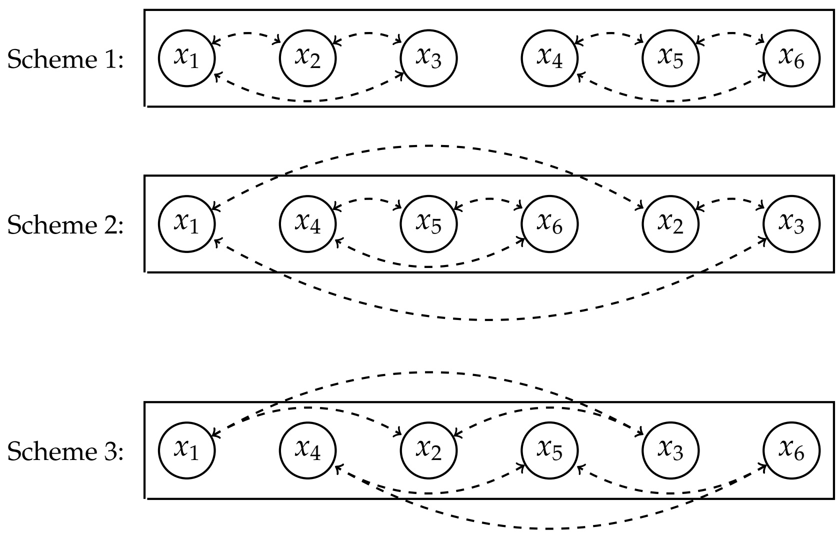

Scheme 1:
x_1 x_2 x_3 x_4 x_5 x_6
Scheme 2:
x_1 x_4 x_5 x_6 x_2 x_3
Scheme 3:
x_1 x_4 x_2 x_5 x_3 x_6

Figure 2.6: Three possible coding schemes for an optimization problem with six decision variables. Decision variables $\{x_1, x_2, x_3\}$ interact, and $\{x_4, x_5, x_6\}$ interact.

with randomly ordering can easily fail.

There has been a number of methods proposed to investigate variable interactions, in order to generate a tight coding for GA building blocks. These methods can be roughly classified into two categories: 1) The adaptive coding method evolves different coding schemes in the evolutionary process, and the tight coding schemes emerge; and 2) The fixed coding method identifies variable interactions and generates a fixed coding scheme based on that. In the following, I describe some of the representative methods in each category.

Adaptive Coding

The inversion operator is one of the first attempts to promote tight coding of decision variables in the evolutionary process [56, 41]. It inverts the order of decision variables that belong to a substring. The strings that are tightly coded have a better chance to survive when the selection operator is applied. However, the main issue of the inversion operator is that the convergence rate of generating tight coding is very slow [21].

The messy GA (mGA) [38] is another pioneer work in building blocks for GAs [21]. In contrast to the simple GAs, which process fixed coding with fixed length strings, the mGA can process variant length strings that may be either under-specified or over-specified with respect to the number of decision variables investigated. For example, for a problem with three decision variables, the binary strings can be $\{(x_1, 0), (x_2, 1)\}$ or $\{(x_1, 0), (x_3, 0), (x_2, 1), (x_2, 0)\}$. In the under-specified string $\{(x_1, 0), (x_2, 1)\}$, where there is no value for the decision variable x_3, the 'partial string partial evaluation' approach is employed to evaluate it. On the other hand, to evaluate the over-specified string $\{(x_1, 0), (x_3, 0), (x_2, 1), (x_2, 0)\}$, the string is scanned from left to right and the first value of x_2 encountered is kept. The specifically designed operators – cut, splice, and mutation – are used to manipulate the strings. By using these operators, the relatively short and good building blocks can be combined to generate more complex strings, and the best building blocks are more likely to stand out in the evolutionary process. Experimental results have shown that the mGA outperforms the simple GAs, and has the ability to solve difficult problems to optimality [38].

The main limitation of the mGA is that it requires a large initial population, thus is computationally expensive. Subsequently, the fast messy GA (fmGA) [37] is proposed to address this issue. In contrast to its predecessor, which uses partially enumerative initialization, the fmGA employs the probabilistically complete initialization. Moreover, the fmGA algorithm only generates high-order strings and performs building blocks filtering via selection and random gene deletion. By doing this, the fmGA algorithm can potentially find an optimal solution in less than sub-quadratic time in terms of the dimensionality.

The Linkage-Learning Genetic Algorithm (LLGA) [49, 51] is another method that evolves gene linkage in the evolutionary process. It uses a dynamic representation of gene positions and values, coded as (position, value) pairs, like the one used in mGA. The 'exchange' operator, similar to the two-point crossover, is employed to manipulate chromosomes and to generate new solutions with different coding structures. A probability measure is used to quantify the level of linkage within a chromosome. It has been shown that when the optimal building block is the donor of genetic material, the distribution of individuals is *skewed* towards those with higher linkages. That is the individuals with high linkage have a better chance to survive under the exchange operator.

When the optimal building block is the recipient of donated material, the distribution of individuals is *shifted* to those with higher linkages.

Fixed Coding

The Linkage Identification by Non-linearity Check (LINC) [110] method was originally proposed to identify pairwise interactions between binary decision variables by perturbation. Two decision variables are said to be interacting if the changes of fitness values by perturbing the two decision variables are not additive. Formally, the change of fitness values by perturbing the binary decision variable x_i or x_j is defined as follows:

$$\Delta f_i = f(\cdots, \bar{x}_i, \cdots) - f(\cdots, x_i, \cdots), \tag{2.12}$$

$$\Delta f_j = f(\cdots, \bar{x}_j, \cdots) - f(\cdots, x_j, \cdots), \tag{2.13}$$

where $\bar{x}_i = 1 - x_i$ and $\bar{x}_j = 1 - x_j$. The change of fitness values by perturbing x_i and x_j simultaneously is defined as:

$$\Delta f_{ij} = f(\cdots, \bar{x}_i, \cdots, \bar{x}_j, \cdots) - f(\cdots, x_i, \cdots, x_j, \cdots). \tag{2.14}$$

If $\Delta f_{ij} = \Delta f_i + \Delta f_j$, x_i and x_j are separable; otherwise, they interact with each other. The LINC method has been successfully applied to identify variable interactions (or linkage groups) for GAs [109, 111].

The Linkage Identification by Non-monotonicity Detection (LIMD) [112] is another perturbation based method, which was also originally proposed for binary decision variables. The LIMD method has been shown to be equivalent to LINC with allowable non-linearity [112]. Two decision variables x_i and x_j interact if there exists a candidate solution such that at least one of the following statements is not true:

1. If $f_i(\mathbf{x}) > f(\mathbf{x})$ and $f_j(\mathbf{x}) > f(\mathbf{x})$, then $f_{ij}(\mathbf{x}) > f_i(\mathbf{x})$ and $f_{ij}(\mathbf{x}) > f_j(\mathbf{x})$;
2. If $f_i(\mathbf{x}) < f(\mathbf{x})$ and $f_j(\mathbf{x}) < f(\mathbf{x})$, then $f_{ij}(\mathbf{x}) < f_i(\mathbf{x})$ and $f_{ij}(\mathbf{x}) < f_j(\mathbf{x})$,

where $f_i(\mathbf{x}) = f(\cdots, \bar{x}_i, \cdots)$, $f_j(\mathbf{x}) = f(\cdots, \bar{x}_j, \cdots)$ and $f_{ij}(\mathbf{x}) = f(\cdots, \bar{x}_i, \cdots, \bar{x}_j, \cdots)$.

It states that two decision variables x_i and x_j are separable only if the changes of fitness values by perturbing x_i and x_j are either monotonely increasing: $f_{ij}(\mathbf{x}) > f_i(\mathbf{x}) > f(\mathbf{x})$, $f_{ij}(\mathbf{x}) > f_j(\mathbf{x}) > f(\mathbf{x})$, or monotonely decreasing: $f_{ij}(\mathbf{x}) < f_i(\mathbf{x}) < f(\mathbf{x})$, $f_{ij}(\mathbf{x}) < f_j(\mathbf{x}) < f(\mathbf{x})$. If any non-monotonicity of the fitness changes is detected, x_i and x_j interact with each other.

A limitation of the LIMD method is that it is unable to handle overlapping problems (e.g., Rosenbrock function [73]). Variable interactions in an overlapping problem are generally more difficult to identify. An extension of the LIMD method has been proposed to address this limitation by using the tightness detection procedure [113]. Firstly, the LIMD method is used to identify the linkage groups by perturbing pairs of binary decision variables. Then the tightness detection procedure starts to calculate the tightness of linkage between pairs of decision variables in the linkage groups. The linkage groups can be identified accurately for overlapping problems by removing the binary decision variables with weak tightness from the linkage groups.

Both the LINC and LIMD methods can not quantify the level of variable interactions in a given problem. Subsequently, the Linkage Identification based on Epistasis Measure (LIEM) [108] method is proposed, which quantifies the degree of variable interactions based on an epistasis measure:

$$e_{ij} = \max_{\mathbf{x} \in Pop} |\Delta f_{ij}(\mathbf{x}) - \Delta f_i(\mathbf{x}) - \Delta f_j(\mathbf{x})|, \tag{2.15}$$

where Pop is a population of candidate solutions, and $\Delta f_i(\mathbf{x})$, $\Delta f_j(\mathbf{x})$, $\Delta f_{ij}(\mathbf{x})$ are defined in Eq. (2.12), Eq. (2.13) and Eq. (2.14) respectively. If $e_{ij} > 0$, decision variables x_i and x_j interact; if $e_{ij} = 0$, they are separable. The magnitude of e_{ij} indicates how strongly the decision variables x_i and x_j interact. To obtain the linkage group for a decision variable x_i, the corresponding e_{ij} ($j = 1 \cdots n$) values are sorted, and the decision variables with top k e_{ij} values are selected and placed into the same group as x_i.

An extension of the LINC method, named LINC-R [176], identifies the interaction between continuous decision variables (with real numbers) by random perturbation:

$$\Delta f_i = f(x_i + \Delta x_i, x_j) - f(x_i, x_j), \tag{2.16}$$

$$\Delta f_j = f(x_i, x_j + \Delta x_j) - f(x_i, x_j), \tag{2.17}$$

$$\Delta f_{ij} = f(x_i + \Delta x_i, x_j + \Delta x_j) - f(x_i, x_j), \tag{2.18}$$

where Δx_i and Δx_j are randomly generated numbers added to x_i and x_j respectively, such that the perturbed points are still in the decision space. In contrast to the optimization problems with binary decision variables, the decision space of a continuous optimization problem is usually much larger. In a heterogeneous fitness landscape where non-linearity cannot be detected in every region, it may require several attempts (more samples) to successfully identify variable interactions.

The perturbation based methods e.g., LINC and LIMD are computationally expensive. The number of Function Evaluations (FEs) used to identify all pairwise interactions between decision variables is typically $O(n^2)$, where n is the number of decision variables. Subsequently, a quasi-linear method – Dependency Detection for Distribution Derived from Df (D^5) [178] – is proposed which combines perturbation based methods with EDAs to efficiently detect variable interactions. It firstly classifies the population of candidate solutions into several sub-populations based on the fitness differences induced by perturbing decision variables. To identify the linkage group (g_i) for decision variable x_i, the decision variable that has the smallest entropy with g_i is repeatedly placed into g_i until a pre-defined group size k is reached. This process is conducted for each sub-population with reasonably large number of candidate solutions, and the linkage group with the smallest entropy is selected for decision variable x_i.

2.2.2 Model Building Algorithms

Estimation of Distribution Algorithms

EDAs [7, 107, 105, 104], also known as probabilistic model-building GAs [21], identify the interaction between decision variables based on probabilistic models. They typically approximate the distribution of decision variables based on promising candidate solutions, and use this model to generate new candidate solutions. In this section, the representative EDAs are briefly described.

The early EDAs build a simple probability model for each decision variable separately. For example the Population-Based Incremental Learning (PBIL) [7] algorithm updates a probability vector which records the percentage of ones of each gene (binary decision variable) over chromosomes. The probability vector is then used to generate new candidate solutions for evaluation. The PBIL algorithm has been adapted to solve continuous optimization problems [153, 34].

Similarly, the Uni-variate Marginal Distribution Algorithm (UMDA) [107] calculates the frequencies of each gene based on promising candidate solutions, and samples new candidate solutions from that. The compact GA [50] uses 'pairwise competition' to update the probability vector. It generates two candidate solutions each time, and the probability vector is updated based on the chromosome of the winner (if the values of decision variables of the winner and loser are different).

These three algorithms perform well when used to solve optimization problems without significant level of variable interactions [104, 50, 128]. However they do not take the underlying structure of variable interactions into considerations and can not successfully model the distribution for non-separable optimization problems.

The Mutual Information Maximization for Input Clustering (MIMIC) [27] algorithm estimates pairwise conditional distributions between decision variables from promising candidate solutions. Then the pairwise conditional distributions are used to approximate the joint probability density function of all decision variables. The Kullback-Liebler divergence is used to measure the agreement between approximated and true distributions. The one that is the closest to the true distribution is selected and used to generate more candidate solutions.

The Optimal Dependency Trees (ODT) [8] method uses a version of Chow and Liu's algorithm [23] to determine the best model for approximating the joint distribution of all decision variables. It firstly approximates the pairwise distributions of decision variables from promising candidate solutions. Then a complete weighted graph is constructed in which the vertices are decision variables and the edge weights are the MI between the corresponding decision variables. The maximum spanning tree of the graph is then computed and the edges in the maximum spanning tree is used to estimate the joint distribution of all decision variables.

The Bivariate Marginal Distribution Algorithm (BMDA) [128] is an extension of the UMDA algorithm. It takes the univariate frequencies as well as the bivariate frequencies of decision variables into consideration when constructing a dependency graph. Then new candidate solutions are generated from the dependency graph. However the BMDA, as well as the MIMIC and ODT methods, only considers pairwise interactions between decision variables, without taking into account any high-order variable interactions.

The Factorized Distribution Algorithm (FDA) [106] was originally proposed to capture the interaction structure of decision variables in additively decomposable problems. It can identify high-order interactions between decision variables. The FDA builds a Boltzmann distribution to generate new candidate solutions. The Boltzmann distribution can be efficiently calculated by the factorization of distributions. However the main limitation of the FDA algorithm is that it requires problem structure or distribution factorization as prior knowledge, which might be non-trivial to compute.

The Bayesian Optimization Algorithm (BOA) [127] constructs a Bayesian network to fit promising candidate solutions. The joint distribution of decision variables encoded in the constructed Bayesian network is then used to generate new candidate solutions. The BOA algorithm also takes high-order variable interactions into account. Moreover it can utilize both the prior information of the interaction structure and the information represented by promising candidate solutions. However in contrast to FDA, the prior knowledge of interaction structure is not essential for BOA.

Co-variance Matrix Adaptation – Evolutionary Strategy

The CMA-ES [46, 47, 45] identifies interactions between decision variables by adapting a covariance matrix of a multi-variate normal search distribution. It typically evolves the mean and covariance matrix of 'promising' candidate solutions in the evolutionary process. The population for the next generation is subsequently generated from the mean and covariance matrix.

The model described by Hansen and Ostermeier in [46] is the earliest description of the CMA-ES algorithm. It introduces a formulation which can be used to adapt arbitrary normal distributions of decision variables with zero mean. By doing so, the Evolutionary Strategy (ES) can successfully adjust to the actual scaling of a given optimization problem. Significantly, ES is made rotationally invariant – not sensible to any rotation of an objective function, by adapting the co-variance matrix.

In [47], the de-randomization and cumulation methods are presented, which can be used for self-adaptation of the distribution of decision variables. By giving high priority to the previously selected search steps, a completely de-randomized self-adaptation scheme is resulted, which adapts arbitrary normal distributions of decision variables. This scheme satisfies the basic demands of self-adaptation of normal distributions, and can be cumulatively improved by using an evolution path instead of a single search step. The experimental results in [47] reveal that ES with CMA outperforms the one without CMA when used to solve non-separable benchmark problems.

The original CMA-ES is designed to adapt a covariance matrix in populations with small number of candidate solutions. It is inefficient to exploit information from large populations. Subsequently, a fast version of CMA-ES is proposed, which is able to scale up the size of population to $10n$, where n is the problem dimensionality [45]. By efficiently incorporating the available information from a large population, it can reduce the number of generations required for convergence to an optimum. The comprehensive experimental results in [45] show that the time complexity of CMA-ES can be reduced from quadratic to linear in many cases.

Auger *et. al.* describe a second order algorithm for CMA-ES, which is called LS-CMA-ES [6]. The LS-CMA-ES algorithm approximates a quadratic model of an objective function by the least-square minimization. Then the covariance matrix is adapted based on the approximated quadratic model. The accuracy of approximation is detected through the evolutionary process. If the approximation is not accurate enough, the original CMA is used to update the covariance matrix. The experimental results in [6] show that the LS-CMA-ES algorithm outperforms the CMA-ES when tested on a large number of benchmark problems.

A restart CMA-ES, named IPOP-CMA-ES [5], increases the population size for each restart. By gradually increasing the population size, the algorithm can explore a fitness landscape more globally. The experimental results in [5] show that the IPOP-CMA-ES algorithm performed better on multi-modal benchmark problems when compared against a local restart strategy with constant population size.

2.2.3 Cooperative Co-evolution

The CC [133] framework solves an optimization problem using a divide-and-conquer strategy. It divides a problem into a number of components that are solved cooperatively. A standard CC algorithm consists of two stages: *decomposition* and *optimization*.

In the decomposition stage, an optimization problem is decomposed into several components. For example, a decomposition of a 6-dimensional optimization problem ($f : \mathbb{R}^6 \rightarrow \bar{\mathbb{R}}$) could possibly be $\{(x_1, x_2), (x_3, x_4), (x_5, x_6)\}$, as shown in Figure 2.7. When the structure of underlying decision variable interactions is considered, this allocation to components may in fact be different. Recent studies have shown that the performance of a CC algorithm relies heavily on the way an optimization problem is decomposed [120, 85, 19, 65]. An ideal decomposition is possible when the interacting decision variables are assigned into the same component, and the interdependence between each component is kept to minimal [120].

In the optimization stage, an (or several) Evolutionary Algorithm(s) can be used to optimize each component based on a context vector. The context vector is a complete candidate solution, typically consisting of the best sub-solutions from each component. When optimizing the i_{th} component, the context vector (excluding the i_{th} sub-solution) is used to combine with the individuals in the i_{th} component, to form complete candidate solutions that can be evaluated, as shown in Figure 2.8. It has been recently found that using only one context vector may be too greedy [173]. Therefore, the adaptive multi-context CC [173] framework is proposed, which employs more than one context vector to co-evolve components. The original CC framework [133] optimizes each component in a round-robin fashion – the number of FEs used to optimize each component is the same. Recently, a contribution based CC framework [196] has been proposed to efficiently allocate computational resources. In each cycle, it selects and optimizes the component that makes the greatest historical contribution to the fitness improvement.

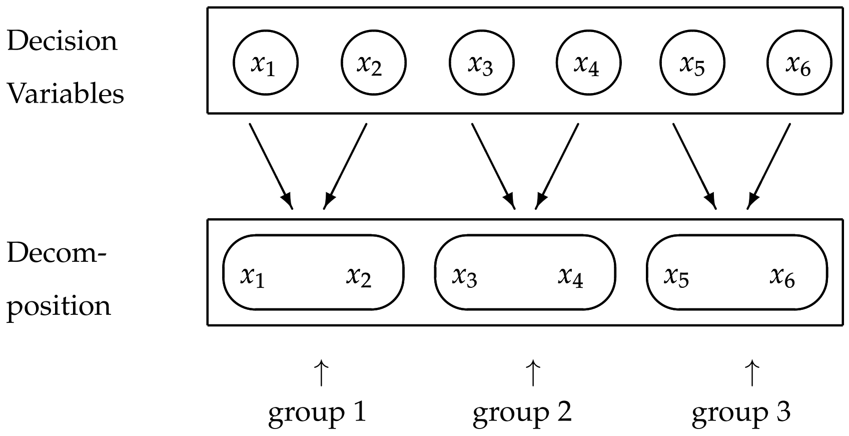

Decision
Variables
x_1 x_2 x_3 x_4 x_5 x_6
Decom-
position
x_1 x_2 x_3 x_4 x_5 x_6
group 1
group 2
group 3

Figure 2.7: An optimization problem consisting of 6 decision variables. The problem has been decomposed into 3 components, each with 2 decision variables.

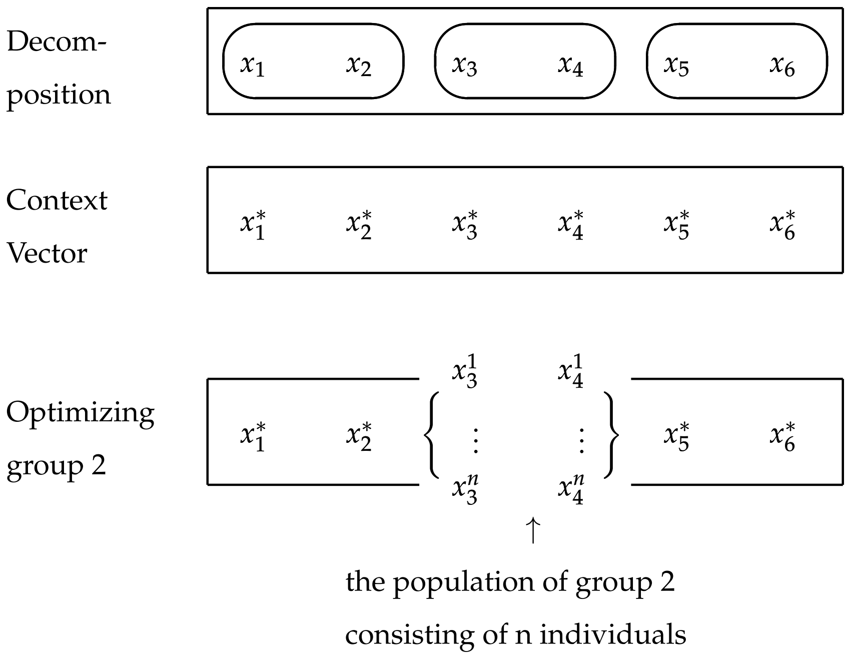

Decom-
position
x_1 x_2 x_3 x_4 x_5 x_6
Context
Vector
x_1^* x_2^* x_3^* x_4^* x_5^* x_6^*
Optimizing
group 2
x_1^* x_2^* x_3^1 x_4^1 x_3^n x_4^n x_5^* x_6^*
the population of group 2
consisting of n individuals

Figure 2.8: The optimization of a 6-dimensional problem using a CC algorithm. The problem has been decomposed into 3 components, each with 2 decision variables. When optimizing the 2_{nd} component, the context vector (excluding the 2_{nd} sub-solution) is used to combine with the individuals in the 2_{nd} component, to form complete candidate solutions that can be evaluated.

The existing decomposition methods can be classified into two very different approaches: In the *Manual Decomposition* method, the structure of components is manually designed. This method typically does not take the underlying structure of variable interactions into consideration. In the second method, *Automatic Decomposition*, the structure of components is determined by the identified decision variable interactions. However, this approach can be computationally expensive – decomposing an n-dimensional problem typically consumes $\mathcal{O}(n^2)$ FEs. This high computational complexity results in an inappropriate allocation of computational resources to the decomposition stage rather than the optimization stage. In the following, a detailed review of the two categories of decomposition methods is presented.

Manual Decomposition

In these methods, the number of components and the size of each component are manually designed. These methods work well when combined with algorithms to solve fully separable problems. However, the performance deteriorates quickly when applied on partially separable problems or fully non-separable problems. The main reason is that it does not take the underlying structure of variable interactions into consideration.

The first and simplest decomposition is the Uni-variable Grouping [133] method, which decomposes an n-dimensional problem into n 1-dimensional components. The Uni-variable Grouping method improved the performance of a GA when solving benchmark separable problems, however it degraded the GA's performance when solving a benchmark non-separable problem – the Rosenbrock's function [133]. This performance difference may be attributed to the fact that the Uni-variable Grouping method decomposes an optimization problem without considering the interaction between decision variables.

The S_k Grouping [179] method is more flexible than the Uni-variable Grouping method when used to decompose an optimization problem. It decomposes an n-dimensional problem into k s-dimensional components, where k is the number of components and s is the size of each component. That is the n decision variables of a problem are evenly and blindly divided into k groups. The S_k Grouping method has been shown to be able to improve the performance of a Particle Swarm Optimization [179] algorithm and a Biogeograph Based Optimization [201] algorithm. However, like the Uni-variable Grouping method, the S_k Grouping method does not take variable interactions into consideration.

The Random Grouping (RG) method has been proposed within the context of a Differential Evolution Cooperative Co-evolution framework [197]. It randomly assigns decision variables to predetermined number of components before each evolutionary cycle. The RG method has been successfully applied to improve the performance of a Particle Swarm Optimization [74] algorithm and an Artificial Bee Colony [140] algorithm. However, it has been shown that the probability of assigning more than two interacting decision variables into one component using the RG method is low [122]. Another limitation of RG is the requirement of setting an appropriate component size. To address this issue, the multilevel CC [198] algorithm takes the component size as a parameter, and selects an appropriate component size according to the historical performance.

The Correlation-Based Adaptive Variable Partitioning (CBAVP) [137] method identifies variable interactions based on the Pearson correlation coefficient. In the first M generations, all of the decision variables are evolved as a whole. In each of the remaining generations, the top 50% good candidate solutions of the population are used to calculate the Pearson correlation coefficient between decision variable pairs. The decision variables with a correlation coefficient greater than a predefined threshold are classified as interacting and are placed into the same component. However, this method may not fully identify variable interactions as (1) the Pearson correlation coefficient can only capture linear relationship between decision variables, and (2) the number of components is pre-determined, which may violate the variable interaction structure.

An improved version of CBAVP (CBAVP2) [160] takes better advantage of variable interaction structures when decomposing a problem. Different from its predecessor, CBAVP2 maintains an archive of all candidate solutions explored, and the 50% good solutions from the archive are used to calculate the Pearson correlation coefficient between decision variable pairs. For each decision variable x_i, the interacting decision variables are placed into the same group i as x_i, where $1 \leq i \leq n$. Then overlaps between the n groups are searched, and the groups sharing the same decision variable(s) are merged. Finally, the decomposition is adjusted by a pre-determined parameter – the maximal number of components. The CBAVP2 method has the potential to identify indirect variable interactions (see Definition 3.1). However, it has the same limitations as its predecessor.

A more sophisticated method – Delta Grouping [123] identifies variable interactions based on the averaged difference in a certain decision variable across the whole population. The rationale behind Delta Grouping is that the improvement interval of non-separable variables is relatively smaller than those of separable variables [151]. The Delta Grouping method sorts decision variables based on the averaged dimension-wise displacements of candidate solutions over the entire population. It generally outperforms the RG method when incorporated with the Differential Evolution Cooperative Co-evolution framework to solve the CEC'2010 benchmark problems [123]. However on benchmark problems with more than one non-separable component, the performance of the Delta Grouping method is low [120].

The Min-variance and Max-variance [81] methods decompose a problem into k s-dimensional components based on the covariance matrix of CMA-ES. Decision variables are sorted based on their variances in the main diagonal of the co-variance matrix. The min-variance method assigns decision variables with similar variances (neighbours in the sorted list of decision variables) into one component, while the max-variance method allocates decision variables with an interval of s in the sorted list (e.g., x_1, x_{1+s}, $\cdots$) into the same component. It is suggested that the min-variance method is appropriate for exploration, while the max-variance method is good for exploitation [81].

The k-means grouping [83] method uses a k-means clustering algorithm to construct decision variable groups. The decision variables with similar effects on the fitness value are placed into the same component. The component with the greatest contribution to the fitness value will be optimized with more iterations (FEs). The idea is similar to the contribution based CC [124] framework. Unlike most decomposition methods which group decision variables based on variable interactions, the k-means grouping method groups decision variables based on their contribution to the fitness improvement. Therefore, it is specifically tailored for problems with unbalanced components [121, 83].

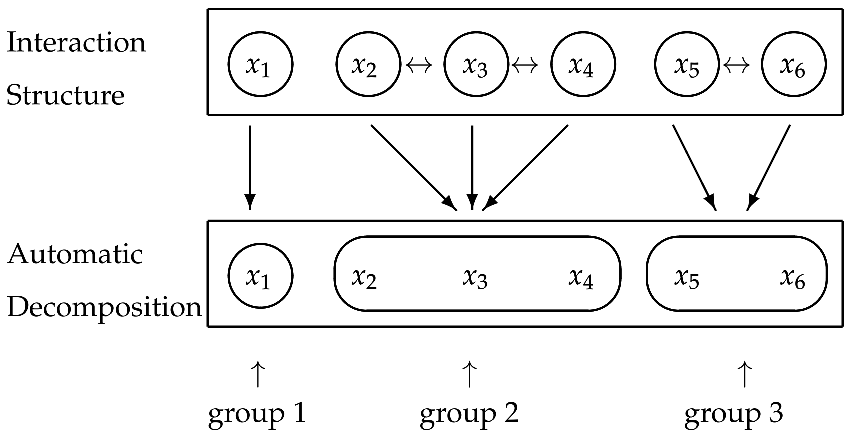


Figure 2.9: The variable interaction structure and automatic decomposition of the objective function shown in Eq. (2.19). The notation $x_i \leftrightarrow x_j$ denotes that decision variable x_i interacts with x_j.

Automatic Decomposition

In these methods, the interacting decision variables are identified and automatically placed into the same component. It is important to note that automatic decomposition caters for the underlying variable interaction structure encapsulated within a search landscape. Taking the following objective function as an example:

$$f(\mathbf{x}) := x_1^2 + (x_2 - x_3)^2 + (x_3 - x_4)^2 + (x_5 - x_6)^2,\ \mathbf{x} \in [-1, 1]^6, \tag{2.19}$$

the decision variables (x_2, x_3, x_4) interact, and (x_5, x_6) interact. Therefore, the decomposition by automatic decomposition methods is $\{(x_1), (x_2, x_3, x_4), (x_5, x_6)\}$, as shown in Figure 2.9. The decomposition methods proposed in Chapter 4 and Chapter 5 fall into this category.

A representative automatic decomposition method – Cooperative Co-evolution with Variable Interaction Learning (CCVIL) [20] – identifies pairwise interactions between decision variables by non-monotonicity detection. If the monotonicity of fitness function with respect to x_i does not change for different values of x_j, x_i and x_j are independent. Otherwise, decision variables x_i and x_j interact, and they are placed into the same component. The rationale behind the CCVIL method is consistent with the LIMD [113] method. The CCVIL method is more accurate than most of the manual decomposition methods when identifying variable interactions. However, it still can not obtain acceptable results when used to decompose some benchmark problems [120].

The Statistical Variable Interdependence Learning (SVIL) [165] method identifies variable interactions based on non-monotonicity detection as well. Unlike the CCVIL method, the SVIL method detects the monotonicity relationship between x_i and x_j multiple times. The probability (p_{ij}) of the observation of non-monotonicity is calculated. If the probability p_{ij} is greater than a user defined threshold, decision variables x_i and x_j interact. The main issue of this method is its high computational complexity. The number of FEs needed to decompose an n-dimensional optimization problem is $4mn^2$, where m is the number of non-monotonicity detections conducted for each pair of decision variables.

To address the high computational complexity of the SVIL method, a Fast Variable Interdependence Searching (FVIS) [35] method is proposed. It greatly improves the decomposition efficiency by examining the interaction between one decision variable (x_i) and a set of other decision variables m times based on non-monotonicity detection. If any interaction is detected, the set of decision variables will be divided into two subsets and the interaction between x_i with each decision variable subset will be checked. The computational cost of decomposing an n-dimensional problem can be reduced to $4mn\log(n)$ in the worst case.

Another automatic decomposition method, named DM-HDMR [84], identifies variable interactions based on a high dimensional model representation technique. The DM-HDMR method constructs a second-order meta-model for approximating a black-box fitness function based on RBF-HDMR [155]. If two decision variables have a cooperative effect on fitness values, they are identified as interacting and placed into the same component. The DM-HDMR method generally outperforms the CCVIL method when tested across a suite of benchmark problems [84]. However, it can not accurately identify all pairwise interactions between decision variables in some benchmark problems. Moreover, the computational cost of the DM-HDMR method when used to decompose some benchmark problems is high.

The DG [120] method identifies variable interactions by detecting the fitness changes when perturbing decision variables. If the fitness change induced by perturbing decision variable x_i varies for different values of x_j, x_i and x_j interact. The rationale behind the DG method is consistent with the LINC-R [176] method. The DG method outperforms the CCVIL method when used to decompose the CEC'2010 benchmark problems [120]. However it has been shown not to be able to completely identify interacting decision variables in overlapping problems [93]. In Chapter 4, I will propose an extended Differential Grouping method to address this issue, by placing both directly and indirectly interacting (see Definition 3.1) decision variables into one component. Another limitation of DG is that it requires users to select a pre-defined threshold value to identify variable interactions, which may require significant domain knowledge. Moreover, the DG method is computationally expensive – decomposing an n-dimensional fully separable problem consumes around n^2 FEs. The high complexity of DG results in an inappropriate allocation of computational resources between decomposition and optimization.

A variant of DG – Global Differential Grouping (GDG) [93] – can decompose an n-dimensional optimization problem using $(n^2 + 3n + 2)/2$ FEs. It firstly identifies pairwise variable interactions based on non-linearity detection, and forms a complete variable interaction matrix. Then an undirected graph is constructed by treating each decision variable as a node. If two decision variables directly interact, an undirected edge is placed between the two corresponding nodes. Then the well-known depth-first search or breadth-first search algorithms can be used to identify strongly connected components, which will be regarded as the components. Both the directly and indirectly interacting decision variables will be placed into one connected component (component). Similar procedure is adopted by the graph based DG [77] method when used to decompose a problem. The GDG method estimates a threshold value ϵ to identify variable interactions based on the magnitude of fitness values.

Another improved version of DG, named DG2 [126], can further reduce the computational cost to $(n^2 + n + 2)/2$ FEs by re-using the samples, which has been shown to be the lower bound of identifying the complete variable interaction matrix based on non-linearity detection. With the complete variable interaction matrix being identified, it is possible to generate an effective decomposition for problems with overlapping components [126]. However, the existing automatic decomposition methods, including DG2, do not utilize this information and simply assign all the linked decision variables into the same component. I argue that it may not need the entire variable interaction matrix to identify the connected components (components). For example, if decision variable x_1 interacts with x_2 and x_3, the interaction between x_2 and x_3 needs not to be checked, as they belong to the same connected component. DG2 adaptively estimates upper and lower bounds on the computational round-off errors, which are used as threshold values to identify variable interactions.

The Fast Interdependency Identification (FII) [57] method can further improve the efficiency of problem decomposition by avoiding the need to identify the complete variable interaction matrix. FII firstly identifies separable decision variables by examining the interaction between one decision variable and other variables. Then the interaction between non-separable decision variables is examined, and all the linked (connected) decision variables are placed into the same component. The FII method is efficient when used to decompose problems with a large portion of separable decision variables. However on problems with indirect variable interactions, the number of FEs used by FII may still be in the magnitude of n^2. It has been shown that FII uses $3n + kn_n + k$ FEs when decomposing an n-dimensional problem with equally sized non-separable components, where n_n is the number of non-separable decision variables, and k is the number of non-separable components [57]. On the fully non-separable Rosenbrock's function [73], n_n is equal to n and each decision variable interact with at most two other decision variables. Therefore the total number of FEs used by FII is more than $n^2/3 \in \Theta(n^2)$. In Chapter 5, I will propose an efficient and robust method that can decompose any n-dimensional problem using less than $6n \log_2(n)$ FEs.

2.3 A Taxonomy of the Methods Identifying Variable Interactions

To conclude this chapter, I present a taxonomy of the methods that can be used to identify or quantify variable interactions in Fig 2.10. The advantages and disadvantages of the methods in each category are discussed.

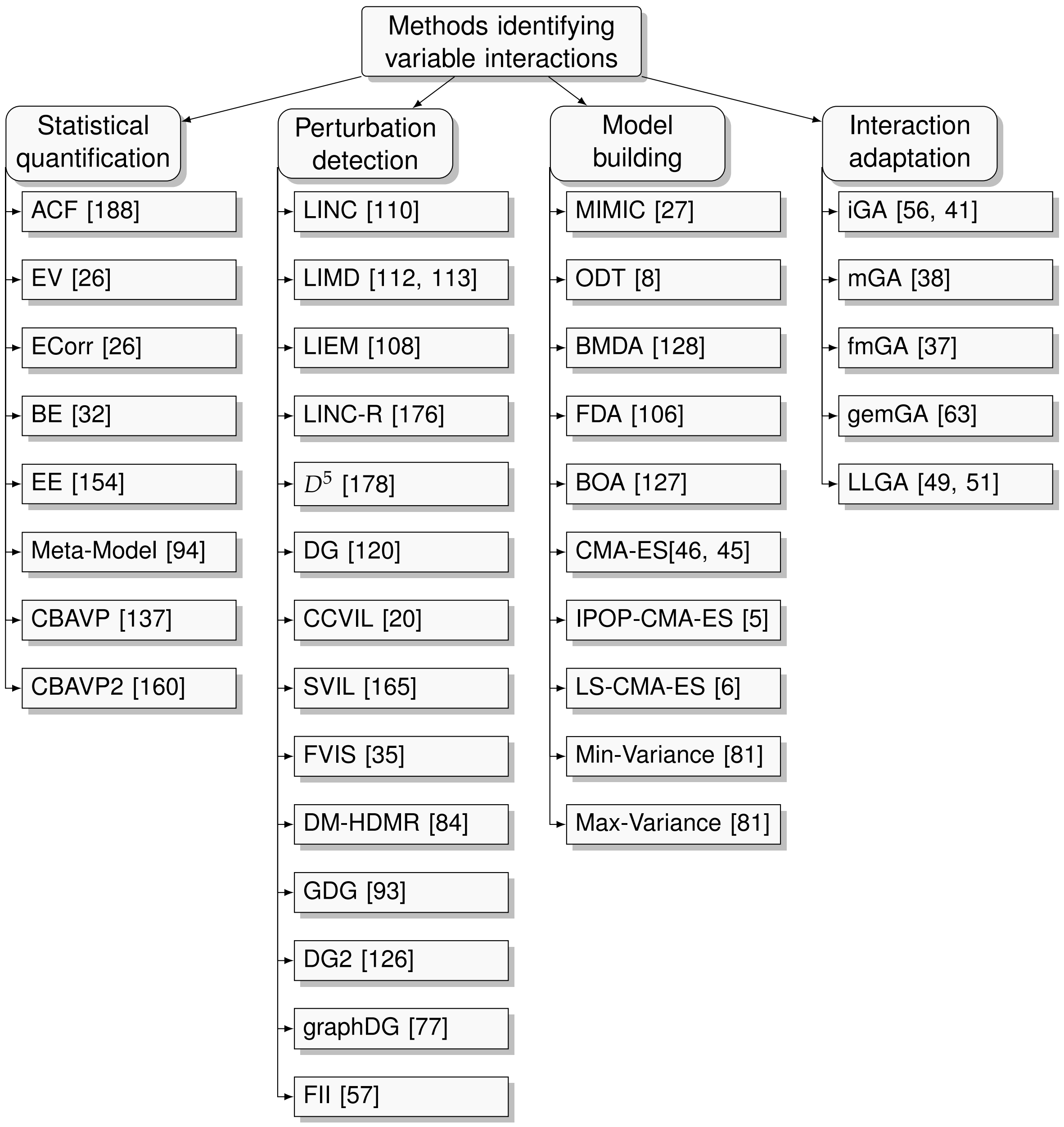


Figure 2.10: A taxonomy of the methods that can be used to identify or quantify variable interactions in the EC domain. The methods are classified into four categories. Note that the review here is not complete/exhaustive. I only summarize the representative methods in each category.

The *statistical quantification* methods identify (or quantify) variable interactions based on 'statistical' measures. The CBAVP and CBAVP2 methods, as well as the ELA measures described in Section 2.1.2 fall into this category. These methods can quantify the level of variable interactions across a fitness landscape. Most of the statistical quantification methods are computationally less intensive, and are not sensitive to noise. However, they are typically not able to accurately identify pairwise variable interactions.

The *perturbation detection* methods identify variable interactions by adding a small perturbation to decision variables and detecting the changes of fitness values. This category includes the fixed coding methods for GA building blocks described in Section 2.2.1 and the automatic decomposition methods for CC described in Section 2.2.3. The perturbation detection methods can explicitly identify pairwise variable interactions in a fitness landscape with isotropic interaction structure. However, they are computationally expensive – identifying all pairwise variable interactions in an n-dimensional optimization problem typically requires $O(n^2)$ FEs. The perturbation detection methods may be sensitive to computational errors and noise in the system. Smooth landscapes pose additional challenges for these methods; the addition of a perturbation to decision variables may not result in significant changes of fitness values and thus may be difficult to detect.

The *model building* methods adapt a probabilistic distribution of decision variables as part of the evolutionary process. The EDAs and CMA-ES fall into this category. The model building methods encapsulate variable interactions in a probabilistic model, therefore they do not require additional FEs. However, the model building methods are sensitive to the magnitude of fitness values. They can not successfully identify variable interactions in components which make less contribution to the fitness, due to low selection pressure. Besides, updating a probabilistic model can be computationally expensive especially for large-scale optimization problems.

The *interaction adaptation* methods evolve variable interactions in the evolutionary process. The adaptive coding methods for GA building blocks described in Section 2.2.1 belong to this category. In these methods, variable interactions emerge as the population evolves, based on the hypothesis that tight building blocks have high probability to survive. The interaction adaptation methods do not require extra FEs to identify variable interactions. However, the emergence rate of the variable interaction structure is usually much slower than the convergence rate of the population. Another limitation is that they typically require a large initial population, resulting in a high computational cost.

Chapter 3

Quantifying Variable Interactions in Continuous Optimization Problems

EXPLORATORY Landscape Analysis (ELA) [95, 94, 96] is a technique that can be used to explore the fitness landscape [162] and capture certain characteristics of a problem. Once characteristics are identified, they can be used to predict problem difficulty [60, 86] or to select an appropriate algorithm to use when solving the problem [114, 115, 116]. A number of ELA measures has been proposed to explore variable interactions in combinatorial optimization problems (see Section 2.1.2 for a review). However there is a lack of research quantifying the level of interactions between continuous variables and the effects such interactions have on problem difficulty.

In this chapter, I investigate the two research questions of this thesis (presented in Section 1.1) in the Black-box Continuous Optimization Problem (BCOP) domain: $\mathfrak{RQ}3.1$ *How can variable interactions be accurately identified and quantified in a given BCOP?* and $\mathfrak{RQ}3.2$ *How can variable interactions be used to effectively solve a given BCOP?*

To answer the first research question ($\mathfrak{RQ}3.1$), I propose a novel ELA measure – *Maximum Entropic Epistasis* (MEE) – to explore variable interactions in BCOPs. The MEE measure identifies the *Interaction Matrix* (IM) of decision variables in a BCOP. Each entry in the IM describes the corresponding pairwise interaction between decision variables, as shown in Figure 3.1. MEE identifies interactions between decision variables x_i and x_j by calculating the Maximal Information Coefficient (MIC) [141] between x_j and the partial derivative of the objective function with respect to x_i (see Section 3.1 and/or [2, 141] for the implementation of MIC). The level of variable interactions in the BCOP is subsequently approximated using three indices derived from the IM.

$$\begin{array}{c|ccc} & x_1 & x_2 & x_3 \\ \hline x_1 & \text{-} & 1 & 0 \\ x_2 & 1 & \text{-} & 0 \\ x_3 & 0 & 0 & \text{-} \end{array}$$

Figure 3.1: The interaction matrix: '1' means 'interacting' and '0' means 'independent'. A '-' on the diagonal entry indicates that it is meaningless to measure the level of interaction between one decision variable and itself.

I evaluate the efficacy of MEE using a suite of 24 benchmark continuous optimization functions with different levels of variable interactions. The experimental results show that MEE can identify the IM and accurately quantify the level of variable interactions. I then show that there is a correlation between the MEE level and the solution quality found by an Evolutionary Algorithm (EA). The rationale behind this particular analysis exercise is based on the hypothesis that the level of variable interaction is a factor that makes a BCOP challenging for some of the EAs to solve [62, 191, 121].

To answer the second research question ($\mathfrak{RQ}$3.2), the MEE measure is used to effectively guide the search for an optimal solution to a given BCOP. The algorithm design framework, proposed in [18], employs the Pearson Correlation Coefficient (PCC) to quantify the level of variable interactions in a BCOP. Based on that, appropriate operator(s) is (are) selected to guide the search. I observe that by using MEE, instead of the PCC to quantify the level of variable interactions, equal or better results can be achieved when solving the benchmark BCOPs.

This chapter is organized as follows. Section 3.1 introduces MIC in detail. Section 3.2 defines two types of variable interactions, and describes the proposed MEE measure. Section 3.3 describes experiments to evaluate the proposed MEE measure. Section 3.4 presents and analyzes the experimental results. Section 3.5 shows how MEE can be embedded within a framework to design efficient optimization algorithms. Section 3.6 concludes this chapter.

3.1 Related Work: Maximal Information Coefficient

In this section, I describe the MIC in detail. Typical measures of correlation between two variables, such as the PCC, capture only linear relationships. In contrast, it is possible to measure statistical dependence between two variables, without assuming the relationship is linear, using the information-theoretic Mutual Information (MI) [157]. The MI between two continuous random variables X and Y is defined as:

$$I(X;Y) = \int_{x \in X} \int_{y \in Y} p(x,y) \log \left(\frac{p(x,y)}{p(x)p(y)} \right), \tag{3.1}$$

where $p(x)$ and $p(y)$ are the marginal PDFs of variables X and Y, and $p(x,y)$ is the joint PDF of X and Y. MI has been used, for example, to detect phase transitions in a variety of systems, including socio-economic systems [13].

Estimating MI accurately from limited data is a difficult problem, especially for continuous variables [161] (see [58] for a review of methods). Consequently, the MIC, was recently introduced to address this problem [141]. MIC is part of the Maximal Information-based Nonparametric Exploration suite, which uses 'bins' as a means to apply MI on continuous random variables, thus capturing a broad range of associations both functional and not. In this chapter, a functional relationship means a distribution (X,Y) in which Y is a function (or a superposition of several functions) of X, potentially with independent noise added.[1] In the continuous optimization domain, most of the optimization problems investigated are functions. Therefore, only functional relationships are considered in this chapter.

MIC computes the MI between two variables at a variety of scales and finds the largest possible MI at any scale. Let D denote a set of ordered pairs, $\{(x_i, y_i), i = 1, \dots, n\}$, G denote a m-by-n grid covering D. That means dimensions x and y are partitioned into m and n intervals respectively. The PDF of a grid cell is proportional to the number of data points inside that cell. The characteristic matrix $M(D)_{m,n}$ represents the highest

[1]The mapping $f : \mathbb{R}^n \to c$ (c is a constant) is not considered as a functional relationship in this chapter.

normalized MI of D with the m-by-n partition, which is defined as

$$M(D)_{m,n} = \frac{\max(MI)}{\log \min(m,n)}, \tag{3.2}$$

where $\max(MI)$ is the maximum MI of D by all possible m-by-n partitions. The MIC of a set D is then defined as

$$\text{MIC}(D) = \max_{0<mn<B(N)} \{M(D)_{m,n}\}, \tag{3.3}$$

where N is the sample size, and the function $B(N) = N^{0.6}$ was heuristically determined by Reshef et al [141]. It represents the maximal value in the characteristic matrix $M(D)$ subject to $0 < mn < B(N)$.

It should be noted that the MIC tends to 1 for functional relationships with probability approaching 1 as the sample size grows, and converges to 0 for statistically independent variables [141].

3.2 Maximum Entropic Epistasis

In this section, I describe two distinct types of variable interactions which may appear in a BCOP. A novel ELA measure – MEE – is then proposed to quantify the level of variable interactions in a BCOP.

In a BCOP, the level of interaction between given decision variables may be different. Take the following objective function as an example:

$$f(\mathbf{x}) := (x_1 - x_2)^2 + (x_2 - x_3)^2 + x_4^2, \quad \mathbf{x} \in [-1,1]^4. \tag{3.4}$$

Both $\{x_1, x_2\}$ and $\{x_1, x_3\}$ interact with each other. However, x_1 and x_2 interact directly; x_1 and x_3 are linked by x_2. The former is called *direct interaction* and the latter is called *indirect interaction*. The formal definitions of direct interaction and indirect interaction are described as follows:

Definition 3.1. *Let $f : \mathbb{R}^n \to \bar{\mathbb{R}}$ be a differentiable function. Decision variables x_i and x_j*

interact directly if a candidate solution $\mathbf{x}^*$ *exists, such that*

$$\frac{\partial^2 f(\mathbf{x}^*)}{\partial x_i \partial x_j} \neq 0. \tag{3.5}$$

denoted by $x_i \leftrightarrow x_j$. *Decision variables* x_i *and* x_j *interact indirectly if for any candidate solution* $\mathbf{x}$,

$$\frac{\partial^2 f(\mathbf{x})}{\partial x_i \partial x_j} = 0, \tag{3.6}$$

and a set of decision variables $\{x_{k1}, \ldots, x_{kt}\} \subset X$ *exists, such that* $x_i \leftrightarrow x_{k1} \leftrightarrow \ldots \leftrightarrow x_{kt} \leftrightarrow x_j$. *Decision variables* x_i *and* x_j *are independent if for any candidate solution* $\mathbf{x}$, *Eq. (3.6) holds and a set of decision variables* $\{x_{k1}, \ldots, x_{kt}\} \subset X$ *does not exist, such that* $x_i \leftrightarrow x_{k1} \leftrightarrow \ldots \leftrightarrow x_{kt} \leftrightarrow x_j$.

Definition 3.1 describes the direct interaction and indirect interaction based on the partial derivative of a differentiable function. However if a function is not differentiable, the partial derivative may not exist. In this case, the partial sub-derivative [146, 102] can be used as a substitute for the partial derivative, which will be detailed in the following.

Lemma 3.1. *Let* $f\colon \mathbb{R}^n \to \mathbb{R}$ *be a differentiable function. If* $\frac{\partial f(\mathbf{x})}{\partial x_i}$ *and* x_j *have a functional relationship,* x_i *and* x_j *interact directly. Otherwise,* x_i *and* x_j *interact indirectly or are independent.*

Proof. Without loss of generality, I assume that

$$\frac{\partial f(\mathbf{x})}{\partial x_i} = g(\mathbf{x}_1), \tag{3.7}$$

where $\mathbf{x}_1$ is the decision vector of the subset of decision variables $\mathbf{X}_1 \subset \mathbf{X}$, and $g(\mathbf{x}_1)$ is a function of $\mathbf{x}_1$.[2] If $\frac{\partial f(\mathbf{x})}{\partial x_i}$ is a function of x_j, then $x_j \in \mathbf{X}_1$. By taking the derivative with respect to x_j,

$$\frac{\partial^2 f(\mathbf{x})}{\partial x_i \partial x_j} = \frac{\partial g(\mathbf{x}_1)}{\partial x_j} \neq 0. \tag{3.8}$$

According to Definition 3.1, x_i and x_j interact directly. On the other hand, if $\frac{\partial f(\mathbf{x})}{\partial x_i}$ is not a

[2]If $\mathbf{X}_1$ is $\emptyset$, $g(\mathbf{x}_1)$ is a constant.

function of x_j, then $x_j \notin \mathbf{X}_1$. By taking the derivative with respect to x_j,

$$\frac{\partial^2 f(\mathbf{x})}{\partial x_i \partial x_j} = \frac{\partial g(\mathbf{x}_1)}{\partial x_j} = 0. \tag{3.9}$$

According to Definition 3.1, x_i and x_j interact indirectly or are independent. □

Lemma 3.2. *Two variables X and Y have a functional relationship, if* $\lim_{m\to\infty} MIC(X,Y) = 1$*; two variables X and Y are independent, if* $\lim_{m\to\infty} MIC(X,Y) = 0$*, where m is the sample size,* $MIC(X,Y)$ *is the MIC of X and Y [141].*

Proposition 3.1. *Let* $f\colon \mathbb{R}^n \to \mathbb{R}$ *be a differentiable function. If* $\lim_{m\to\infty} MIC(\frac{\partial f(\mathbf{x})}{\partial x_i}, x_j) = 1$*,* x_i *and* x_j *interact directly; if* $\lim_{m\to\infty} MIC(\frac{\partial f(\mathbf{x})}{\partial x_i}, x_j) = 0$*,* x_i *and* x_j *interact indirectly or are independent.*

Proof. Refer to Lemma 3.1 and Lemma 3.2. □

Proposition 3.1 shows a straightforward way to identify the direct interaction between decision variable pairs in a differentiable function. If the MIC between $\partial f(\mathbf{x})/\partial x_i$ and x_j is greater than a given threshold, then x_i and x_j can be regarded as directly interacting. The threshold value will be determined based on empirical studies. However, in a non-differentiable function, the partial derivative ($\partial f(\mathbf{x})/\partial x_i$) may not exist. Therefore, Proposition 3.1 can not be directly applied to identify variable interactions in a non-differentiable function. In this case, the partial sub-derivative ($\hat{\partial} f(\mathbf{x})/\partial x_i$) [146] can be used as a replacement for the partial derivative ($\partial f(\mathbf{x})/\partial x_i$).

The partial sub-derivative ($\hat{\partial} f(\mathbf{x})/\partial x_i$) is defined as follows:

$$\frac{\hat{\partial} f(\mathbf{x})}{\partial x_i} = \left\{ v \,\middle|\, \liminf_{\delta x_i \to 0} \frac{f(\cdots, x_i + \delta x_i, \cdots) - f(\cdots, x_i, \cdots) - v \cdot \delta x_i}{|\delta x_i|} \geq 0 \right\}. \tag{3.10}$$

The lim inf denotes the *lower limits*, defined as:

$$\liminf_{x \to \bar{x}} \varphi(x) = \lim_{r \to 0} \Big(\inf_{x \in B(\bar{x}, r)} \varphi(x) \Big), \tag{3.11}$$

where $B(\bar{x}, r)$ is a ball of center $\bar{x}$ and radius r (excluding $\bar{x}$), and φ is a function. Each $v \in \hat{\partial} f(\mathbf{x})/\partial x_i$ is a first-order partial sub-derivative of f with respect to x_i.

In a BCOP, the partial (sub-)derivative information is not known. Therefore, I approximate this value using the expression:

$$\frac{\partial f(\mathbf{x})}{\partial x_i} \text{ or } \frac{\hat{\partial} f(\mathbf{x})}{\partial x_i} \approx \frac{f(\cdots, x_i + \delta x_i, \cdots) - f(\cdots, x_i, \cdots)}{\delta x_i}, \tag{3.12}$$

where δx_i is a small perturbation of x_i.[3] It is important to note that the estimation of partial (sub-)derivatives consumes additional Function Evaluations (FEs), which will be analyzed in the end of this section.

The direct interaction between decision variables can be identified by Proposition 3.1. The question then, is: 'How to identify indirect interaction between decision variables?'. According to Definition 3.1, if there is not a direct interaction between x_i and x_j, but a set of interacting decision variables can be found to link them together, then x_i and x_j are regarded as indirectly interacting. Inspired by this, I propose a method to comprehensively capture both direct and indirect interactions between decision variables.

The proposed algorithm consists of two stages. The first stage is to identify direct interaction between decision variables (Algorithm 1), and the second stage is to identify indirect interaction between decision variables.

In Algorithm 1, the inputs are: f is the objective function, n is the dimensionality, **ub** and **lb** are the upper and lower bounds of the search space, m is the sample size, ϵ is the threshold to identify direct interaction between pairwise decision variables, β is employed to remove the computational errors in the system and δx is used to calculate the partial derivatives.

Algorithm 1 begins by initializing the IM to 0. The IM is a $n \times n$ symmetric matrix, which contains the information of all pairwise interactions between decision variables. $IM(i, j) = 1$ means decision variables x_i and x_j interact with each other; while $IM(i, j) = 0$ means x_i and x_j are independent.

For each pair of decision variables (x_i, x_j), $1 \leq i < j \leq n$, the direct interaction is identified as follows. It begins by randomly generating a candidate solution $\mathbf{x}^*$ and an array, A, of m elements from the search space. Firstly, the algorithm substitutes the j_{th}

[3]The value of δx_i relies on the scale of decision variables. The default value of δx_i is set to 10^{-6} in the empirical studies.

Algorithm 1 MEE: Identifying Direct Interaction

```
Require: $f, n, \mathbf{ub}, \mathbf{lb}, m, \epsilon, \beta, \delta x$
1: Initialize all the elements in IM to 0
2: for $i = 1$ to $n$ do
3:   for $j = i + 1$ to $n$ do
4:     Randomly generate a candidate solution $\mathbf{x}^*$ from the search space.
5:     Randomly generate an array $(A)$ of $m$ elements from dimension $x_j$
6:     for $k = 1$ to $m$ do
7:       Substitute the $j_{th}$ element of $\mathbf{x}^*$ $\big(\mathbf{x}^*(j)\big)$ with the $k_{th}$ element of $A$ $\big(A[k]\big)$
8:       Calculate the fitness value of $\mathbf{x}^*$: $y_1 \leftarrow f(\mathbf{x}^*)$
9:       Calculate fitness when adding $\delta x$ to the $i_{th}$ element of $\mathbf{x}^*$: $y_2 \leftarrow f\big(\mathbf{x}^*(\cdots, x_i + \delta x, \cdots)\big)$
10:      Place the approximated derivative in the $k_{th}$ element of $D$: $D[k] \leftarrow \frac{y_2 - y_1}{\delta x}$
11:    end for
12:    Calculate the mean of the elements in $D$: $ave \leftarrow mean(D)$
13:    for $k = 1$ to $m$ do
14:      if $|(D[k] - ave| < \beta$ then
15:        $D[k] \leftarrow ave$
16:      end if
17:    end for
18:    if $MIC(D, A) > \epsilon$ then
19:      $IM(i, j) \leftarrow 1$ and $IM(j, i) \leftarrow 1$
20:    end if
21:  end for
22: end for
23: return $IM$ as $IM_d$ // The IM only with direct interaction.
```

element of $\mathbf{x}^*$ with the first element of the array $A[1]$. Then the algorithm calculates the partial derivative of f with respect to x_i and puts it in the first element of an empty array D. Secondly, the algorithm substitutes the j_{th} element of $\mathbf{x}^*$ with the second element of the array $A[2]$. Again the algorithm calculates $\partial f(\mathbf{x}^*)/\partial x_i$ and puts it in the second element of D. This process continues until all elements in A are used and m elements of D are obtained. $\partial f(\mathbf{x}^*)/\partial x_i$ is calculated as follows: calculate the fitness value of $\mathbf{x}^*$, denoted by y_1; add a small value δx to the i_{th} element of $\mathbf{x}^*$ and calculate the fitness value again, denoted by y_2; calculate $(y_2 - y_1)/\delta x$ as the approximation of $\partial f(\mathbf{x}^*)/\partial x_i$.

If x_i and x_j do not directly interact with each other, theoretically $\partial f(\mathbf{x})/\partial x_i$ should maintain the same value when x_j changes. However, due to numerical computational errors, $\partial f(\mathbf{x})/\partial x_i$ may fluctuate slightly around the true value. The small computational errors of the partial derivative in turn may result in large computational errors of the MIC between $\partial f(\mathbf{x})/\partial x_i$ and x_j. This may significantly affect the accuracy of identifying interacting variables. In order to remove the computational errors, the parameter β is employed, such that if the difference between $\partial f(\mathbf{x})/\partial x_i$ and the mean of the m number of $\partial f(\mathbf{x})/\partial x_i$ is less than β, the algorithm sets $\partial f(\mathbf{x})/\partial x_i$ to the mean.

Then, the algorithm calculates the MIC between the filtered D and the array A. If $MIC(D, A)$ is greater than or equal to a predetermined threshold ϵ, then x_i and x_j are regarded as directly interacting with each other, and the corresponding entries of the IM $\big(IM(i,j)$ as well as $IM(i,j)\big)$ are set to 1.

Once the IM_d (IM only with direct interactions) is identified, the second stage is deployed, attempting to identify the indirect interactions. This stage begins by constructing an interaction graph based on the IM_d: setting each decision variable x_i as a vertex i; connecting vertices i and j if $IM_d(i,j) = 1$. Then the Breadth First Search algorithm is employed to search for the strongly connected components in the graph. Finally, all the pairs of vertices i and j in each identified strongly connected components are connected and the corresponding $IM(i,j)$ and $IM(j,i)$ are set to 1. The algorithm returns the IM as the output.

With the IM and IM_d identified, three measures can be calculated: *Degree of Direct Variable Interaction* (DDVI), *Degree of Indirect Variable Interaction* (DIVI) and *Degree of Variable Interaction* (DVI):

$$DDVI = \frac{\sum\limits_{1 \le i \ne j \le n} IM_d(i,j)}{n(n-1)} \tag{3.13}$$

$$DVI = \frac{\sum\limits_{1 \le i \ne j \le n} IM(i,j)}{n(n-1)} \tag{3.14}$$

$$DIVI = DVI - DDVI \tag{3.15}$$

I conclude this section with a brief discussion of the associated computational cost (complexity) of the proposed MEE measure. For each pair of decision variables, the computational cost when examining direct interactions is $2m$ with respect to FEs, where m is the sample size. In a n-dimensional problem, the total number of decision variable pairs is $n(n-1)/2$. Therefore, the total number of FEs used to obtain the IM is $mn(n-1)$. Note that the second stage of MEE does not use any FE.

3.3 Experimental Methodology

To evaluate the efficacy of the proposed MEE measure, detailed numerical experiments are conducted to investigate the following questions:

Q3.1 Is it possible to accurately identify the IM of a BCOP using MEE? (Section 3.3.3)

Q3.2 Can the MEE measure be used to quantify the DVI in a BCOP? (Section 3.3.4)

Q3.3 Is the MEE measure correlated with the solution quality found by an EA? (Section 3.3.5)

It is important to note that the three questions Q3.1, Q3.2 and Q3.3 only refer to the first research question ($\mathfrak{RQ}$3.1).

3.3.1 Benchmark Problems

To adequately investigate the three questions, I extend the benchmark problems from the CEC 2013 special session on large scale global optimization [73]. The original benchmark suite consisted of 15 benchmark problems with fixed dimensionality (1000 or 905). I extend the suite to 24 benchmark problems with tunable dimensionality, which is more reliable and flexible to evaluate the methods identifying variable interactions. Operators such as rotating, shifting, symmetry breaking and adding ill-condition [121] are also available in this extended benchmark suite.

The extended benchmark suite consists of 6 categories with 24 functions in total:

I fully separable functions: f_1 to f_4,

II partially separable functions with $n/2$ separable components: f_5 to f_8,

III partially separable functions with $n/2$ non-separable components: f_9 to f_{13},

IV partially separable functions with 2 non-separable components: f_{14} to f_{16},

V overlapping functions: f_{17} to f_{20},

VI fully non-separable functions: f_{21} to f_{24}.

The details of the 24 benchmark problems and base functions are presented in Table A.2 and Table A.1 in Appendix A. The letter R represents the rotation operator, which is employed to generate interactions between decision variables. The condition number for all of the benchmark problems is 100 except for f_1. The optima of all the benchmark problems are shifted to $f(\mathbf{0}) = 0$.

As the interactions between decision variables are known for the benchmark problems, I can use this suite of functions to investigate the accuracy of any method used to identifying variable interactions.

3.3.2 Performance Metrics

Standard machine learning performance metrics are introduced to evaluate the performance of the proposed method when used to identify variable interactions: *precision*, *recall* and *F-score* (Q3.1). *Precision* measures the accuracy of identifying interacting decision variables, while *recall* measures the comprehensiveness of identifying interacting decision variables. The *F-score* is the harmonic mean of *precision* and *recall* .

Definition 3.2. *For a given objective function $f : \mathbb{R}^n \to \mathbb{R}$, and a method ϕ identifying variable interactions, let IM_f be the true IM of $f(\mathbf{x})$, and IM_ϕ be the IM returned by ϕ. Then the precision, recall and F-score of ϕ on f are defined as follows:*

$$precision = \frac{\sum\limits_{1 \le i \ne j \le d} IM_f(i,j) \cdot IM_\phi(i,j)}{\sum\limits_{1 \le i \ne j \le d} IM_\phi(i,j)}, \tag{3.16}$$

$$recall = \frac{\sum\limits_{1 \le i \ne j \le d} IM_f(i,j) \cdot IM_\phi(i,j)}{\sum\limits_{1 \le i \ne j \le d} IM_f(i,j)}, \tag{3.17}$$

$$F\text{-}score = \frac{\sum\limits_{1\leq i\neq j\leq d} 2\cdot IM_f(i,j)\cdot IM_\phi(i,j)}{\sum\limits_{1\leq i\neq j\leq d} IM_\phi(i,j)+IM_f(i,j)}. \tag{3.18}$$

Example 3.1. *For the objective function listed in Equation (3.4), the true IM (IM_f) is shown on the left hand side of (3.19).*

$$\begin{bmatrix} - & \boldsymbol{1} & 1 & 0 \\ \boldsymbol{1} & - & 1 & 0 \\ 1 & 1 & - & 0 \\ 0 & 0 & 0 & - \end{bmatrix} \qquad \begin{bmatrix} - & \boldsymbol{1} & 0 & 1 \\ \boldsymbol{1} & - & 0 & 0 \\ 0 & 0 & - & 0 \\ 1 & 0 & 0 & - \end{bmatrix} \tag{3.19}$$

Assume the IM returned by ϕ (IM_ϕ) is shown on the right hand side of Equation (3.19). Entries equal to 1 in both IM_f and IM_ϕ are highlighted in bold. Then, precision $= 1/2$; recall $= 1/3$; F-score $= 2/5$.

3.3.3 Identification of Pairwise Interaction (Q3.1)

In this section, I describe numerical experiments used to evaluate the efficacy of the MEE method in terms of identifying IM.

The MEE is calculated for each of the benchmark problems proposed in Section 3.3.1, and evaluated by the performance metrics introduced in Section 3.3.2. The dimensionality of the benchmark problems are set to $n = 10$. The parameter setting for MEE is $m = 50$, $\epsilon = 0.2$, $\beta = 10^{-3}$, $\delta x = 10^{-6}$. For each benchmark problem and each method, 25 independent runs are conducted. The average *precision*, *recall* and *F-score* values for the 25 independent runs are recorded. The performance of MEE is compared with Differential Grouping (DG) [120], a state-of-art method used to identify variable interactions. The parameter setting for DG is consistent with the original paper.

As part of the analysis, I replace the MIC values with the PCC and the Spearman's Rank Correlation Coefficient (SRCC) when examining the pairwise interactions between decision variables (In Algorithm 1 line 18, simply change $MIC(D, A) > \epsilon$ to $PCC(D, A) > \epsilon$ or $SRCC(D, A) > \epsilon$). I refer to the model using PCC / SRCC as 'PCC' / 'SRCC' in the results section. MEE is compared with PCC and SRCC on the 24 benchmark problems.

Table 3.1: Parameter settings. The values emphasized in bold are the default parameter settings.

Parameters	Selected values
ϵ	$0.1, \mathbf{0.2}, 0.3, 0.4, 0.5, 0.6$
β	$10^{-1}, 10^{-2}, \mathbf{10^{-3}}, 10^{-4}, 10^{-5}, 10^{-6}$
m	$10, \mathbf{50}, 100, 200, 300, 400$
δx	$10^{-3}, 10^{-4}, 10^{-5}, \mathbf{10^{-6}}, 10^{-7}, 10^{-8}$

The parameter setting for PCC and SRCC are exactly the same as those used in the MEE experiments.

The Kruskal-Wallis nonparametric one-way ANOVA test [158] with 95% confidence interval is employed to determine whether at least one method is significantly different from the others. Then a series of Wilcoxon rank-sum tests ($\alpha = 0.05$) with Holm p-value correction [158] are conducted in a pairwise fashion to find the best performing method.

To test the sensitivity of MEE to the parameters, six different values for each parameter are selected, as shown in Table 3.1. The values emphasized in bold are the default parameter setting. To test the sensitivity of MEE to each parameter, I set other parameters to the default values and set this parameter to the six different values one by one. For example, to test the sensitivity of MEE to ϵ, I set $\beta = 10^{-3}$, $m = 50$, $\delta x = 10^{-6}$, and let $\epsilon = \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6\}$ one by one. Thus, $6 \times 4 = 24$ parameter settings were tested. To test the pairwise interaction between parameters, I vary the parameter pair over all possible combinations of values ($6 \times 6 = 36$) and set other parameters to the default values. For each of the parameter settings, MEE is calculated for each of the 24 benchmark problems with 25 independent runs. Therefore, $24 \times 25 = 600$ independent runs are conducted for each parameter setting. The average values for *precision*, *recall* and *F-score* of the 600 independent runs are recorded and used when evaluating the parameter sensitivity of MEE.

3.3.4 Quantification of Variable Interaction (Q3.2)

In this section, I describe numerical experiments used to evaluate the efficacy of the MEE measure in terms of quantifying the level of variable interactions.

The MEE method is used to calculate the three measures: DDVI, DIVI and DVI for each of the 24 benchmark problems using the default parameter setting (highlighted in bold in Table 3.1). The MEE method is compared with Entropic Epistasis (EE) and Meta-model methods (see Section 2.1.2 for a detailed description).

The EE method uses MI to quantify the level of variable interactions. The *k-d* partitioning method is used to estimate the MI between continuous variables [164]. In each step, a set of samples is partitioned by their median along one axis, producing two sub-partitions of equal probability. This process is recursively conducted until each cell has a uniform distribution. The sample size used to estimate n-dimensional entropy should be greater than 2^n [164]. As the 24 benchmark problems are all 10-dimensional, I set the sample size $m = 5000$ for EE.

The Meta-model method builds a linear or quadratic regression model based on m samples. It has been suggested that the sample size m should be greater than $50 + 8n$ [42]. Therefore, I set the sample size $m = 100n$. Both the linear and quadratic regression models with interaction effects are considered in this chapter, and the corresponding adjusted coefficient of determination ($\bar{R}_l^2$ and $\bar{R}_q^2$) are calculated.

For each function and each method, 25 independent runs are conducted to remove randomness, and the mean value of each measure is recorded. For each measure, the Kruskal-Wallis nonparametric one-way ANOVA test [158] with 95% confidence interval is employed to determine whether consistent results can be obtained across the functions in each category.

3.3.5 Correlation with Algorithm Performance (Q3.3)

It is well-known that the interaction between decision variables is a factor that makes an optimization problem difficult for some of the EAs to solve [62, 191, 121]. Thus, it is reasonable to assume that as the level of variable interactions increases in an optimization problem, it may become more difficult for some of the EAs to solve. If a measure successfully quantifies the level of variable interactions, it should be correlated with the performance of some algorithms. In this section, I describe numerical experiments used to investigate the correlation between MEE and algorithm performance.

Table 3.2: Parameter settings for the selected optimizers.

Optimizers	Parameter settings
DE	Population size = 50, mutation factor = 0.1, crossover rate = 0.5, mutation strategy = DE/rand/1, crossover strategy = binomial, maximal iteration = 2000, population initialization = uniform.
GA	Population size = 50, crossover rate = 0.5, mutation rate = 0.1, mutation strategy = adding a small number, crossover strategy = uniform, maximal iteration = 2000, population initialization = uniform.
PSO	Population size = 50, velocity clamping factor = 2, individual learning rate = 2, social parameter = 2, minimal inertia weight = 0.4, maximal inertia weight = 0.9, maximal iteration = 2000, population initialization = uniform.
CMA-ES	Population size = 4 + floor(3*log(n)), parent number = floor(population size/2), maximal FE = 10^5, population initialization = Gaussian distribution.

Three fully separable benchmark problems – Elliptic, Rastrigin and Alpine – are used in this section (see Table A.2). Separable functions are selected, as it is possible to set the level of variable interactions by rotating a subset of the decision variables [121]. To the best of my knowledge, there is no technique available to control the level of variable interactions in a fully non-separable function. For each benchmark problem, I set $n = 100$ and vary the number of interacting decision variables over $\{5i, 0 \leq i \leq 20\}$. Therefore, for each benchmark problem, 21 instances are obtained with different levels of variable interactions (number of interacting decision variables). The optimal solution for all the instances is $f(\mathbf{0}) = 0$. Note that for each benchmark problem, the scale of the 21 instances are the same as rotation does not change the scale of a function.

There are generally two ways to measure algorithm performances. The first way is to use the expected running time, which can be estimated by the expected number of FEs used to find a target solution at the first time. An alternative way is to compare the quality of the best solution found within a given computational budget. In the situation that it is difficult to find a target solution, the second criterion is often used to measure algorithm performance. The benchmark problems used in this section are high-dimensional (100) and difficult for optimizers to solve, therefore, I use the best solution found within the fixed number of FEs as the measurement of algorithm performance.

MEE is used to calculate the DVI for each instance with default parameter setting. I treat direct interaction and indirect interaction as equally important to problem difficulty. Therefore, only the DVI measure is considered in this section. Four widely used algorithms: *CMA-ES* [45], *DE* [163], *PSO* [66] and *GA* [39] are selected to solve each instance. Note that the selection of the four optimizers is biased. Due to the large number of existing algorithms, I can not take all of them into consideration. Here, my goal is simply to see whether insights into relationships between algorithm performance and variable interaction levels as calculated by MEE can be documented. Therefore, I only test the four widely used algorithms with default parameter settings, presented in Table 3.2. Note that the maximal number of FEs used by each optimizer is always 10^5.

For each optimizer and each instance, 25 independent runs are conducted. The mean of the best solution found in the fixed number of FEs is recorded. For each function and each optimizer, the SRCC (r_s), PCC (r_p) and MIC (r_m) between DVI and the mean of best solutions found in the fixed number of FEs are calculated across the 21 instances. To complete the analysis, I also compare results from the MEE with EE and Meta-Model ($\bar{R}_l^2$ and $\bar{R}_q^2$). The statistic tests employed to determine the best performances are the same as for Section 3.3.3.

3.4 Experimental Results

3.4.1 Identification of Pairwise Interaction (Q3.1)

Performances Comparison

Table 3.3 lists the experimental results when MEE was used to identify pairwise interactions between decision variables in the 24 benchmark problems. The performances of MEE is compared with DG, PCC and SRCC in terms of *precision*, *recall*, and *F-score*. The different categories of the benchmark problems are divided by the double line. The best performances are highlighted in bold. I now provide a detailed analysis of the experimental results for each category.

Each of the methods, MEE, DG and PCC, performs equally well on Category I functions (see Section 3.3.1). Category I consists of 4 fully separable functions (f_1 to f_4), each with 10 separable decision variables. The MEE, DG and PCC methods successfully identify all decision variables as being separable. Therefore, each method correctly records a score of 1.00 in terms of *precision*, *recall* and *F-score*. The SRCC method falsely classifies some separable decision variables as non-separable, resulting in a low *precision* and *F-score*.

The MEE method achieves equal or better results than DG, PCC or SRCC on Category II functions. Category II consists of 4 partially separable functions (f_5 to f_8). Each function has 5 separable and 5 non-separable decision variables. The MEE method achieves the best score in terms of *precision*, *recall* and *F-score* on f_5, f_6, f_7. However on f_8, MEE identifies all the 10 decision variables as being non-separable, resulting in a low *precision*. The reason for this is that the computational error for f_8 is large. The DG and PCC methods correctly identify the 5 separable and 5 non-separable decision variables on f_6, f_7. However on f_5, DG and PCC falsely classify some interacting decision variables as independent. On f_8, DG and PCC falsely classify some interacting variables as independent and some independent variables as interacting, resulting in a poor *precision* and *recall*. The SRCC method identifies all the decision variables as being interacting, resulting in a low *precision* and *F-score*.

The MEE method achieves equal or better results than DG, PCC or SRCC on Category III functions. Category III consists of 4 partially separable functions (f_9 to f_{12}). Each function has 5 non-separable components and each component has 2 decision variables. The MEE method accurately identifies all separable and non-separable decision variables on each function. The DG and PCC methods achieve best score on f_{11} and f_{12}. However, on f_9 and f_{10}, the performances of DG and PCC are worse than MEE in terms of *recall* and *F-score*. The SRCC method is outperformed by the other methods on Category III functions.

The MEE method achieves comparable or better results than DG on Category IV functions. Category IV consists of 4 partially separable functions (f_{13} to f_{16}). Each function

Table 3.3: The experimental results of the MEE, DG, PCC and SRCC methods on the 24 benchmark problems in terms of *Precision*, *Recall* and *F-Score*. The mean results based on 25 independent runs are presented. PCC and SRCC methods are employed to show the efficacy of MIC compared with PCC and SRCC. The best performances from the four methods are highlighted in bold (Wilcoxon rank-sum tests ($\alpha = 0.05$) with Holm p-value correction).

Cate	Func	*Precision*				*Recall*				*F-Score*			
		MEE	DG	PCC	SRCC	MEE	DG	PCC	SRCC	MEE	DG	PCC	SRCC
I	f_1	**1.00**	**1.00**	**1.00**	0.00	1.00	1.00	1.00	1.00	**1.00**	**1.00**	**1.00**	0.00
	f_2	**1.00**	**1.00**	**1.00**	0.00	1.00	1.00	1.00	1.00	**1.00**	**1.00**	**1.00**	0.00
	f_3	**1.00**	**1.00**	**1.00**	0.00	1.00	1.00	1.00	1.00	**1.00**	**1.00**	**1.00**	0.00
	f_4	**1.00**	**1.00**	**1.00**	0.00	1.00	1.00	1.00	1.00	**1.00**	**1.00**	**1.00**	0.00
II	f_5	**1.00**	**1.00**	**1.00**	0.22	**1.00**	0.95	0.30	**1.00**	**1.00**	0.97	0.46	0.36
	f_6	**1.00**	**1.00**	**1.00**	0.22	1.00	1.00	1.00	1.00	**1.00**	**1.00**	**1.00**	0.36
	f_7	**1.00**	**1.00**	**1.00**	0.22	1.00	1.00	1.00	1.00	**1.00**	**1.00**	**1.00**	0.36
	f_8	0.22	**0.24**	0.22	0.22	**1.00**	0.26	**1.00**	**1.00**	**0.36**	0.25	**0.36**	**0.36**
III	f_9	**1.00**	**1.00**	**1.00**	0.11	**1.00**	0.99	0.20	**1.00**	**1.00**	0.99	0.33	0.20
	f_{10}	**1.00**	**1.00**	**1.00**	0.11	**1.00**	0.99	0.20	**1.00**	**1.00**	0.99	0.33	0.20
	f_{11}	**1.00**	**1.00**	**1.00**	0.11	1.00	1.00	1.00	1.00	**1.00**	**1.00**	**1.00**	0.20
	f_{12}	**1.00**	**1.00**	**1.00**	0.11	1.00	1.00	1.00	1.00	**1.00**	**1.00**	**1.00**	0.20
IV	f_{13}	0.44	**0.77**	0.44	0.44	**1.00**	0.82	**1.00**	**1.00**	0.61	**0.79**	0.61	0.61
	f_{14}	**1.00**	**1.00**	**1.00**	0.44	**1.00**	0.95	0.35	**1.00**	**1.00**	0.97	0.51	0.61
	f_{15}	**1.00**	**1.00**	**1.00**	0.44	**1.00**	**1.00**	0.80	**1.00**	**1.00**	**1.00**	0.80	0.61
	f_{16}	**1.00**	**1.00**	**1.00**	0.44	1.00	1.00	1.00	1.00	**1.00**	**1.00**	**1.00**	0.61
V	f_{17}	1.00	1.00	1.00	1.00	**1.00**	0.11	**1.00**	0.99	**1.00**	0.20	**1.00**	0.99
	f_{18}	1.00	1.00	1.00	1.00	**1.00**	0.11	0.04	**1.00**	**1.00**	0.20	0.08	**1.00**
	f_{19}	1.00	1.00	1.00	1.00	0.98	0.38	0.33	**1.00**	0.99	0.55	0.50	**1.00**
	f_{20}	**1.00**	**1.00**	0.44	0.44	**1.00**	0.20	**1.00**	**1.00**	**1.00**	0.33	0.61	0.61
VI	f_{21}	1.00	1.00	1.00	1.00	**1.00**	**1.00**	**1.00**	0.93	**1.00**	**1.00**	**1.00**	0.95
	f_{22}	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	f_{23}	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	f_{24}	1.00	1.00	1.00	1.00	**1.00**	**1.00**	**1.00**	0.97	**1.00**	**1.00**	**1.00**	0.98

has 2 non-separable components and each component has 5 decision variables. The MEE method achieves the best score for *precision*, *recall* and *F-score* on f_{14}, f_{15} and f_{16}. On f_{13}, MEE falsely identifies all decision variables as interacting, resulting in a poor *precision* (0.44). The DG method achieves the best *precision*, *recall* and *F-score* on f_{15} and f_{16}. However DG performs poorly on f_{13}, with *precision*=0.77 and *recall*=0.82 (1 is the best score). On f_{14}, DG sometimes fails to capture some interacting decision variables. The PCC and SRCC methods perform worse than MEE on f_{13}, f_{16}.

The MEE method achieves better results than DG or PCC on Category V functions. Category V consists of 4 overlapping functions (f_{17} to f_{20}). The MEE method achieves the best score on f_{17}, f_{18}, f_{20} and nearly best score on f_{19} for *recall* and *F-score*. The DG method is unable to identify all variable interactions in this category. The *recall* rates of the DG method on the 4 overlapping functions are very low (less than 0.40), resulting in a low *F-score*. The *recall* rates of PCC on f_{18} and f_{19} are also very low. On f_{20}, the PCC method falsely classifies some independent variables as interacting, resulting in a low *precision*. The SRCC method achieves comparable results with the MEE method on f_{17}, f_{18} and f_{19}.

The MEE, DG and PCC methods perform equally well on Category VI functions. Category VI consists of 4 fully non-separable functions (f_{21} to f_{24}), each with 10 non-separable decision variables. The MEE, DG and PCC methods accurately identify all the decision variables as non-separable. Therefore, all of them achieve the best score for *precision*, *recall* and *F-score*. The SRCC method performs slightly worse than the other methods.

In summary, the proposed MEE method achieves the best score (1.00) in terms of *precision*, *recall* and *F-score* on 21 out of the 24 benchmark problems. MEE outperforms the DG, PCC and SRCC methods on the 24 benchmark problems. The comparison between MEE and PCC / SRCC confirms that the MIC is more reliable than the PCC or SRCC when identifying functional relationships. The reason is that the PCC can only capture linear relationships and the SRCC can only capture increasing or decreasing relationships.

Parameter Sensitivity

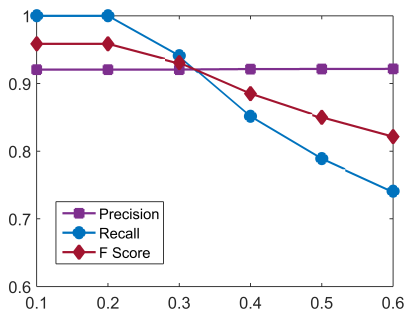
The experimental results for the MEE method with different parameter settings on the 24 benchmark problems are presented in Figure 3.2. Figures 3.2a, 3.2b, 3.2c, and 3.2d present the sensitivity to ϵ, β, m and δx respectively.

Figure 3.2a shows that the MEE method performs well for a wide range of ϵ values. In fact, when ϵ is in the range of 0.1 to 0.3, *precision*, *recall* and *F-score* are greater than 0.90. When ϵ increases, *precision* increases slightly, while *recall* and *F-score* decreases. The reason is that when ϵ increases, the threshold to identify variable interaction becomes higher. Therefore, some pairwise interacting decision variables with small MIC value will be falsely identified as independent, resulting in a decrease in *recall*. On the other hand, some independent pairwise decision variables with large MIC value will be correctly identified as independent, resulting in an increase in *precision*. Consider two extreme cases: a). If $\epsilon = 0$, all pairwise decision variables will be identified as interacting, therefore $recall = 1$; b). If $\epsilon = 1$, all pairwise decision variables will be identified as independent, therefore $recall = 0$, $precision = 1$.

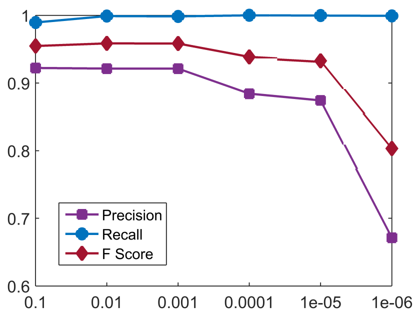
Figure 3.2b shows that the MEE method performs well across a wide range of β values. When β is in the range of 10^{-3} to 10^{-1}, *precision*, *recall*, and *F-score* are greater than 0.90. When $\beta = 10^{-6}$, the *precision* and *F-score* decrease significantly. The reason is that the computational error in the system is generally greater than 10^{-6}. Therefore $\beta = 10^{-6}$ can no longer remove the computational errors from the system. This result suggests that the selection of β value should be greater than 10^{-6}.

Figure 3.2c shows that the performance of the MEE method is not sensitive to the sample size m. When m is in the range of 10 to 400, *precision*, *recall* and *F-score* are all greater than 0.92. Ideally, when m increases, *precision*, *recall* and *F-score* should also increase. However, such phenomenon cannot be observed from Figure 3.2c (The optimal sample size $m = 50$). The explanation for this phenomenon is left for future work.

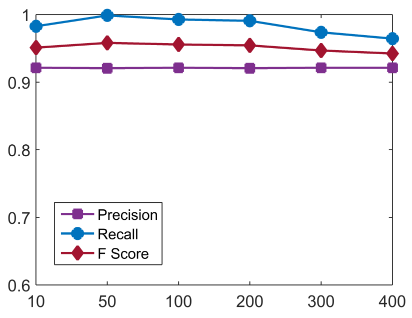
Figure 3.2d shows that the MEE method performs well when δx is in the range of 10^{-6} to 10^{-3} (*precision*, *recall* and *F-score* greater than 0.90). The parameter δx is one of the main factors that result in computational errors in the system. When δx is less than 10^{-6}, *precision* and *F-score* decreases quickly. The reason is that that the computational



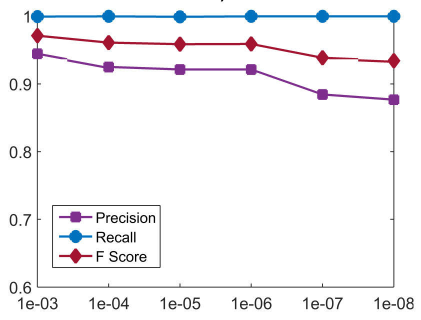
(a) ϵ



(b) β



(c) m



(d) δx

Figure 3.2: Experimental results of the MEE measure with different parameter settings on the 24 benchmark problems. (a) Sensitivity to ϵ; (b) Sensitivity to β; (c) Sensitivity to m; (d) Sensitivity to δx. The horizontal axis represents the corresponding parameter values. The vertical axis represents the averaged *precision*, *recall* and *F-score*, where "□" represents *precision*, "○" represents *recall*, "◇" represents *F-score*.

errors increase significantly when $\delta x < 10^{-6}$. It suggests that the selection of δx should be greater than or equal to 10^{-6}.

The pairwise interactions between parameters are also investigated. The detailed results are not presented in this chapter. However, two conclusions can be drawn from the results: a). No significant pairwise interactions can be observed between parameters except for $(\beta, \delta x)$, and b). The proposed MEE method achieves high accuracy across a wide range of parameter settings when identifying variable interactions.

3.4.2 Quantification of Variable Interaction (Q3.2)

This section presents the experimental results of the proposed MEE measure when quantifying the level of variable interactions in the 24 benchmark problems. The performance of MEE is compared against the EE and Meta-model.

Table 3.4: The experimental results of the MEE, EE and Meta-Model on the 24 benchmark problems in terms of quantifying variable interactions. The mean results based on 25 independents runs are presented. For each method, consistent results across the benchmark problems in each category are highlighted in bold (Kruskal-Wallis one-way ANOVA test with $\alpha = 0.05$).

Category	Function	EE	Meta-Model		MEE		
			$\bar{R}_l^2$	$\bar{R}_q^2$	DVI	DIVI	DDVI
I	f_1	0.29	0.00	1.00	**0.00**	**0.00**	**0.00**
	f_2	1.21	0.00	0.99	**0.00**	**0.00**	**0.00**
	f_3	2.01	0.25	0.90	**0.00**	**0.00**	**0.00**
	f_4	0.22	0.07	0.59	**0.00**	**0.00**	**0.00**
II	f_5	0.53	0.00	0.99	0.22	0.00	0.22
	f_6	1.38	0.23	0.93	0.22	0.00	0.22
	f_7	2.38	0.19	0.74	0.22	0.00	0.22
	f_8	0.82	0.06	0.28	1.00	0.29	0.71
III	f_9	0.05	0.00	0.05	**0.11**	**0.00**	**0.11**
	f_{10}	0.70	0.46	0.85	**0.11**	**0.00**	**0.11**
	f_{11}	0.57	0.25	0.89	**0.11**	**0.00**	**0.11**
	f_{12}	0.78	0.36	0.86	**0.11**	**0.00**	**0.11**
IV	f_{13}	1.25	0.18	0.46	1.00	0.25	0.75
	f_{14}	2.12	0.19	0.72	0.44	0.00	0.44
	f_{15}	0.60	0.21	0.73	0.44	0.00	0.44
	f_{16}	0.95	0.00	0.99	0.44	0.00	0.44
V	f_{17}	1.83	0.35	0.71	**1.00**	0.16	0.84
	f_{18}	0.54	0.15	0.54	**1.00**	0.80	0.20
	f_{19}	0.42	0.02	0.66	**1.00**	0.36	0.64
	f_{20}	1.53	0.36	0.74	**1.00**	0.38	0.62
VI	f_{21}	0.63	0.18	0.70	**1.00**	**0.00**	**1.00**
	f_{22}	0.72	0.05	0.28	**1.00**	**0.00**	**1.00**
	f_{23}	0.50	0.26	0.88	**1.00**	**0.00**	**1.00**
	f_{24}	0.76	0.16	0.50	**1.00**	**0.00**	**1.00**

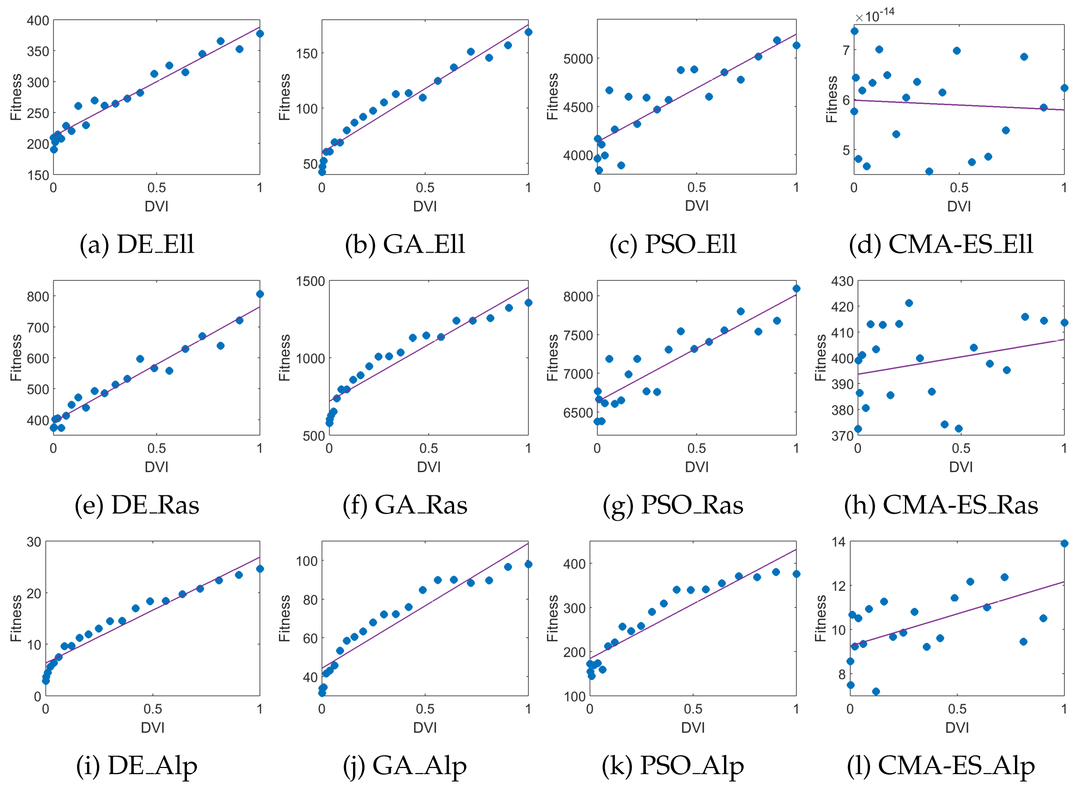
As shown in Table 3.4, consistent results can be obtained by MEE for each category in most cases, while EE and Meta-Model ($\bar{R}_l^2$, $\bar{R}_q^2$) generate diverse results. In Category I, MEE identifies all of the 4 functions as being fully separable ($DVI = 0$), while the results obtained by EE and Meta-Model are diverse: EE ranging from 0.22 to 2.01; $\bar{R}_l^2$ ranging from 0.00 to 0.25; $\bar{R}_q^2$ ranging from 0.59 to 1.00. In Category II, MEE identifies all the functions as partially separable with $DVI = 0.22$ expect for f_8. On f_8, MEE unexpectedly identifies all decision variables as interacting. In Category III, the MEE measure identifies all the functions as partially separable with $DVI = 0.11$, while I cannot observe any consistent results obtained by EE or Meta-Model. In Category IV, the DVI obtained by MEE for f_{14}, f_{15} and f_{16} are all 0.44, while for f_{13}, $DVI = 1.00$. The reason why MEE identifies all decision variables as being interacting in f_{13} is the same with f_8. In Category V and VI, MEE identifies all the functions as being fully non-separable ($DVI = 1.00$). However, Category V is fully non-separable with indirect interaction ($DIVI > 0$), while Category VI is fully non-separable only with direct interaction ($DIVI = 0$). The results obtained by EE, $\bar{R}_l^2$ and $\bar{R}_q^2$ for Category V and VI are diverse. In summary, MEE is more robust and accurate when compared against EE and Meta-Model when quantifying variable interactions in a BCOP.

3.4.3 Correlation with Algorithm Performances (Q3.3)

Figure 3.3 shows the linear regression between the DVI and the mean of the best solutions obtained by each algorithm when solving the Elliptic/Rastrigin/Alpine benchmark problems. The corresponding PCC (r_p), SRCC (r_s) and MIC (r_m) values are listed in Table 3.5.

High correlation can be observed between DVI and the mean of best solutions found by DE/GA/PSO on the three benchmark problems. In fact, the r_p, r_s and r_m are close to 1 in most cases. The plot shows that when DVI increases from 0 to 1, the benchmark problem becomes more difficult for DE/GA/PSO to solve.

Take DE as an example, there is a strictly increasing relationship between DVI and the mean of best solutions found on Alpine benchmarks (Figure 3.3i). The easiest function is the one with $DVI = 0$, and the hardest function is the one with $DVI = 1$. On the



(a) DE_Ell (b) GA_Ell (c) PSO_Ell (d) CMA-ES_Ell

(e) DE_Ras (f) GA_Ras (g) PSO_Ras (h) CMA-ES_Ras

(i) DE_Alp (j) GA_Alp (k) PSO_Alp (l) CMA-ES_Alp

Figure 3.3: The linear regression results for the DVI and the mean of the best solutions (fitness) found by each algorithm on each of the benchmark problems. (Top) Elliptic function; (Middle) Rastrigin function; (Bottom) Alpine function. The vertical axis represents the mean of the best solutions found by the algorithm for a benchmark problem. The horizontal axis represents the corresponding DVI returned by MEE.

Elliptic and Rastrigin functions, there is a high correlation between DVI and the mean of best solutions found ($r_s = 0.98/0.97$). Similar conclusions can be drawn for GA and PSO. However for CMA-ES, there is no significant correlation between DVI and the mean of best solutions found on Elliptic/Rastrigin benchmarks ($r_s = -0.11/0.35$). The reason for this is that CMA-ES is designed to be rotationally invariant. Therefore, it performs equally well on separable and non-separable functions [48].

I cannot observe significant correlation between EE and the mean of the best solutions found by DE/GA/PSO/CMA-ES on the three benchmark problems. Therefore, EE may not be a reliable measure in terms of quantifying interactions between continuous variables.

The correlations between $\bar{R}_l^2$ and the mean of the best solutions found by DE/GA/PSO

Table 3.5: The correlation between DVI/EE/$\bar{R}_l^2$/$\bar{R}_q^2$ and the mean of the best solutions found by each optimizer on each benchmark problem. The r_p represents the PCC; r_s represents the SRCC; r_m represents the MIC. The highest correlation (absolute value) is highlighted in bold.

Func	Meas	DE			GA			PSO			CMA-ES		
		r_p	r_s	r_m	r_p	r_s	r_m	r_p	r_s	r_m	r_p	r_s	r_m
Ell	DVI	**0.97**	**0.98**	**0.91**	**0.97**	**0.99**	**0.99**	**0.87**	**0.86**	**0.76**	-0.07	-0.11	0.19
	EE	0.38	0.29	0.37	0.50	0.37	0.40	0.46	0.29	0.37	**-0.13**	-0.08	0.24
	$\bar{R}_l^2$	0.32	0.36	0.38	0.35	0.38	0.38	0.47	0.48	0.38	**-0.13**	**-0.15**	0.24
	$\bar{R}_q^2$	-0.92	**-0.98**	**0.91**	-0.96	-0.98	**0.99**	-0.86	-0.85	**0.76**	0.11	0.10	**0.30**
Ras	DVI	**0.97**	**0.97**	0.90	**0.95**	**0.99**	**0.99**	**0.90**	**0.88**	0.76	0.28	0.35	0.33
	EE	0.64	0.62	0.52	0.67	0.63	0.50	0.67	0.57	0.57	0.38	**0.46**	0.36
	$\bar{R}_l^2$	-0.81	-0.89	0.91	-0.86	-0.89	0.91	-0.76	-0.75	0.76	**-0.41**	-0.40	**0.41**
	$\bar{R}_q^2$	-0.67	-0.93	**0.99**	-0.81	-0.95	**0.99**	-0.67	-0.85	**0.81**	-0.36	-0.27	0.26
Alp	DVI	0.96	**1.00**	**0.99**	0.94	**0.98**	**0.99**	0.93	**0.97**	**0.99**	**0.59**	**0.56**	0.37
	EE	0.58	0.57	0.50	0.54	0.58	0.50	0.46	0.48	0.37	0.22	0.02	0.31
	$\bar{R}_l^2$	0.87	0.87	0.86	0.87	0.85	0.84	0.85	0.84	0.84	0.37	0.42	**0.50**
	$\bar{R}_q^2$	**-0.98**	-0.99	**0.99**	**-0.98**	-0.97	**0.99**	**-0.96**	-0.96	**0.99**	-0.58	-0.55	0.37

on Rastrigin and Alpine benchmark problems are generally significant. However paradoxically, on the Rastrigin function the correlation is negative ($r_s < 0$), while on Alpine function the correlation is positive ($r_s > 0$).

The $\bar{R}_q^2$ measure performs well on the three benchmark problems. Note that the correlation between $\bar{R}_q^2$ and the mean of the best solutions found by each optimizer should be negative ($r_s < 0$). The reason for this is that as the DVI increases, a BCOP becomes more complex, therefore the model accuracy $\bar{R}_q^2$ decreases. However I observe that the model accuracy $\bar{R}_q^2$ varies significantly across different benchmark problems with the same DVI. For example the model accuracy $\bar{R}_q^2 = 0.99$ on fully separable Elliptic function; while $\bar{R}_q^2 = 0.48$ on fully separable Alpine function. The reason for this is that, apart from variable interactions, other important problem characteristics e.g. modality also have significant impacts on the model accuracy $\bar{R}^2_q$ [94]. This observation is consistent with the results shown in Table 3.4.

3.5 Algorithm Design Using Maximum Entropic Epistasis

In this section, I investigate how the DVI within a BCOP can be used to determine the choice of algorithms used. Here, I examine the overall optimization performance when the MEE measure is embedded into an algorithm design framework, therefore addressing the second research question ($\mathfrak{RQ}3.2$).

The Separability Prototype for Automatic Memes (SPAM) [18] framework consists of two stages: a separability analysis stage and an optimization stage. In the first stage, the level of variable interactions is approximated. Then the SPAM framework selects appropriate operator(s) to guide the search based on the level of variable interactions. In the original paper [18], two evolutionary operators are employed – the S operator [177] and the R operator [149]. The S operator searches along each dimension, which is efficient to solve separable problems. The R operator (which is the Rosenbrock algorithm) perturbs all the variables simultaneously to follow the gradient of the fitness landscape. The R operator is designed for non-separable optimization problems. In this chapter, I use the same candidate operators (R and/or S) used by Caraffini et al [18]. Note that other efficient algorithms for separable or non-separable problems can also be used as the candidate operators.

In the original SPAM framework, CMA-ES is employed to obtain the covariance matrix (C) within a given computational budget. Then the PCC (r_p) between each pair of decision variables is calculated:

$$r_p(i,j) = \frac{C_{i,j}}{\sqrt{C_{i,i}C_{j,j}}}. \tag{3.20}$$

The $|r_p(i,j)|$ is normalized to 0 if $0 \leq |r_p(i,j)| < 0.2$, to 0.3 if $0.2 \leq |r_p(i,j)| < 0.4$, to 0.5 if $0.4 \leq |r_p(i,j)| < 0.6$, to 0.7 if $0.6 \leq |r_p(i,j)| < 0.8$, to 1 if $0.8 \leq |r_p(i,j)| \leq 1$. Then an index ξ is calculated based on the normalized matrix $\bar{r}_p$ as the approximation of the level of variable interactions:

$$\xi = \frac{2}{n(n-1)} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \bar{r}_p(i,j). \tag{3.21}$$

The index ξ is then employed to calculate the activation probabilities of the S and R operators. Note that the best solution found by CMA-ES in this stage (separability analysis stage) will be passed to the following stage as the initial point.

In the subsequent optimization stage, the index ξ is used to assign an activation probability to each operator. If $\xi = 0$, the optimization is fully separable, therefore only the S operator is used to solve the problem. If $\xi \geq 0.5$, the BCOP is regarded as fully non-separable and only the R operator is used to guide the search. If $0 < \xi < 0.5$, the BCOP is considered as partially separable and two operators coexist with activation probabilities defined as follows:

$$p_s = 1 - 2\xi, \quad p_r = 2\xi. \tag{3.22}$$

The original SPAM framework has been tested on multiple benchmark suites and achieved good results compared with the-state-of-the-art optimization algorithms [18]. However, the framework does have some limitations:

1. In the original SPAM framework, the PCC is used to identify variable interactions, which is not reliable as I have shown in Section 3.4.1.

2. Normalizing the PCC matrix $|r_p|$ to $\bar{r}_p$ adds bias.

3. It is not reasonable to consider the BCOP as fully non-separable when $\xi \geq 0.5$.

To address these limitations, the MEE measure can be used to quantify the level of variable interactions, and the DVI value generated by MEE can be used to assign the activation probabilities to the S and R operators:

$$p_s = 1 - DVI, \quad p_r = DVI. \tag{3.23}$$

Given the discussion above, I test this new version of SPAM framework where MEE is embedded, denoted as SPAM_MEE. Note that the best solution found in the separability analysis stage (MEE stage) will also be passed to the optimization stage.

The results obtained by SPAM_MEE are compared against the results generated using SPAM and SPAM0.5 on the 24 benchmark problems (Table 3.6). SPAM0.5 is a version of SPAM with the activation probabilities of the S and R operators both equal to 0.5. To save computational cost in the separability analysis stage, the sample size used in MEE was set to $m = 10$, while other parameters are the default values highlighted in Table 3.1. Therefore the number of FEs used by MEE is $10n(n-1)$. The maximum number of FEs was set to $3 \times 10^3 n$, divided between the separability analysis stage and optimization stage. Other parameter settings are consistent with the original paper [18]. To complete the analysis, I also compare SPAM_MEE with CMA-ES [45]. The parameter setting for CMA-ES is consistent with the one presented in Table 3.2, but the maximum number of FEs was set to $3 \times 10^3 n$.

For each algorithm and benchmark problem combination, 25 independent runs were carried out. The mean and standard deviation of the best solutions found within the given computational budget are recorded and shown in Table 3.6. The two-sided Wilcoxon test ($\alpha = 0.05$) with Holm p-value correction [158] was used to determine the best performance from SPAM_MEE, SPAM and SPAM0.5 in a pairwise fashion. The best performances are highlighted in bold in Table 3.6. I observed that SPAM_MEE achieves equal or better results than SPAM and SPAM0.5 on the 24 benchmark problems. Note that SPAM_MEE uses more FEs than SPAM or SPAM0.5 in the separability analysis stage. Therefore, the number of FEs used by SPAM_MEE in the optimization stage is less than that used by SPAM or SPAM0.5. This finding suggests that it is worth assigning a portion of computational budget to gain an insight into the fitness landscape of a BCOP. The information, in turn, may guide the search towards better region of the fitness landscape. The SPAM_MEE achieves comparable results with CMA-ES when solving the 24 benchmark problems.

3.6 Conclusion

In this chapter, I have investigated the effects of interactions between decision variables in BCOPs. Here, the overarching goal was to examine whether it was possible to improve

Table 3.6: The experimental results of SPAM_MEE, SPAM and SPAM0.5 on the 24 benchmark problems. The best performances of the 3 algorithms are highlighted in bold. The performance of SPAM_MEE is also compared with CMA-ES. The better results obtained by SPAM_MEE and CMA-ES is marked with † (two-sided Wilcoxon rank-sum test with the confidence interval of 95%).

Func	SPAM_MEE		SPAM		SPAM0.5		CMA-ES	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std
f_1	6.89e-62	3.35e-62	8.60e-62	8.38e-62	8.60e-62	7.92e-62	8.19e-62	7.46e-62
f_2	5.68e-60†	1.01e-59	5.79e-60	9.16e-60	5.79e-60	8.58e-60	2.04e-59	7.82e-59
f_3	**1.04e+02**	4.38e+01	1.49e+02	7.28e+01	1.30e+02	5.63e+01	2.11e+01†	1.20e+01
f_4	**3.34e-14**†	2.59e-14	1.24e-13	1.33e-13	1.66e-13	1.25e-13	5.50e-13	1.92e-12
f_5	**1.27e+01**	4.81e+00	1.55e+01	4.26e+00	1.46e+01	4.72e+00	1.22e+01	5.85e+00
f_6	1.80e-45	8.74e-45	1.19e-45	2.76e-45	6.60e-39	3.30e-38	8.16e-47	2.28e-46
f_7	**9.98e+01**	4.73e+01	1.69e+02	1.01e+02	1.31e+02	9.15e+01	1.14e+02	6.70e+01
f_8	9.30e+00	3.77e+00	7.92e+00	3.32e+00	8.23e+00	3.84e+00	8.46e+00	4.57e+00
f_9	**7.73e-01**†	1.83e-01	1.44e+00	4.21e-01	1.46e+00	3.64e-01	1.32e+00	3.24e-01
f_{10}	**1.36e+02**	1.40e+02	4.07e+03	4.42e+03	3.76e+03	3.28e+03	1.21e+01†	1.11e+01
f_{11}	6.33e-01	2.48e-01	6.35e-01	3.12e-01	6.92e-01	2.54e-01	5.25e-01	2.76e-01
f_{12}	**1.39e-01**†	5.60e-01	4.71e-01	1.06e+00	7.68e-01	1.83e+00	3.96e-01	1.21e+00
f_{13}	1.41e+01	7.78e+00	1.21e+01	4.48e+00	1.32e+01	5.45e+00	1.17e+01	8.34e+00
f_{14}	2.00e-01†	1.09e-01	2.80e-01	2.06e-01	3.80e-01	2.46e-01	3.63e-01	2.17e-01
f_{15}	8.09e+01	4.09e+01	9.44e+01	7.53e+01	7.48e+01	5.41e+01	1.58e+01†	1.25e+01
f_{16}	**3.06e+00**	2.57e+00	3.85e+00	4.32e+00	3.19e+00	1.71e+00	2.35e-12†	6.40e-12
f_{17}	**4.55e+00**	1.73e+01	3.58e+01	1.08e+02	5.60e+01	1.21e+02	2.02e-01†	8.00e-01
f_{18}	**3.08e+00**†	6.74e-01	4.37e+00	1.11e+00	4.46e+00	1.30e+00	4.12e+00	1.20e+00
f_{19}	**2.73e+01**†	2.17e+01	7.38e+01	3.52e+01	8.26e+01	2.35e+01	4.77e+01	4.53e+01
f_{20}	**2.86e+00**	9.43e+00	4.99e+01	8.18e+01	2.80e+02	5.89e+02	1.33e+00	2.24e+00
f_{21}	**1.55e+02**	6.81e+01	2.30e+02	1.31e+02	3.17e+02	1.83e+02	2.98e+01†	1.86e+01
f_{22}	5.38e+01	1.22e+01	5.17e+01	1.05e+01	5.43e+01	1.52e+01	4.73e+01	1.43e+01
f_{23}	**3.24e-18**	4.19e-18	5.32e-18	1.35e-17	2.15e-15	1.07e-14	2.05e-18†	5.88e-18
f_{24}	3.54e+02	2.22e+02	3.49e+02	2.24e+02	3.25e+02	1.68e+02	2.97e+02	1.63e+02

the performance of an optimization algorithm by guiding the search based on the level of variable interactions. A robust ELA measure for BCOPs, MEE, was introduced. MEE identifies the IM of decision variables, which is then used to generate an accurate measure of the DVI (encapsulating both direct and indirect variable interactions) spanning a suite of benchmark BCOPs. I have shown that the solution quality found by an EA is correlated with the level of variable interaction in a given problem. This observation suggests that fitness landscape characteristic captured by MEE can be used as a source of information to predict algorithm performance. Significantly, when the MEE measure was embedded into an optimzation algorithm design framework, the performance of my model was at least as good, and in many cases outperformed, the SPAM model on the suite of benchmark BCOPs. These results confirm my hypothesis that quantifying the level of variable interactions can be used to guide the search towards better regions of the fitness landscapes.

Chapter 4

Extended Differential Grouping for Large Scale Optimization

LARGE-SCALE Black-box Continuous Optimization Problems (BCOPs) are very difficult for Evolutionary Algorithms (EAs) to solve [121, 191, 29]. The Cooperative Co-evolution (CC) framework has been used with some success when 'scaling up' EAs to tackle large-scale BCOPs [120, 197, 20]. It divides a large-scale BCOP into a number of sub-problems, and typically uses an EA to solve each sub-problem cooperatively, as described in Section 2.2.3. A number of studies have shown that the strategy used to decompose a large-scale BCOP into sub-problems can have a significant impact on the performance of a CC framework (e.g., [120, 19, 78]). Ideally, the interacting decision variables should be assigned to the same sub-problem, and the interaction between different sub-problems should be kept to minimal.

In this chapter, I investigate the two research questions of this thesis (presented in Section 1.1) in the large-scale BCOP domain: $\mathfrak{RQ}$4.1 *How can variable interactions be accurately identified in a given large-scale BCOP?* $\mathfrak{RQ}$4.2 *How can variable interactions be used to effectively solve a given large-scale BCOP?* Specifically, I explore the effects of variable interactions and decomposition methods within the CC framework when solving large-scale BCOPs.

To answer $\mathfrak{RQ}$4.1, I show that a recently proposed decomposition method – *Differential Grouping* (DG) [120] – can only capture *direct interaction* and fails to capture *indirect interaction* between decision variables (see Section 3.2 for formal definitions). Subsequently I propose an *eXtended Differential Grouping* (XDG) method to cater for different forms of variable interactions. In XDG, after allocating decision variables that interact directly to nominated components, "overlaps" between components are identified. That is, when an overlap is observed, the components that contain the same decision variable(s) are merged. This searching–merging technique is employed to capture indirect interaction between decision variables. The efficacy of XDG is evaluated using large-scale benchmark BCOPs [172]. The experimental results show that XDG achieves perfect decomposition on all of the benchmark problems investigated.

To answer $\mathfrak{RQ}4.2$, XDG is embedded into a CC framework to solve large-scale BCOPs. Empirical studies show that on the benchmark problems with indirect interaction, the proposed approach achieves statistically significantly better solution quality than the CC with DG. On the benchmark problems without indirect interaction, the proposed approach achieves comparable results with the CC with DG.

This chapter is organized as follows. Section 4.1 describes the DG method in detail. Section 4.2 shows the limitation of DG and proposes an extension to address this limitation. Section 4.3 sets up experiments to evaluate the proposed XDG method. Section 4.4 presents and analyses the experimental results. Section 4.5 concludes this chapter.

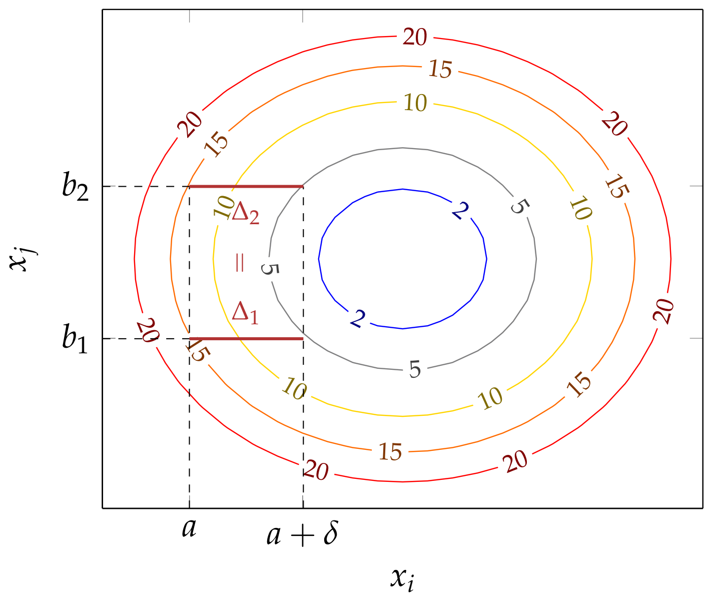
4.1 Related Work: Differential Grouping

In this section, I describe the DG method introduced by Omidvar et al. [120] in detail. The DG method is one of the first automatic decomposition methods, which decomposes a large-scale BCOP based on the underlying structure of variable interactions. It identifies variable interactions according to the following theorem [120]:

Theorem 4.1. *Let $f\colon \mathbb{R}^n \to \mathbb{R}$ be an additively separable function. Two decision variables x_i and x_j interact with each other if there exist real numbers l, l_1, l_2 and δx_i such that*

$$f(\mathbf{x})|_{x_i=l+\delta x_i, x_j=l_1} - f(\mathbf{x})|_{x_i=l, x_j=l_1} \neq f(\mathbf{x})|_{x_i=l+\delta x_i, x_j=l_2} - f(\mathbf{x})|_{x_i=l, x_j=l_2}. \tag{4.1}$$

This theorem states that when the change of fitness values caused by adding a perturbation δx_i to decision variable x_i varies for different values of x_j, x_i interacts with x_j. The rationale of DG when used to identify interacting decision variables is shown in Fig. 4.1.



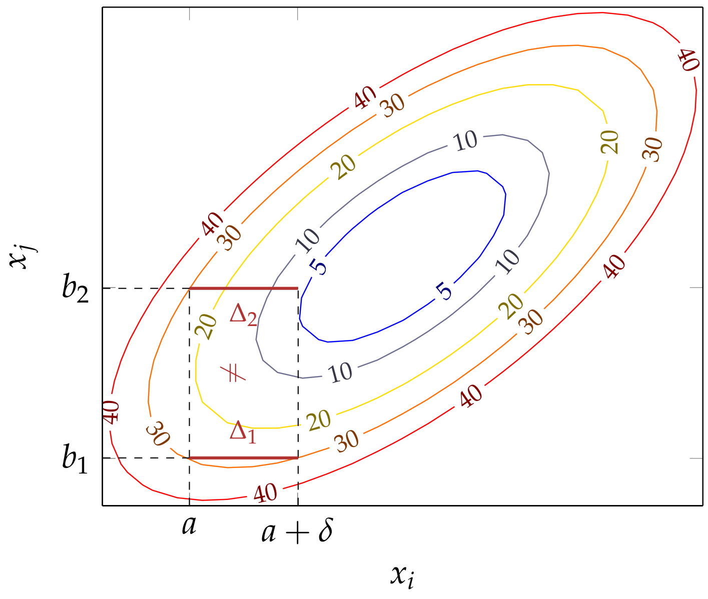


Figure 4.1: The rationale of DG when used to identify separable and non-separable decision variables. The top figure is a separable contour plot; while the bottom figure is a non-separable contour plot. In the top figure, the fitness change induced by adding a perturbation δ to decision variable x_i is the same for different values of x_j. Therefore, decision variable x_i and x_j are separable. However in the bottom figure, the fitness change induced by perturbing x_i varies for different values of x_j; therefore, x_i and x_j interact.

With this theorem, the DG method starts by examining the interaction between the first decision variable x_1 with each of the remaining decision variables. If there exist some interactions between x_1 and the remaining decision variables, these interacting decision variables will be excluded from the remaining variables and placed into the same non-separable group as x_1. If no interaction is identified, the first decision variable x_1 will be placed into the separable group and the algorithm will move on to the next decision variable that has not been grouped. This process is repeated until all of the decision variables have been grouped.

4.2 Extended Differential Grouping

In this section, I show that the DG method cannot identify *indirect interaction*. Subsequently, an extension of DG, named XDG, is proposed to address this limitation.

4.2.1 Limitation of Differential Grouping

Proposition 4.1. *Let f: $\mathbb{R}^n \to \mathbb{R}$ be a differentiable function. If for any candidate solution* $\mathbf{x}$*,*

$$\frac{\partial^2 f(\mathbf{x})}{\partial x_i \partial x_j} = 0, \tag{4.2}$$

then for any real numbers l,l_1,l_2 and δx_i, the following equality holds:

$$f(\mathbf{x})|_{x_i=l+\delta x_i, x_j=l_1} - f(\mathbf{x})|_{x_i=l, x_j=l_1} = f(\mathbf{x})|_{x_i=l+\delta x_i, x_j=l_2} - f(\mathbf{x})|_{x_i=l, x_j=l_2}. \tag{4.3}$$

Proof. By integrating both sides of Eq. (4.2) with respect to x_j,

$$\frac{\partial f(\mathbf{x})}{\partial x_i} = \int 0 \, \mathrm{d}x_j = g(\mathbf{x}_1), \tag{4.4}$$

where $g(\mathbf{x}_1)$ denotes a function of $\mathbf{x}_1$, and $\mathbf{x}_1$ denotes the decision vector of a subset of decision variables $\mathbf{X}_1 \subset \mathbf{X}$, such that $x_j \notin \mathbf{X}_1$. This states $\partial f(\mathbf{x})/\partial x_i$ is not a function of x_j.

Therefore for any real numbers l_1 and l_2, the following equality holds:

$$\left.\frac{\partial f(\mathbf{x})}{\partial x_i}\right|_{x_j=l_1} = \left.\frac{\partial f(\mathbf{x})}{\partial x_i}\right|_{x_j=l_2}. \tag{4.5}$$

By integrating both sides of Eq. (4.5) with respect to x_i from l to $l+\delta x_i$, where l and δx_i are any real numbers,

$$\left.\int_l^{l+\delta x_i} \frac{\partial f(\mathbf{x})}{\partial x_i}\,\mathrm{d}x_i\right|_{x_j=l_1} = \left.\int_l^{l+\delta x_i} \frac{\partial f(\mathbf{x})}{\partial x_i}\,\mathrm{d}x_i\right|_{x_j=l_2}. \tag{4.6}$$

Therefore Eq. (4.3) holds and the proposition is proved. □

If two decision variables x_i and x_j interact indirectly, Eq. (4.2) holds according to Definition 3.1 (presented in Chapter 3). As proved in Proposition 4.1, Eq. (4.3) also holds and DG will classify x_i and x_j as independent. Therefore, DG can not capture indirect interaction between decision variables. This is the main limitation of DG.

In the following, a second example is introduced to further describe the limitation of DG.

Example 4.1. *In the objective function $f(\mathbf{x}) := (x_1 - x_2)^2 + (x_2 - x_3)^2$, $\mathbf{x} \in [-1,1]^3$, decision variables x_1 and x_3 interact indirectly. However, DG classifies x_1 and x_3 as independent.*

Proof. On the one hand,

$$\begin{aligned} f(\mathbf{x})|_{x_1=l+\delta x_1, x_3=l_1} - f(\mathbf{x})|_{x_1=l, x_3=l_1} &= (l+\delta x_1 - x_2)^2 + (x_2 - l_1)^2 - (l - x_2)^2 - (x_2 - l_1)^2 \\ &= (l+\delta x_1 - x_2)^2 - (l - x_2)^2, \end{aligned} \tag{4.7}$$

where l, l_1 and δx_1 are any real numbers. On the other hand,

$$\begin{aligned} f(\mathbf{x})|_{x_1=l+\delta x_1, x_3=l_2} - f(\mathbf{x})|_{x_1=l, x_3=l_2} &= (l+\delta x_1 - x_2)^2 + (x_2 - l_2)^2 - (l - x_2)^2 - (x_2 - l_2)^2 \\ &= (l+\delta x_1 - x_2)^2 - (l - x_2)^2, \end{aligned} \tag{4.8}$$

where l, l_2 and δx_1 are any real numbers. Therefore the following equality holds for any real numbers l, l_1, l_2 and δx_1:

$$f(\mathbf{x})|_{x_1=l+\delta x_1, x_3=l_1} - f(\mathbf{x})|_{x_1=l, x_3=l_1} = f(\mathbf{x})|_{x_1=l+\delta x_1, x_3=l_2} - f(\mathbf{x})|_{x_1=l, x_3=l_2}. \tag{4.9}$$

Therefore DG classifies x_1 and x_3 as independent. $\square$

4.2.2 Extended Differential Grouping

Can the limitation of DG be addressed? Definition 3.1 states that two decision variables x_i and x_j interact indirectly if Eq. (4.2) holds and a set of decision variables can be found to link x_i and x_j. Inspired by this, I propose the XDG method to address this limitation.

The inputs to the XDG methods are: f is the objective function; n is the dimensionality; **ub** and **lb** are the upper bounds and lower bounds of the search space; ϵ is the threshold to identify direct interaction between decision variables. The outputs are the separable variable group (*seps*) and the non-separable variable groups (*nonseps*). The *seps* contains *one* group of all separable decision variables. The *nonseps* contains *several* groups of non-separable decision variables. Each group of decision variables will form a component.

The XDG method consists of three stages. The first stage is to identify *direct interaction* between decision variables (Algorithm 2). The second stage is to identify *indirect interaction* between decision variables (Algorithm 3). The third stage is to assign all of *independent* decision variables into the same component (Algorithm 4).

In the first stage (Algorithm 2), the algorithm starts by identifying the decision variables that interact directly with x_1, and placing them in the first group $G(1)$ with x_1. Note that the interaction between decision variables in the same group will not be examined in the following procedure in order to save computational cost. For example if x_1 directly interacts with x_3 and x_4, the interaction between x_3 and x_4 needs not to be checked, as they are already placed into the same group. Secondly, the algorithm examines the direct interaction between x_2 and all of the other decision variables except x_1, as x_1 and x_2 have been examined in the previous procedure. This process is conducted for each of the decision variables and n interacting groups G are formed.

Algorithm 2 XDG: Identifying Direct Interaction

```
Require: $f, n, \mathbf{ub}, \mathbf{lb}, \epsilon$
1: Initialize all the elements in the Interaction Matrix (IM) to 0
2: Set all decision variables to the lower bounds: $\mathbf{x}_{l,l} \leftarrow \mathbf{lb}$
3: Calculate the fitness value: $y_{l,l} \leftarrow f(\mathbf{x}_{l,l})$
4: for $i$ from 1 to $n$ do
5:     Assign decision variable $x_i$ to the $i_{th}$ group $G(i)$
6:     Set the corresponding element of $x_i$ in $\mathbf{x}_{l,l}$ to upper bound: $\mathbf{x}_{u,l} \leftarrow \mathbf{x}_{l,l}$, and
       $\mathbf{x}_{u,l}(x_i) \leftarrow \mathbf{ub}(x_i)$
7:     Calculate the fitness change: $\delta_1 \leftarrow y_{l,l} - f(\mathbf{x}_{u,l})$
8:     for $j$ from $i+1$ to $n$ do
9:         if $IM(i,j)$ is equal to 1 then
10:            Copy decision variable $x_j$ to the group $G(i)$: $G(i) \leftarrow G(i) \cup \{x_j\}$
11:        else
12:            Set $x_j$ of $\mathbf{x}_{l,l}$ to middle of search space: $\mathbf{x}_{l,m} \leftarrow \mathbf{x}_{l,l}$ and $\mathbf{x}_{l,m}(x_j) \leftarrow \big(\mathbf{lb}(x_j) +$
               $\mathbf{ub}(x_j)\big)/2$
13:            Set $x_j$ of $\mathbf{x}_{u,l}$ to middle of search space: $\mathbf{x}_{u,m} \leftarrow \mathbf{x}_{u,l}$ and $\mathbf{x}_{u,m}(x_j) \leftarrow \big(\mathbf{lb}(x_j) +$
               $\mathbf{ub}(x_j)\big)/2$
14:            Calculate the fitness change: $\delta_2 \leftarrow f(\mathbf{x}_{l,m}) - f(\mathbf{x}_{u,m})$
15:            if the non-linearity term $\lambda = |\delta_1 - \delta_2|$ is greater than $\epsilon$ then
16:                $IM(i,j) \leftarrow 1$
17:                Copy decision variable $x_j$ to the group $G(i)$: $G(i) \leftarrow G(i) \cup \{x_j\}$
18:            end if
19:        end if
20:    end for
21:    Set the corresponding entries of IM to 1 for each pair of decision variables in $G(i)$.
22: end for
23: return $G$
```

The technique used to identify direct interaction between decision variables x_i and x_j is the same as the DG method, which is detailed as follows: (1). Set all the decision variables to the lower bounds of the search space ($\mathbf{x}_{l,l}$); (2). Perturb the decision variable x_i of $\mathbf{x}_{l,l}$ from the lower bound to the upper bound, denoted by $\mathbf{x}_{u,l}$; (3). Calculate the fitness difference (δ_1) between $\mathbf{x}_{l,l}$ and $\mathbf{x}_{u,l}$; (4). Perturb the decision variable x_j of $\mathbf{x}_{l,l}$ and $\mathbf{x}_{u,l}$ from the lower bound to the middle between the lower bound and upper bound, denoted by $\mathbf{x}_{l,m}$ and $\mathbf{x}_{u,m}$ respectively; (5). Calculate the fitness difference (δ_2) between $\mathbf{x}_{l,m}$ and $\mathbf{x}_{u,m}$; and (6). If the difference between δ_1 and δ_2 is greater than the predetermined threshold ϵ, decision variables x_1 and x_2 directly interact. The two subscripts of $\mathbf{x}$ denote the values of x_i and x_j respectively: 'l' means lower bound, 'u' means upper bound, and

Algorithm 3 XDG: Identifying Indirect Interaction

```
Require: $G, n$
1: while the total number of decision variables in all the groups G is not equal to $n$ do
2:     for each pair of groups $G(i)$ and $G(j)$ in $G$ do
3:         if there is some overlap between $G(i)$ and $G(j)$: $G(i) \cap G(j) \neq \emptyset$ then
4:             Merge group $G(j)$ into $G(i)$.
5:         end if
6:     end for
7: end while
8: return $G$
```

Algorithm 4 XDG: Identifying Independent Decision Variables

```
Require: $G$
1: Initialize *seps* and *nonseps* as *empty* groups
2: for each group $G(i)$ in $G$ do
3:     if $G(i)$ contains one decision variable then
4:         Add $G(i)$ to *seps*
5:     else
6:         Add $G(i)$ to *nonseps*
7:     end if
8: end for
9: return *seps* and *nonseps*
```

'm' means the middle between the lower bound and upper bound.

In the second stage (Algorithm 3), the algorithm searches for overlaps between the n groups of interacting decision variables (G) formed in the first stage. If the same decision variable appears in two groups, the algorithm merges the two groups into one. This process is repeated until all of the interacting groups are disjoint and the total number of decision variables in all the groups is equal to n. This is employed as the stopping criterion of the second stage.

In the third stage (Algorithm 4), the algorithm counts the number of decision variables in each group $G(i)$. If $G(i)$ contains more than one decision variable, $G(i)$ will be placed into *nonseps* as a non-separable group. If $G(i)$ contains only one decision variable, it will be assigned to the separable group *seps*. Note that assigning all the separable decision variables into one group is arbitrary, which may not be the best decomposition strategy. However, I follow the approach used in DG to group separable variables.

To further expand on the processing stages, consider Example 4.2 and Figure 4.2.

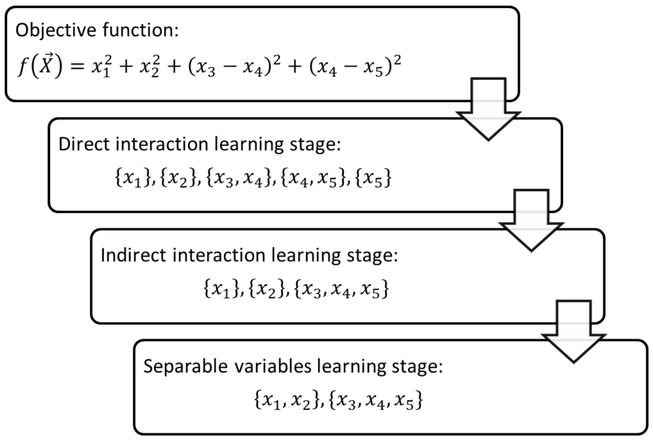

Objective function:
$f(\vec{X}) = x_1^2 + x_2^2 + (x_3 - x_4)^2 + (x_4 - x_5)^2$
Direct interaction learning stage:
$\{x_1\}, \{x_2\}, \{x_3, x_4\}, \{x_4, x_5\}, \{x_5\}$
Indirect interaction learning stage:
$\{x_1\}, \{x_2\}, \{x_3, x_4, x_5\}$
Separable variables learning stage:
$\{x_1, x_2\}, \{x_3, x_4, x_5\}$

Figure 4.2: The decomposition processes used in XDG for a sample objective function. In the direct interaction learning stage, there are 5 components. In the indirect interaction learning stage the number of components is reduced to 3. The separable variable learning stage results in 2 components.

Example 4.2. *Taking the objective function* $f(\mathbf{x}) := x_1^2 + x_2^2 + (x_3 - x_4)^2 + (x_4 - x_5)^2$, $\mathbf{x} \in [-1,1]^5$ *as an example, the three stages of XDG are illustrated in Figure 4.2.*

4.3 Experimental Methodology

In this section, detailed numerical experiments are conducted to investigate the efficacy of XDG. Two questions guide the experimental design:

Q4.1 Can the proposed XDG method address the limitation of the DG method identified in Section 4.2?

Q4.2 Can the proposed XDG method outperform the DG method when it is embedded in a CC framework to solve large-scale BCOPs?

The question Q4.1 refers to the first research question ($\mathfrak{RQ}$4.1); while Q4.2 refers to the second research question ($\mathfrak{RQ}$4.2).

To answer the two questions, benchmark problems from the CEC 2010 special session on large scale global optimization [172] are used. This benchmark suite was also used to evaluate the DG method in [120]. The suite consists of 20 benchmark problems with 5 categories:

1. fully separable problems (f_1 to f_3);
2. partially separable problems with 1 non-separable component (f_4 to f_8);
3. partially separable problems with 10 non-separable components (f_9 to f_{13});
4. partially separable problems with 20 non-separable components (f_{14} to f_{18});
5. fully non-separable problems (f_{19} to f_{20}).

The dimensionality of the 20 benchmark problems is $n = 1000$, and the number of decision variables in each component is 50.

To investigate Q4.1, the proposed XDG method is tested on the CEC'2010 benchmark problems. The threshold ϵ was set to 10^{-1}. (If not specified $\epsilon = 10^{-1}$ in the rest of the chapter). Other values of ϵ (10^{-3} and 1) were used to test the sensitivity of XDG. Two metrics were employed to evaluate the performance of XDG: (1) the number of function evaluations used to decompose the problem; and (2) the percentage of interacting decision variables that are correctly grouped, defined as

Decomposition Accuracy. *Let $G = \{g_1, \dots, g_m\}$ denote the groups of interacting decision variables in a problem f, and $\tilde{G} = \{\tilde{g}_1, \dots, \tilde{g}_n\}$ denote the groups of interacting decision variables that are identified by a decomposition method. Let $G_i = \{g_{i,1}, \dots, g_{i,m}\}$ denote the i_{th} permutation of G, where $1 \leq i \leq m!$, and $\tilde{G}_j = \{\tilde{g}_{j,1}, \dots, \tilde{g}_{j,n}\}$ denote the j_{th} permutation of $\tilde{G}$, where $1 \leq j \leq n!$. The decomposition accuracy (DA) of the decomposition method on f is defined as*

$$DA = \frac{\max\limits_{i,j} \left\{ \sum\limits_{k=1}^{\min\{m,n\}} |g_{i,k} \cap \tilde{g}_{j,k}| \right\}}{\sum\limits_{i=1}^{m} |g_i|}, \tag{4.10}$$

where $|g_i|$ denotes the number of decision variables in g_i.

The performance of XDG is compared with the performance of DG. The value 10^{-3} is selected as the threshold ϵ for DG, as suggested in [120]. If not specified, all the parameter selections are consistent with the original paper. Note that the parameter settings for XDG and DG are different. For DG, $\epsilon = 10^{-3}$ performs better than $\epsilon = 10^{-1}$, as shown in [120]. This is the reason why I select 10^{-3} instead of 10^{-1} as the ϵ value for DG.

To investigate Q4.2, the XDG method is embedded into the Differential Evolution Cooperative Co-evolution (DECC) framework to solve the CEC'2010 benchmark problems. The DECC framework uses a variant of DE – SaNSDE [199] – to optimize each component. The population size is set to 50. The maximal number of function evaluations is set to 3×10^6, divided between the decomposition phase and the evolutionary optimization phase. The performance of XDG is compared with DG and Delta grouping (D) [123], where each decomposition method is incorporated into the DECC. These three algorithms are denoted by DECC-XDG (DECC with XDG), DECC-DG (DECC with DG) and DECC-D (DECC with delta grouping). For each algorithm, 25 independent runs are conducted for each benchmark problem. The mean and standard deviation of the best solutions found in the fixed number of function evaluations are recorded to evaluate the performance of the algorithms.

The Kruskal-Wallis nonparametric one-way ANOVA test [158] with 95% confidence interval was used to determine whether the performance of at least one algorithm was significantly different from the others. Then a series of Wilcoxon rank-sum tests (significance level α=0.05) with Holm p-value correction [158] was conducted in a pairwise fashion to find the better performing algorithm.

4.4 Experimental Results

Comprehensive experimental results are presented and discussed in this section. Section 4.4.1 presents the decomposition comparison between the XDG and DG methods, thus addressing Q4.1. Section 4.4.2 presents the optimization comparison between the XDG, DG and Delta grouping methods when embedded into the DECC framework to solve the benchmark problems, thus addressing Q4.2.

4.4.1 Performance of XDG

The decomposition results of XDG and DG on the CEC'2010 benchmark problems are presented in Table 4.1, which are separated by "/". The different categories of benchmark problems are separated by the double lines. The first 4 columns are the details for each benchmark problem, and the last 6 columns are the results of the decomposition methods on each benchmark problem. Specifically, the ninth column records the number of function evaluations used by XDG/DG to decompose each benchmark problem. The last column records the accuracy of XDG/DG when grouping interacting variables into the same component on each benchmark problem.

Table 4.1 shows that XDG outperforms DG. In terms of decomposition accuracy, XDG achieves 100% accuracy on all of the CEC'2010 benchmark problems. DG achieves 100% accuracy only on 13 benchmark problems. On problems f_{18} and f_{20}, the decomposition accuracy of DG is very low (less than 30%). Note that the number of function evaluations used by XDG is equal or larger than that used by DG. The extra function evaluations are used to identify indirect interaction between decision variables.

The first category of the CEC'2010 benchmark problems (see Section 4.3) contains three fully separable problems (f_1 to f_3). Both XDG and DG successfully identify all of the 1000 decision variables as separable. Note that the function evaluations used by XDG/DG on the three separable problems are the same, which is 1001000/1001000.

Category 2 consists of 5 partially separable problems (f_4 to f_8). Each problem contains one non-separable group with 50 decision variables. On f_5 f_6 and f_7, XDG successfully identifies all the 50 interacting variables and 950 separable variables. On f_4, XDG identifies all of the 1000 decision variables as interacting variables. It may seem odd that the decomposition accuracy is still reported as 100%. The reason is that XDG (or DG) only focuses on grouping interacting variables. As long as all of the interacting variables are correctly grouped, the decomposition accuracy is 100%. This criterion is suggested in [120]. On f_8, XDG forms two non-separable groups, one with 50 interacting variables and one with 950 separable variables. Note that XDG (or DG) automatically places all of the separable variables into the same component. Therefore, the unexpected grouping of 950 separable variables will not affect the final decomposition result on f_8. DG also achieves perfect decomposition on f_5 and f_6. On f_7, the number of interacting variables identified by DG is 34 out of 50. Therefore the decomposition accuracy of DG on f_7 is 68%. On f_8, the decomposition accuracy of DG is 90%, with 45 out of 50 interacting variables successfully identified.

Table 4.1: Decomposition results of XDG and DG on the CEC'2010 large-scale benchmark problems. XDG and DG values are separated by "/".

				Extended Differential Grouping ($\epsilon = 10^{-1}$) / Differential Grouping ($\epsilon = 10^{-3}$)					
Function	Sep Vars	Non-sep Vars	Non-Sep Groups	Captured Sep Vars	Captured Non-sep Vars	Formed Non-sep Groups	Misplaced Vars	Function Evaluations	Decomposition Accuracy
f_1	1000	0	0	1000/1000	0/0	0/0	0/0	1001000/1001000	100%/100%
f_2	1000	0	0	1000/1000	0/0	0/0	0/0	1001000/1001000	100%/100%
f_3	1000	0	0	1000/1000	0/0	0/0	0/0	1001000/1001000	100%/100%
f_4	950	50	1	0/33	50/50	1/10	0/0	80526/14554	100%/100%
f_5	950	50	1	950/950	50/50	1/1	0/0	998648/905450	100%/100%
f_6	950	50	1	950/950	50/50	1/1	0/0	998648/906332	100%/100%
f_7	950	50	1	950/248	50/34	1/4	0/16	998648/67742	100%/68%
f_8	950	50	1	0/134	50/45	2/5	0/5	121658/23286	100%/90%
f_9	500	500	10	500/500	500/500	10/10	0/0	977480/270802	100%/100%
f_{10}	500	500	10	500/500	500/500	10/10	0/0	977480/272958	100%/100%
f_{11}	500	500	10	500/501	500/499	10/10	0/1	978528/270640	100%/99.8%
f_{12}	500	500	10	500/500	500/500	10/10	0/0	977480/271390	100%/100%
f_{13}	500	500	10	500/131	500/159	10/34	0/341	1000154/50328	100%/31.8%
f_{14}	0	1000	20	0/0	1000/1000	20/20	0/0	953960/21000	100%/100%
f_{15}	0	1000	20	0/0	1000/1000	20/20	0/0	953962/21000	100%/100%
f_{16}	0	1000	20	0/4	1000/996	20/20	0/4	956286/21128	100%/99.6%
f_{17}	0	1000	20	0/0	1000/1000	20/20	0/0	953960/21000	100%/100%
f_{18}	0	1000	20	0/78	1000/230	20/50	0/770	999340/39624	100%/23%
f_{19}	0	1000	1	0/0	1000/1000	1/1	0/0	3998/2000	100%/100%
f_{20}	0	1000	1	0/33	1000/287	1/241	0/713	1001000/155430	100%/28.7%

Category 3 problems (f_9 to f_{13}) contain 10 non-separable components, each with 50 interacting decision variables. The number of separable and non-separable decision variables are both 500 in each problem. On all of the 5 problems, XDG successfully identifies the 500 separable decision variables and 10 non-separable groups, each with 50 interacting variables. DG also achieves perfect decomposition on f_9, f_{10} and f_{12}. On f_{11}, DG only misplaces 1 non-separable decision variable. On f_{13}, DG forms 34 non-separable groups. The number of interacting decision variables correctly identified is 159. Note that f_{13} is a Rosenbrock function, which contains indirect interaction. It also shows that DG can not capture indirect interaction between decision variables. For this reason, DG may break one non-separable group into several groups. For example, in the problem: $f(\mathbf{x}) := (x_1 - x_2)^2 + (x_2 - x_3)^2 + (x_3 - x_4)^2$, there is only one non-separable group: $\{x_1, x_2, x_3, x_4\}$. However DG will form two non-separable groups: $\{x_1, x_2\}$ and $\{x_3, x_4\}$. This is why DG forms 34 non-separable groups on f_{13}.

Category 4 problems (f_{14} to f_{18}) contain 20 non-separable groups, each with 50 interacting decision variables. XDG achieves perfect decomposition on all of the 5 problems. DG achieves 100% decomposition accuracy on f_{14} f_{15} and f_{17}. On f_{16}, DG unexpectedly classifies 4 interacting variables as separable. Function f_{18} is another problem with indirect interaction. DG forms 50 non-separable groups. The number of interacting decision variables correctly identified is 230. Therefore, the decomposition accuracy of DG on f_{18} is 23%, which is very low. Note that the function evaluations used by XDG on all of the 5 problems are around one million, which is much greater than the function evaluations used by DG (around twenty thousand).

Category 5 consists of two fully non-separable problems with 1000 interacting decision variables (f_{19} and f_{20}). XDG successfully assigns all of the 1000 interacting variables into the same component. DG also achieves 100% decomposition accuracy on f_{19}. However on f_{20}, DG only correctly identifies 287 interacting decision variables. Note that f_{20} is also a Rosenbrock function. The number of function evaluations used by XDG and DG on f_{19} are both small. However on f_{20}, the number of function evaluations used by XDG is about 6 times greater than that used by DG.

In sum, DG achieves high decomposition accuracy on most of the benchmark problems. However, it performs poorly on problems with indirect interaction, such as f_7 f_{13} f_{18} and f_{20}. XDG achieves 100% decomposition accuracy on all of the CEC'2010 benchmark problems. It can successfully identify both direct interaction and indirect interaction. Therefore, the extension in XDG can address the limitation of the DG method. However, XDG is more computational expensive than DG as it requires computational resources to identify indirect interaction between decision variables. This represents a classic trade-off between accuracy and efficiency.

The parameter ϵ is the threshold to classify pairwise decision variables as direct interaction or independent. The larger the ϵ is, the more likely the interacting decision variables are classified as independent. However, if the ϵ is too small, the independent decision variables may be classified as interacting due to the computational errors in the system. Table 4.2 presents the grouping results of XDG with $\epsilon = 1$ and $\epsilon = 10^{-3}$, which are separated by "/".

According to Table 4.1 and Table 4.2, the grouping results of XDG with $\epsilon = 1$ are the same with $\epsilon = 10^{-1}$ on the CEC'2010 benchmark problems except f_{11} and f_{16}. On f_{11} and f_{16}, XDG identifies all of the decision variables as separable variables. The reason may be that f_{11} and f_{16} are very smooth problems. The fitness difference between two candidate solutions in the search space is less than 1. Therefore, $\epsilon = 1$ is too large to identify interacting variables.

When $\epsilon = 10^{-3}$, the decomposition accuracy of XDG on 18 out of 20 benchmark problems is 100%. On problem f_{13}, two non-separable groups are formed with only 50 interacting decision variables correctly identified. The reason for this is that the computational errors in the system is large compared with the value of ϵ: 10^{-3}. XDG unexpectedly assigns some separable decision variables into the non-separable group. Besides, the 10 groups of non-separable decision variables are assigned to the same group. Note that the second stage of XDG is to learn indirect interaction. An unexpected grouping of interacting decision variables in the first stage may result in two non-separable groups merged as one. It will cause a rapid deterioration of the decomposition accuracy. Similar results can be found with f_{18}, where XDG places all of the 20 groups of interacting decision variables into the same group. Therefore, the decomposition accuracy of XDG on f_{18} is 5%.

Table 4.2: Decomposition results of XDG with different values of threshold ϵ on CEC'2010 benchmark problems. The results of $\epsilon = 1$ and the results of $\epsilon = 10^{-3}$ are separated by "/".

				Extended Differential Grouping ($\epsilon = 1$)/ Extended Differential Grouping ($\epsilon = 10^{-3}$)					
Function	Sep Vars	Non-sep Vars	Non-Sep Groups	Captured Sep Vars	Captured Non-sep Vars	Formed Non-sep Groups	Misplaced Vars	Function Evaluation	Decomposition Accuracy
f_1	1000	0	0	1000/1000	0/0	0/0	0/0	1001000/1001000	100%/100%
f_2	1000	0	0	1000/1000	0/0	0/0	0/0	1001000/1001000	100%/100%
f_3	1000	0	0	1000/1000	0/0	0/0	0/0	1001000/1001000	100%/100%
f_4	950	50	1	0/0	50/50	1/1	0/0	80526/80526	100%/100%
f_5	950	50	1	950/950	50/50	1/1	0/0	998648/998648	100%/100%
f_6	950	50	1	950/950	50/50	1/1	0/0	998648/998648	100%/100%
f_7	950	50	1	950/0	50/50	1/1	0/0	998648/45290	100%/100%
f_8	950	50	1	0/0	50/50	2/2	0/0	121658/1216586	100%/100%
f_9	500	500	10	500/500	500/500	10/10	0/0	977480/977480	100%/100%
f_{10}	500	500	10	500/500	500/500	10/10	0/0	977486/ 977480	100%/100%
f_{11}	500	500	10	1000/500	0/500	0/10	500/0	1001000/977482	0%/100%
f_{12}	500	500	10	500/500	500/500	10/10	0/0	977480/977480	100%/100%
f_{13}	500	500	10	500/4	500/50	10/2	0/450	1000154/ 573776	100%/10%
f_{14}	0	1000	20	0/0	1000/1000	20/20	0/0	953960/953960	100%/100%
f_{15}	0	1000	20	0/0	1000/1000	20/20	0/0	953978/ 953960	100%/100%
f_{16}	0	1000	20	1000/0	0/1000	0/20	1000/0	1001000/953968	0%/100%
f_{17}	0	1000	20	0/0	1000/1000	20/20	0/0	953960/ 953960	100%/100%
f_{18}	0	1000	20	0/0	1000/50	20/1	0/950	999340/ 171370	100%/5%
f_{19}	0	1000	1	0/0	1000/1000	1/1	0/0	3996/3996	100%/100%
f_{20}	0	1000	1	0/0	1000/1000	1/1	0/0	1001000/ 705698	100%/100%

In sum, the selection of ϵ should not be too large or too small. A large value of ϵ will fail to identify some of the interacting decision variables, while a small value of ϵ will result in fault grouping of interacting decision variables. The value 10^{-1} is used for XDG in the rest of the chapter. See Section 5.2 for more discussion about the selection of the ϵ value.

4.4.2 DECC Comparison

The experimental results of the DECC-XDG, DECC-DG and DECC-D (with Delta grouping) when used to solve the CEC'2010 benchmark problems are presented in Table 4.3.

On fully separable problems, DECC-D (with Delta grouping) outperforms DECC-DG and DECC-XDG significantly. The reason may be that DG and XDG put all of the separable decision variables into the same component. When the number of separable variables is large (hundreds and thousands), it may not be a good choice to place all of the separable variables into the same component [159]. An alternative decomposition approach could be to treat each separable variable as one component, however, such an approach is not ideal. The intermediate decomposition between these two extreme cases has been shown to be more efficient [125]. Another reason may be that the number of function evaluations used by DECC-XDG or DECC-DG in the decomposition phase is too large, resulting in few function evaluations left for the evolutionary optimization phase. According to Table 4.1, the number of function evaluations used by DECC-XDG/DG in the decomposition phase is 1001000, which is approximately equal to 1/3 of the maximal number of function evaluations (3×10^6). Note that the performances of DECC-XDG and DECC-DG on the 3 separable problems are almost the same.

On f_8, f_{13}, f_{18} and f_{20}, DECC-XDG outperforms DECC-DG significantly. Note that all the of four problems have indirect interaction between decision variables. XDG achieves perfect decomposition on the four benchmark problems, while the decomposition accuracy of DG is low according to Table 4.1. Although the number of function evaluations used by DECC-XDG in the decomposition phase is greater than that of DECC-DG (see Table 4.1), DECC-XDG still significantly outperforms DECC-DG on the four benchmark problems. For f_{18}, DECC-XDG used 25 times the number of function evaluations used by

Table 4.3: The comparison between the performances of DECC-XDG, DECC-DG and DECC-D (with Delta grouping) on the CEC'2010 benchmark problems. The significant better performances are highlighted in bold (Wilcoxon rank-sum tests ($\alpha = 0.05$) with Holm p-value correction)

Func	Stats	DECC-XDG	DECC-DG	DECC-D
f_1	Mean	2.23e+04	1.12e+04	**4.07e-24**
	Std	8.01e+04	3.37e+04	1.75e-23
f_2	Mean	4.44e+03	4.42e+03	**2.82e+02**
	Std	1.64e+02	1.59e+02	2.40e+01
f_3	Mean	1.66e+01	1.67e+01	**1.52e-13**
	Std	4.12e-01	3.05e-01	8.48e-15
f_4	Mean	**7.84e+11**	4.63e+12	4.12e+12
	Std	1.67e+11	1.35e+12	1.46e+12
f_5	Mean	**1.68e+08**	1.98e+08	2.48e+08
	Std	1.74e+07	4.58e+07	4.79e+07
f_6	Mean	**1.63e+01**	**1.62e+01**	5.34e+07
	Std	3.28e-01	2.82e-01	8.79e+07
f_7	Mean	**1.39e+03**	1.63e+04	6.89e+07
	Std	2.61e+03	8.93e+03	4.96e+07
f_8	Mean	**4.78e+05**	2.51e+07	1.09e+08
	Std	1.32e+06	2.54e+07	4.87e+07
f_9	Mean	1.12e+08	**5.60e+07**	6.13e+07
	Std	1.13e+07	6.59e+06	6.28e+06
f_{10}	Mean	**5.31e+03**	**5.22e+03**	1.29e+04
	Std	1.55e+02	1.28e+02	2.27e+02
f_{11}	Mean	1.04e+01	9.94e+00	**1.55e-13**
	Std	1.15e+00	9.57e-01	8.19e-15
f_{12}	Mean	1.24e+04	**2.83e+03**	4.30e+06
	Std	2.32e+03	9.92e+02	1.79e+05
f_{13}	Mean	**1.21e+03**	5.35e+06	**1.19e+03**
	Std	2.25e+02	4.89e+06	5.02e+02
f_{14}	Mean	5.83e+08	3.43e+08	**1.93e+08**
	Std	4.11e+07	2.23e+07	1.06e+07
f_{15}	Mean	**5.91e+03**	**5.84e+03**	1.60e+04
	Std	7.56e+01	8.93e+01	4.24e+02
f_{16}	Mean	1.81e-08	**7.32e-13**	1.70e+01
	Std	1.57e-09	4.62e-14	8.48e+01
f_{17}	Mean	1.26e+05	**3.99e+04**	7.48e+06
	Std	7.47e+03	1.80e+03	4.03e+05
f_{18}	Mean	**1.41e+03**	1.44e+10	3.32e+03
	Std	1.88e+02	2.50e+09	7.09e+02
f_{19}	Mean	**1.59e+06**	1.72e+06	2.32e+07
	Std	4.96e+04	6.83e+06	5.56e+06
f_{20}	Mean	5.55e+05	6.69e+10	**1.18e+03**
	Std	1.75e+06	9.24e+09	8.25e+01

DECC-DG in the decomposition phase. However, the mean of the best solutions found by DECC-XDG is 2.60e+03, while the mean of the best solutions found by DECC-DG is 1.44e+10. The former is much better than the latter, which demonstrates the effectiveness of the extension in XDG. It also shows that an ideal decomposition can significantly improve the performance of a CC algorithm.

On problems without indirect interaction such as f_{15}, f_{16}, f_{17}, DECC-XDG achieves the same or slightly worse results than DECC-DG. Note that the decomposition accuracy achieved by DECC-XDG and DECC-DG are both the same on these problems. However the number of function evaluations used by DECC-XDG in the decomposition phase is much greater than that of DECC-DG. Therefore, the number of function evaluations left for DECC-XDG to optimize each component is much less than that of DECC-DG.

On problem f_{20}, DECC-XDG successfully places all of the 1000 interacting decision variables into the same component. However, the quality of the best solution found by DECC-XDG is worse than that of DECC-D (with Delta grouping). It may be an indication that assigning all of the interacting decision variables into the same component may not be a good decomposition method when the number of interacting variables is large.

4.5 Conclusion

In this chapter, I have investigated the effects of decision variable interaction and decomposition methods within the CC framework when used to solve the large-scale BCOPs. I have shown that the state-of-art decomposition method DG can not capture indirect interaction, and this in turn can be used to explain relatively poor performance on some of the benchmark problems. As a result, I have introduced the XDG method, to address this limitation. Results from comprehensive numerical simulation experiments clearly illustrated that XDG can achieve perfect decomposition on all of the benchmark problems investigated. When XDG was embedded within the DECC framework, it achieved significantly better performance than DECC-DG and DECC-D (with Delta grouping) on the benchmark problems with indirect interaction. It also achieved comparable performance with DECC-DG on the benchmark problems without indirect interaction.

This page intentionally left blank.

Chapter 5
A Recursive Decomposition Method for Large Scale Optimization

THE EXISTING decomposition methods can be classified into two very different approaches (see Section 2.2.3 for a review). In the *manual decomposition* method, the structure of components is manually designed (e.g., Random Grouping (RG) [197]). This method does not take the underlying structure of variable interactions into consideration. In the second method, *automatic decomposition*, the structure of components is determined by the identified decision variable interactions (e.g., eXtended Differential Grouping (XDG) as described in Chapter 4). However, this approach can be computationally expensive – decomposing an n-dimensional Black-box Continuous Optimization Problem (BCOP) typically consumes $\mathcal{O}(n^2)$ Function Evaluations (FEs). This high computational complexity results in an inappropriate allocation of computational resources to the decomposition stage rather than the optimization stage.

In this chapter, I further investigate the two research questions of this thesis in the large-scale BCOP domain. However I shift my focus from accuracy to efficiency: $\mathfrak{RQ}$5.1 *How can variable interactions be efficiently identified in a given large-scale BCOP?* and $\mathfrak{RQ}$5.2 *How can variable interactions be used to effectively solve a given large-scale BCOP?*

To investigate $\mathfrak{RQ}$5.1, I propose a *Recursive Differential Grouping* (RDG) method, which can decompose an n-dimensional problem using $\mathcal{O}\big(n\log(n)\big)$ FEs. The RDG method examines the interaction between a selected decision variable x_i and the remaining decision variables based on non-linearity detection. If any interaction is identified, the remaining decision variables will be divided into two equally sized groups, and the interaction between x_i and each group is checked. This process is carried out recursively until all of the individual decision variables interacting with x_i have been identified and have been placed into the same component as x_i.

I have evaluated the efficacy of the RDG method using large-scale benchmark BCOPs (problems from the special sessions on large-scale global optimization at CEC'2010 [172] and CEC'2013 [73]). Comprehensive numerical simulations showed that the RDG method can decompose the benchmark problems efficiently in terms of time complexity.

To answer $\mathfrak{RQ}5.2$, I verified that the component groupings dictated by RDG were in fact useful for optimization. When embedded into a Cooperative Co-evolution (CC) framework, the optimization results generated using the RDG method were statistically better in terms of solution quality when compared against seven other decomposition methods across the benchmark suite.

The remainder of this chapter is organized as follows. Section 5.1 describes the state-of-the-art algorithms for large-scale BCOPs excluding CC. Section 5.2 describes the proposed RDG method in detail. Section 5.3 describes experiments to evaluate the proposed RDG method. Section 5.4 presents and analyzes the experimental results. Section 4.5 concludes this chapter.

5.1 Related Work: Large-scale Optimizers Excluding Cooperative Co-evolution

In addition to CC (described in Section 2.2.3), there are other techniques that can be used to 'scale up' Evolutionary Algorithms (EAs) to address the additional challenges inherent in large-scale BCOPs. To paint a complete picture, representative techniques are introduced below. Note that some of the algorithms introduced in this section will be used in the experiments (Section 5.3).

5.1.1 Scaling up Estimation of Distribution Algorithms

The Model Complexity Control [29] method employs a divide-and-conquer strategy to scale up Estimation of Distribution Algorithms. The Model Complexity Control method identifies the weakly interacting decision variables and applies a univariate model on them. The strongly interacting decision variables are divided into several components, and a multivariate model for each component is estimated. The Random Projection [61] method projects the set of promising (best) candidate solutions to several low-dimensional subspaces. The candidate solutions are evolved in each subspace, and are combined to form a new population in the whole search space.

5.1.2 Scaling up Differential Evolution

Many techniques have been proposed to improve the performance of Differential Evolution (DE) to solve large-scale BCOPs. For example, the jDElscop [14] algorithm employs a self-adapted DE with three strategies: DE/rand/1/bin, DE/rand/1/exp and DE/best/1/bin. The population size is gradually reduced in the evolutionary process. The Generalized Opposition-based DE [186] algorithm employs a generalized opposition-based learning method to initialize the population and jump between generations. The mDE-bES [3] algorithm divides the population into subgroups, each with different mutation and update strategies.

5.1.3 Scaling up Partial Swarm Optimization

The Social Learning [136] and Pairwise Competition [22] methods are proposed to scale up the Particle Swarm Optimization. The Social Learning method updates the position of each particle by learning from the better particles in the current swarm. The Pairwise Competition method conducts pairwise competition between particles, and the particle that loses the competition will update its position by learning from the winner.

5.1.4 Hybridizing Algorithms

The Multiple Offspring Sampling (MOS) [69] framework can be used to hybridize both EAs and local search methods. In each generation, the composition algorithms are evaluated and the better performed algorithm will be used to generate more offspring. The MOS algorithm achieved the best performance in the 2011 special issue of the Soft Computing journal. The Memetic algorithm – MA-SW-Chains [99] – assigns to each individual a local search intensity that depends on its features, by chaining different local search applications. The MA-SW-Chains algorithm achieved the best performance in the CEC 2010 special session and competition on large-scale global optimization. In MA-SSW-Chains [100], a variation of Solis Wets method, Subgrouping Solis Wets algorithm is used. The MA-SW-Chain algorithm is also adapted to the GPU-based parallel architecture [68].

5.2 Recursive Differential Grouping

In this section, the proposed decomposition method – RDG – is described in detail. Then, the computational complexity of the RDG method is presented.

Notation. *Let X be the set of decision variables $\{x_1, \ldots, x_n\}$; U_X be the set of unit vectors in the decision space $\mathbb{R}^n$. Let X_1 be a subset of decision variables $X_1 \subset X$; and U_{X_1} be a subset of U_X such that any unit vector $\mathbf{u} = (u_1, \ldots, u_n) \in U_{X_1}$, I have*

$$u_i = 0, \qquad \text{if } x_i \notin X_1. \tag{5.1}$$

Directional Derivative. *Let $f\colon \mathbb{R}^n \to \bar{\mathbb{R}}$ be a differentiable function, and $\mathbf{u} = (u_1, \ldots, u_n)$ be a vector from U_X. The directional derivative of f in the direction $\mathbf{u}$, denoted $\mathcal{D}_{\mathbf{u}} f(\mathbf{x})$, is given by*

$$\mathcal{D}_{\mathbf{u}} f(\mathbf{x}) = \sum_{i=1}^{n} \frac{\partial f(\mathbf{x})}{\partial x_i} u_i. \tag{5.2}$$

Proposition 5.1. *Let $f\colon \mathbb{R}^n \to \bar{\mathbb{R}}$ be a differentiable function; $X_1 \subset X$ and $X_2 \subset X$ be two mutually exclusive subsets of decision variables: $X_1 \cap X_2 = \emptyset$. If there exist two unit vectors $\mathbf{u}_1 \in U_{X_1}$ and $\mathbf{u}_2 \in U_{X_2}$, and a candidate solution $\mathbf{x}^*$ in the decision space such that*

$$\mathcal{D}_{\mathbf{u}_1} \mathcal{D}_{\mathbf{u}_2} f(\mathbf{x}^*) \neq 0, \tag{5.3}$$

there is some interaction between decision variables in X_1 and X_2.

Proof. Without loss of generality, I assume that $X_1 = \{x_{1,1}, \ldots, x_{1,p}\}$, $X_2 = \{x_{2,1}, \ldots, x_{2,q}\}$, where p, q are the number of decision variables in X_1 and X_2 respectively; $\mathbf{u}_1 = (u_1^1, \ldots, u_n^1)$

and $\mathbf{u}_2 = (u_1^2, \ldots, u_n^2)$. According to Directional Derivative,

$$\mathcal{D}_{\mathbf{u}_1}\mathcal{D}_{\mathbf{u}_2} f(\mathbf{x}) = \sum_{i=1}^{n}\sum_{j=1}^{n} \frac{\partial^2 f(\mathbf{x})}{\partial x_i \partial x_j} u_i^1 u_j^2. \tag{5.4}$$

As $\mathbf{u}_1$ and $\mathbf{u}_2$ are two unit vectors from U_{X_1} and U_{X_2} respectively,

$$u_i^1 = 0, \qquad \text{if } x_i \notin X_1, \tag{5.5}$$

$$u_j^2 = 0, \qquad \text{if } x_j \notin X_2. \tag{5.6}$$

Therefore,

$$\mathcal{D}_{\mathbf{u}_1}\mathcal{D}_{\mathbf{u}_2} f(\mathbf{x}) = \sum_{i=1}^{p}\sum_{j=1}^{q} \frac{\partial^2 f(\mathbf{x})}{\partial x_{1,i} \partial x_{2,j}} u_{1,i}^1 u_{2,j}^2, \tag{5.7}$$

If (5.3) holds,

$$\sum_{i=1}^{p}\sum_{j=1}^{q} \frac{\partial^2 f(\mathbf{x}^*)}{\partial x_{1,i} \partial x_{2,j}} u_{1,i}^1 u_{2,j}^2 \neq 0. \tag{5.8}$$

Therefore, there exists at least one pair of (i, j), such that

$$\frac{\partial^2 f(\mathbf{x}^*)}{\partial x_{1,i} \partial x_{2,j}} \neq 0. \tag{5.9}$$

Based on Definition 3.1, at least one pair of decision variables $x_{1,i} \in X_1$ and $x_{2,j} \in X_2$ interact. □

Corollary 5.1. *Let f: $\mathbb{R}^n \to \bar{\mathbb{R}}$ be an objective function; $X_1 \subset X$ and $X_2 \subset X$ be two mutually exclusive subsets of decision variables: $X_1 \cap X_2 = \emptyset$. If there exist two unit vectors $\mathbf{u}_1 \in U_{X_1}$ and $\mathbf{u}_2 \in U_{X_2}$, two real numbers $l_1, l_2 > 0$, and a candidate solution $\mathbf{x}^*$ in the decision space, such that*

$$f(\mathbf{x}^* + l_1\mathbf{u}_1 + l_2\mathbf{u}_2) - f(\mathbf{x}^* + l_2\mathbf{u}_2) \neq f(\mathbf{x}^* + l_1\mathbf{u}_1) - f(\mathbf{x}^*), \tag{5.10}$$

there is some interaction between decision variables in X_1 and X_2.

Proof. With Proposition 5.1, I only need to prove the following statement:

Statement 1. *If there exist two unit vectors $\mathbf{u}_1 \in U_{X_1}$ and $\mathbf{u}_2 \in U_{X_2}$, two real numbers $l_1, l_2 > 0$, and a candidate solution $\mathbf{x}^*$ in the decision space, such that Eq. (5.10) holds, then Eq. (5.3) is* *true.*

It is equivalent to prove its contraposition:

Statement 2. *If for any two unit vectors $\mathbf{u}_1 \in U_{X_1}$ and $\mathbf{u}_2 \in U_{X_2}$, and for any candidate solution $\mathbf{x}^*$ in the decision space, the following condition holds:*

$$\mathcal{D}_{\mathbf{u}_1}\mathcal{D}_{\mathbf{u}_2}f(\mathbf{x}^*) = 0, \tag{5.11}$$

then

$$f(\mathbf{x}^* + l_1\mathbf{u}_1 + l_2\mathbf{u}_2) - f(\mathbf{x}^* + l_2\mathbf{u}_2) = f(\mathbf{x}^* + l_1\mathbf{u}_1) - f(\mathbf{x}^*), \tag{5.12}$$

for any $l_1, l_2 > 0$.

In order to prove Statement 2, I first introduce *line integral*.

Line Integral. *Let L be a curve with end points A and B in the decision space R^n and the arc length of L be l. Let C be any point on L and the coordinate of C ($\mathbf{x}$) can be uniquely determined by the length of the arc AC (s): $\mathbf{x} = \mathbf{x}(s)$, $s \in [0, l]$. The integral of a function $g : \mathbb{R}^n \to \bar{\mathbb{R}}$ along the curve L is given by*

$$\int_L g(\mathbf{x})\,\mathrm{d}s = \int_0^l g\big(\mathbf{x}(s)\big)\,\mathrm{d}s. \tag{5.13}$$

Let A_2 ($\mathbf{x}^*$) be any point in R^n, and B_2 be $\mathbf{x}^* + l_2\mathbf{u}_2$, where $\mathbf{u}_2$ is any vector in U_{X_2} and l_2 is any positive real number. Let C_2 be any point on the segment A_2B_2. Therefore, the length of the segment A_2B_2 is l_2, and the coordinate of C_2 ($\mathbf{x}$) can be uniquely determined by the length of the segment A_2C_2 (s_2): $\mathbf{x}(s_2) = \mathbf{x}^* + s_2\mathbf{u}_2, s_2 \in [0, l_2]$. If Eq. (5.11) holds for any candidate solution in the decision space, then

$$\mathcal{D}_{\mathbf{u}_1}\mathcal{D}_{\mathbf{u}_2}f(\mathbf{x}) = 0. \tag{5.14}$$

As $\mathcal{D}_{\mathbf{u}_1}\mathcal{D}_{\mathbf{u}_2}f(\mathbf{x}) = \mathcal{D}_{\mathbf{u}_2}\mathcal{D}_{\mathbf{u}_1}f(\mathbf{x})$, by integrating both sides of Eq. (5.14) along the segment A_2B_2, I can obtain

$$\int_0^{l_2} \mathcal{D}_{\mathbf{u}_1}\mathcal{D}_{\mathbf{u}_2}f(\mathbf{x})\,\mathrm{d}s_2 = \int_0^{l_2} \mathcal{D}_{\mathbf{u}_2}\mathcal{D}_{\mathbf{u}_1}f(\mathbf{x})\,\mathrm{d}s_2 = 0. \tag{5.15}$$

As

$$\int_0^{l_2} \mathcal{D}_{\mathbf{u}_2}\Big(\mathcal{D}_{\mathbf{u}_1} f\big(\mathbf{x}(s_2)\big)\Big)\,\mathrm{d}s_2 = \mathcal{D}_{\mathbf{u}_1} f\big(\mathbf{x}(s_2)\big)\Big|_{s_2=0}^{s_2=l_2}, \tag{5.16}$$

thus

$$\mathcal{D}_{\mathbf{u}_1} f\big(\mathbf{x}(s_2)\big)\Big|_{s_2=0}^{s_2=l_2} = 0, \tag{5.17}$$

and

$$\mathcal{D}_{\mathbf{u}_1} f(\mathbf{x}^* + l_2\mathbf{u}_2) - \mathcal{D}_{\mathbf{u}_1} f(\mathbf{x}^*) = 0. \tag{5.18}$$

As A_2 ($\mathbf{x}^*$) is any point in R^n, therefore

$$\mathcal{D}_{\mathbf{u}_1} f(\mathbf{x} + l_2\mathbf{u}_2) = \mathcal{D}_{\mathbf{u}_1} f(\mathbf{x}). \tag{5.19}$$

Let A_1 ($\mathbf{x}^*$) be any point in R^n, and B_1 be $\mathbf{x}^* + l_1\mathbf{u}_1$, where $\mathbf{u}_1$ is any vector in U_{X_1} and l_1 is any positive real number. Let C_1 be any point on the segment A_1B_1. Therefore, the length of the segment A_1B_1 is l_1, and the coordinate of C_1 ($\mathbf{x}$) can be uniquely determined by the length of the segment A_1C_1 (s_1): $\mathbf{x}(s_1) = \mathbf{x}^* + s_1\mathbf{u}_1, s_1 \in [0, l_1]$. Similarly, by integrating both sides of Eq. (5.19) along the segment A_1B_1, I can obtain

$$\int_0^{l_1} \mathcal{D}_{\mathbf{u}_1} f(\mathbf{x}(s_1) + l_2\mathbf{u}_2)\,\mathrm{d}s_1 = \int_0^{l_1} \mathcal{D}_{\mathbf{u}_1} f(\mathbf{x}(s_1))\,\mathrm{d}s_1. \tag{5.20}$$

Therefore,

$$f(\mathbf{x}^* + l_1\mathbf{u}_1 + l_2\mathbf{u}_2) - f(\mathbf{x}^* + l_2\mathbf{u}_2) = f(\mathbf{x}^* + l_1\mathbf{u}_1) - f(\mathbf{x}^*). \tag{5.21}$$

Thus, Statement 2 is proved, and Statement 1 and Corollary 5.1 are true. □

With Corollary 5.1, the interaction between two subsets of decision variables (X_1 and X_2) can be identified using the following procedures:

1. Set all the decision variables to the lower bounds (**lb**) of the search space ($\mathbf{x}_{l,l}$);

2. Perturb the decision variables X_1 of $\mathbf{x}_{l,l}$ from the lower bounds to the upper bounds (**ub**), denoted by $\mathbf{x}_{u,l}$;

3. Calculate the fitness difference (δ_1) between $\mathbf{x}_{l,l}$ and $\mathbf{x}_{u,l}$;

4. Perturb the decision variables X_2 of $\mathbf{x}_{l,l}$ and $\mathbf{x}_{u,l}$ from the lower bounds to the middle between the lower bounds and upper bounds, denoted by $\mathbf{x}_{l,m}$ and $\mathbf{x}_{u,m}$ respectively;

5. Calculate the fitness difference (δ_2) between $\mathbf{x}_{l,m}$ and $\mathbf{x}_{u,m}$;

6. If the non-linearity term $\lambda = |\delta_1 - \delta_2|$ is greater than a threshold ϵ, there is some interaction between X_1 and X_2.

The two subscripts of $\mathbf{x}$ denote the values of X_1 and X_2 respectively: 'l' means lower bounds, 'u' means upper bounds, and 'm' means the middle of the lower and upper bounds.

In theory, the threshold ϵ can be set to zero, as any positive value of the non-linearity term ($\lambda = |\delta_1 - \delta_2|$) implies an interaction between the subset of decision variables in examination. However in practice, the value of λ for separable decision variables may be non-zero, due to the computational round-off errors incurred by floating-point operations (see [126] for details). Therefore, a positive threshold value ($\epsilon > 0$) is required to differentiate the genuine non-zero λ values. In this chapter, the threshold value is estimated based on the magnitude of fitness values [93]:

$$\epsilon = \alpha \cdot \min\left\{|f(\mathbf{x}_1)|, \cdots, |f(\mathbf{x}_k)|\right\}, \tag{5.22}$$

where $\mathbf{x}_1, \cdots, \mathbf{x}_k$ are k randomly generated solutions, and α is the control coefficient [93].[1]

Based on Corollary 5.1, I propose the RDG (Algorithm 5) method to efficiently decompose an optimization problem. The decomposition by the RDG method caters for the underlying structure of variable interactions. Taking the following objective function as an example:

$$f(\mathbf{x}) := x_1^2 + (x_2 - x_3)^2 + (x_3 - x_4)^2 + (x_5 - x_6)^2,\ \mathbf{x} \in [-1, 1]^6, \tag{5.23}$$

the decision variables (x_2, x_3, x_4) interact, as well as (x_5, x_6). Therefore, the decomposition by the RDG method is $\{(x_1), (x_2, x_3, x_4), (x_5, x_6)\}$.

[1]In a follow-up paper [168], I estimated a threshold value for RDG based on computational round-off errors.

Algorithm 5 Recursive Differential Grouping

```
Require: $f$, $\mathbf{ub}$, $\mathbf{lb}$, $\epsilon$
1: Initialize seps and nonseps as empty groups
2: Set all decision variables to the lower bounds: $\mathbf{x}_{l,l} = \mathbf{lb}$
3: Calculate the fitness: $y_{l,l} = f(\mathbf{x}_{l,l})$
4: Assign the first variable $x_1$ to the variable subset $X_1$
5: Assign the rest of variables to the variable subset $X_2$
6: while $X_2$ is not empty do
7:    $[X_1^*]$ = INTERACT($X_1$, $X_2$, $\mathbf{x}_{l,l}$, $y_{l,l}$, $\epsilon$)
8:    if $X_1^*$ is the same with $X_1$ then
9:       if $X_1$ contains one decision variable then
10:          Add $X_1$ to seps
11:      else
12:          Add $X_1$ to nonseps
13:      end if
14:      Empty $X_1$ and $X_1^*$
15:      Assign the first variable of $X_2$ to $X_1$
16:      Delete the first variable in $X_2$
17:   else
18:      $X_1 = X_1^*$
19:      Delete the variables of $X_1$ from $X_2$
20:   end if
21: end while
22: return seps and nonseps
```

Algorithm 6 INTERACT$\{X_1, X_2, \mathbf{x}_{l,l}, y_{l,l}, \epsilon\}$

```
1: $\mathbf{x}_{u,l}$=$\mathbf{x}_{l,l}$; $\mathbf{x}_{u,l}(X_1) = \mathbf{ub}(X_1)$ //Set $X_1$ to the ub
2: Calculate the fitness change: $\delta_1 = y_{l,l} - f(\mathbf{x}_{u,l})$
3: $\mathbf{x}_{l,m} = \mathbf{x}_{l,l}$; $\mathbf{x}_{l,m}(X_2) = \big(\mathbf{lb}(X_2) + \mathbf{ub}(X_2)\big)/2$
4: $\mathbf{x}_{u,m} = \mathbf{x}_{u,l}$; $\mathbf{x}_{u,m}(X_2) = \big(\mathbf{lb}(X_2) + \mathbf{ub}(X_2)\big)/2$
5: Calculate the fitness change: $\delta_2 = f(\mathbf{x}_{l,m}) - f(\mathbf{x}_{u,m})$
6: if $|\delta_1 - \delta_2| > \epsilon$ then
7:    if $X_2$ contains one variable then
8:       $X_1 = X_1 \cup X_2$
9:    else
10:      Divide $X_2$ into equally-sized groups $G_1$, $G_2$
11:      $[X_1^1]$=INTERACT($X_1$, $G_1$, $\mathbf{x}_{l,l}$, $y_{l,l}$, $\epsilon$)
12:      $[X_1^2]$=INTERACT($X_1$, $G_2$, $\mathbf{x}_{l,l}$, $y_{l,l}$, $\epsilon$)
13:      $[X_1]$= $X_1^1 \cup X_1^2$
14:   end if
15: end if
16: return $X_1$
```

The inputs to the RDG method are the fitness function (f), the upper bounds ($\mathbf{ub}$), the lower bounds ($\mathbf{lb}$), and the threshold (ϵ) which is estimated using Eq. (5.22). The outputs are the separable variable group (*seps*) and the non-separable variable groups (*nonseps*). The *seps* contains *one* group of all separable decision variables. The *nonseps* contains *several* groups of non-separable decision variables. Each group of decision variables will form a component.

The RDG method begins by identifying the interaction between the first decision variable x_1 and the remaining decision variables. If no interaction is detected, x_1 will be placed in the separable decision variable group, and the algorithm will move on to the next decision variable x_2. If any interaction is detected, the remaining decision variables will be divided into two (nearly) equally-sized groups G_1 and G_2. Then the interaction between x_1 and G_1, x_1 and G_2 will be identified respectively. This process is recursively conducted until all the individual decision variables that directly interact with x_1 are identified and placed in the decision variable subset X_1 with x_1.

Then, the RDG method examines the interaction between X_1 and the remaining decision variables (excluding the decision variables in X_1) to identify the individual decision variables that indirectly interact with x_1 (linked by other decision variables). If any interaction is identified, the interacting decision variables will be placed into X_1. This process is repeated until no interaction can be further detected between X_1 and the remaining decision variables (exclusive X_1). The decision variables in X_1 will be placed in a non-separable group.

The RDG method moves on to the next decision variable that has not been grouped (x_i). The interaction between x_i and the remaining decision variables will be examined, and the interacting (both directly and indirectly) decision variables will be placed into one group with x_i. This process is repeated until all of the decision variables are grouped. It returns the separable (*seps*) and the non-separable (*nonseps*) decision variable groups as the outputs.

The computational complexity of the RDG method when used to decompose an n-dimensional problem is $\mathcal{O}\big(n\log(n)\big)$, which is analyzed as follows:

1. When decomposing an n-dimensional fully separable problem, the computational

complexity of the RDG method is $\Theta(n)$ in terms of the number of FEs. For each decision variable, 3 FEs are used to determine its separability. Therefore, totally about $3n$ FEs are needed.

2. When decomposing an n-dimensional fully non-separable problem with one component, the computational complexity of the RDG method is $\Theta(n)$. When grouping the n interacting decision variables, the function 'INTERACT' is executed about $\sum_{i=0}^{k}(n/2^i)$ times, where $k = \log_2(n)$.

$$\sum_{i=0}^{k} \frac{n}{2^i} = n\left(2 - \left(\frac{1}{2}\right)^k\right) < 2n \tag{5.24}$$

It consumes 3 FEs each time the function 'INTERACT' is executed. Therefore totally about $6n$ FEs are used.

3. When decomposing an n-dimensional fully non-separable problem with n/m components, the computational complexity of the RDG method is $\Theta\big(n\log(n)\big)$, where n is the dimensionality and m is the number of decision variables in each component. When grouping m interacting decision variables into one component, the function 'INTERACT' is executed less than $2m \times \log_2(n)$ times, each time consuming 3 FEs. The number of components is n/m. Therefore, totally less than $3 \times 2m \times \log_2(n) \times n/m = 6n\log_2(n)$ FEs are used.

4. When decomposing an n-dimensional partially separable problem with an m dimensional non-separable component, the computational complexity of the RDG method is $\mathcal{O}\big(\max\{n, m\log(n)\}\big)$. The RDG algorithm consumes about $3(n-m)$ FEs to identify the $n-m$ separable decision variables, and consumes less than $6m \times \log_2(n)$ FEs to identify the m interacting decision variables. Therefore, totally less than $3(n-m) + 6m \times \log_2(n)$ FEs are used.

5. When decomposing an n-dimensional overlapping problem (e.g., Rosenbrock's function [73]), the computational complexity of the RDG method is $\Theta\big(n\log(n)\big)$. Starting from x_1, it consumes about $3 \times 2 \times 2\log_2(n) = 12\log_2(n)$ FEs to identify the two decision variables (x_p and x_q) that interact with x_1. Then it also consumes about

$12 \log_2(n)$ FEs to identify the two decision variables that interact with (x_p, x_1, x_q). Therefore, totally about $n/2 \times 12 \log_2(n) = 6n \log_2(n)$ FEs are used.

5.3 Experimental Methodology

In this section, comprehensive numerical experiments are designed to evaluate the proposed RDG method. Two questions guide the experimental design:

Q5.1 Can the proposed RDG method decompose the CEC'2010 and CEC'2013 benchmark problems more efficiently when compared against other well-known decomposition methods?

Q5.2 Can the proposed RDG method outperform other well-known decomposition methods when embedded into a CC framework to solve the CEC'2010 and CEC'2013 benchmark problems?

The question Q5.1 corresponds to the first research question $\mathfrak{RQ}$5.1, while Q5.2 corresponds to the second research question $\mathfrak{RQ}$5.2.

To answer Q5.1, the proposed RDG method was used to decompose the CEC'2010 [172] and CEC'2013 [73] benchmark problems. Two metrics were employed to evaluate the performance of a decomposition method: (1) FEs: the number of FEs used to decompose the problem; and (2) Decomposition Accuracy (DA): the percentage of interacting decision variables that are correctly grouped, defined in Section 4.3. The performance of the RDG method was then compared to the performance of GDG [93], XDG, DG [120], as well as two recently published methods – DG2 [126], and FII [57]. The control coefficient α and sample size k were set to 10^{-12} and 10 respectively for the RDG and GDG methods. The parameter settings for other decomposition methods were consistent with the original papers (listed in Table 5.1). The threshold values (ϵ) estimated by the RDG (or GDG) method for each problem were recorded. Note that the RDG and GDG methods used the same approach (Eq. (5.22)) to estimate the threshold values.

To answer Q5.2, the proposed RDG method was embedded into the DECC [197] / CMAESCC [93] framework to solve the CEC'2010 and CEC'2013 benchmark problems.

Table 5.1: The parameter settings for all the decomposition methods used in the experiments. DG2 is a non-parametric method.

Decomposition Methods	Parameter Settings
RDG	control coefficient $\alpha = 10^{-12}$ and $k = 10$
GDG	control coefficient $\alpha = 10^{-12}$ and $k = 10$
XDG	threshold $\epsilon = 10^{-1}$
DG	threshold $\epsilon = 10^{-3}$
DG2	parameter free
FII	threshold $\epsilon = 10^{-2}$
D (delta grouping)	component size $sub\_dim = 100$
RG	component size $sub\_dim = 100$

The DECC is the most widely used CC framework, which employs a variant of DE – SaNSDE [199] – to solve each component cooperatively. The CMAESCC framework uses the well-known CMA-ES [45] algorithm to solve each component. It performs well when used to solve the CEC'2010 and CEC'2013 benchmark problems. The parameter settings for the DECC and CMAESCC frameworks were consistent with the original papers. The maximum number of FEs was set to 3×10^6, divided between the decomposition stage and optimization stage. For each benchmark problem, the mean and standard deviation of the best solutions found by the DECC/CMAESCC-RDG algorithm based on 25 independent runs were recorded.

The performance of the RDG method was compared against the performance of the XDG, GDG, DG, DG2, FII, as well as two manual decomposition methods – D (delta grouping [123]) and RG [197] methods, when embedded in each CC framework. The performance of DECC/CMAESCC-RDG was also compared to the performance of two state-of-the-art hybrid algorithms – MOS [69] and MA-SW-Chains [99] with default parameter settings. The statistical test used to determine the better performing algorithm is the same with the one used in Section 4.3.

5.4 Experimental Results

Comprehensive experimental results are presented and discussed in this section. Section 5.4.1 presents the decomposition comparison between the RDG method and five other methods, thus addressing Q5.1. Section 5.4.2 presents the optimization comparison between the RDG method and seven other methods when embedded into the DECC or CMAESCC framework to solve the benchmark problems, thus addressing Q5.2.

5.4.1 Decomposition Comparison

Table 5.2 and Table 5.3 list the decomposition results of the RDG, GDG, XDG and DG methods on the CEC'2010 and CEC'2013 benchmark problems respectively. "DA" represents the decomposition accuracy; "FEs" represents the number of FEs used in the decomposition stage; "ϵ" represents the threshold used to identify interactions between decision variables. Note that the threshold values used by RDG are the same as those used by GDG. The entries with the best decomposition accuracy achieved using the smallest number of FEs are highlighted in bold.

The RDG and GDG methods obtain nearly the same decomposition accuracy across all the benchmark problems. The reason for this is that RDG uses the same approach (Eq. (5.22)) as GDG to estimate the threshold values. However, RDG is much more efficient than GDG in terms of FEs used. Note that the number of FEs used by GDG to decompose an n-dimensional problem is fixed: $(n^2 + 3n + 2)/2$.

The first three problems (f_1-f_3) from each benchmark suite are fully separable. Therefore, decomposition accuracy is not applicable to these problems. On f_1 and f_2 (CEC'2010 and CEC'2013), the RDG method successfully identifies all the decision variables as separable, using a small number of FEs. However on f_3, RDG and GDG identify all the separable variables as non-separable, and place them into one component. The reason for this is that the threshold value is under-estimated by RDG and GDG on f_3.

The CEC'2013 f_{13} and f_{14} are benchmark problems with overlapping (conforming or conflicting) components. It is not clear yet what is the best approach to decompose these problems [73, 121]. The RDG, GDG and XDG methods place all the overlapped components into one group. On the other benchmark problems where the components are independent with each other, the 'ideal' decomposition can possibly be achieved (see [121, 120, 73, 172] for more information). Note that the 100% decomposition accuracy in Table 5.2 and Table 5.3 corresponds to the ideal decomposition.

Table 5.2: The experimental results of the proposed RDG method when used to decompose the CEC'2010 benchmark problems. Note that the threshold values used by RDG are the same as those used by GDG. The performance of the RDG method is compared with the performances of the GDG, XDG and DG methods. The entries with the best DA achieved using the lowest FEs are highlighted in bold.

Bench-	Func	RDG			GDG		XDG ($\epsilon = 10^{-1}$)		DG ($\epsilon = 10^{-3}$)	
marks	Num	DA	FEs	ϵ	DA	FEs	DA	FEs	DA	FEs
CEC'2010	f_1	–	3.00e+03	4.11e-01	–	5.01e+05	–	1.00e+06	–	1.00e+06
	f_2	–	3.00e+03	2.49e-08	–	5.01e+05	–	1.00e+06	–	1.00e+06
	f_3	–	6.00e+03	2.15e-11	–	5.01e+05	–	1.00e+06	–	1.00e+06
	f_4	**100%**	**4.20e+03**	1.03e+04	100%	5.01e+05	100%	8.05e+04	100%	1.45e+04
	f_5	**100%**	**4.15e+03**	1.14e-03	100%	5.01e+05	100%	9.98e+05	100%	9.05e+05
	f_6	**100%**	**5.00e+04**	2.13e-05	100%	5.01e+05	100%	9.98e+05	100%	9.06e+05
	f_7	**100%**	**4.23e+03**	5.17e+00	100%	5.01e+05	100%	9.98e+05	68.0%	6.77e+04
	f_8	**100%**	**5.60e+03**	2.62e+05	100%	5.01e+05	100%	1.21e+05	90.0%	2.32e+04
	f_9	**100%**	**1.40e+04**	4.88e-01	100%	5.01e+05	100%	9.77e+05	100%	2.70e+05
	f_{10}	**100%**	**1.40e+04**	2.52e-08	100%	5.01e+05	100%	9.77e+05	100%	2.72e+05
	f_{11}	**100%**	**1.36e+04**	2.36e-10	100%	5.01e+05	100%	9.78e+05	99.8%	2.70e+05
	f_{12}	**100%**	**1.43e+04**	4.26e-05	100%	5.01e+05	100%	9.77e+05	100%	2.71e+05
	f_{13}	**100%**	**2.92e+04**	3.71e+00	100%	5.01e+05	100%	1.00e+06	31.8%	5.03e+04
	f_{14}	**100%**	**2.05e+04**	4.15e-01	100%	5.01e+05	100%	9.53e+05	100%	2.10e+04
	f_{15}	**100%**	**2.05e+04**	2.53e-08	100%	5.01e+05	100%	9.53e+05	100%	2.10e+04
	f_{16}	**100%**	**2.09e+04**	4.30e-10	100%	5.01e+05	100%	9.56e+05	99.6%	2.11e+04
	f_{17}	**100%**	**2.07e+04**	1.10e-04	100%	5.01e+05	100%	9.53e+05	100%	2.10e+04
	f_{18}	**100%**	**4.98e+04**	8.19e+00	100%	5.01e+05	100%	9.99e+05	23.0%	3.96e+04
	f_{19}	100%	6.00e+03	6.14e-04	100%	5.01e+05	100%	3.99e+03	**100%**	**2.00e+03**
	f_{20}	**100%**	**5.08e+04**	8.53e+00	100%	5.01e+05	100%	1.00e+06	28.7%	1.55e+05

Table 5.3: The experimental results of the proposed RDG method when used to decompose the CEC'2013 benchmark problems. Note that the threshold values used by RDG are the same as those used by GDG. The performance of the RDG method is compared with the performances of the GDG, XDG and DG methods. The entries with the best DA achieved using the lowest FEs are highlighted in bold.

Bench-	Func	RDG			GDG		XDG ($\epsilon = 10^{-1}$)		DG ($\epsilon = 10^{-3}$)	
marks	Num	DA	FEs	ϵ	DA	FEs	DA	FEs	DA	FEs
CEC'2013	f_1	–	3.00e+03	4.20e-01	–	5.01e+05	–	1.00e+06	–	1.00e+06
	f_2	–	3.00e+03	1.31e-07	–	5.01e+05	–	1.00e+06	–	1.00e+06
	f_3	–	6.00e+03	2.16e-11	–	5.01e+05	–	1.00e+06	–	1.00e+06
	f_4	**100%**	**9.84e+03**	7.22e+01	100%	5.01e+05	33.3%	3.97e+05	95.3%	1.56e+04
	f_5	**100%**	**1.01e+04**	8.03e-05	100%	5.01e+05	0.00%	1.00e+06	0.00%	1.00e+06
	f_6	**100%**	**1.32e+04**	1.07e-06	100%	5.01e+05	50.0%	9.90e+05	82.6%	5.79e+05
	f_7	**100%**	**9.82e+03**	5.82e+05	100%	5.01e+05	33.3%	2.66e+04	39.6%	1.14e+04
	f_8	80.0%	1.95e+04	1.20e+06	80.0%	5.01e+05	10.0%	6.83e+04	**85.6%**	**2.26e+04**
	f_9	100%	1.92e+04	6.07e-03	100%	5.01e+05	99.9%	9.35e+05	**100%**	**1.76e+04**
	f_{10}	82.7	1.91e+04	9.80e-05	**90.0%**	**5.01e+05**	79.6%	9.52e+05	79.8%	4.86e+04
	f_{11}	10.0%	1.06e+04	1.52e+06	10.0%	5.01e+05	10.0%	2.20e+04	**37.7%**	**9.10e+03**
	f_{12}	**100%**	**5.08e+04**	8.57e+00	100%	5.01e+05	100%	1.00e+06	39.0%	1.49e+05
	f_{13}	–	8.39e+03	1.83e+06	–	4.10e+05	–	1.29e+04	–	5.86e+03
	f_{14}	–	1.61e+04	5.45e+06	–	4.10e+05	–	3.57e+04	–	1.39e+04
	f_{15}	100%	6.16e+03	2.70e+06	100%	5.01e+05	100%	3.99e+03	**100%**	**2.00e+03**

On the CEC'2010 partially separable problems (f_4-f_{18}), the RDG method consistently achieves the best results when compared against GDG, XDG and DG. The RDG, GDG and XDG methods achieve the ideal decomposition on all of these benchmark problems. However, the number of FEs used by the GDG and XDG methods is usually several magnitude larger than that used by the RDG method. The DG method performs well on the benchmark problems without conditional variable interactions. On f_{19}, the DG method uses the smallest number of FEs to decompose the problem. However on problems with conditional variable interactions (overlapping problems e.g., CEC'2010 f_{13}, f_{18} and f_{20}), the decomposition accuracy of the DG method is low. The reason for this is that DG is unable to completely identify variable interactions in overlapping problems [93].

On the CEC'2013 partially separable problems (f_4-f_{11}), the RDG method achieves the best results on 4 out of 8 benchmark problems. I observe that it is generally more difficult to identify the variable interactions in the CEC'2013 than the CEC'2010 benchmark problems. None of the four methods can perfectly decompose the CEC'2013 f_8, f_{10} and f_{11} problems. On CEC'2013 f_{11}, the threshold value is underestimated by the RDG and GDG methods $\big($Eq. (5.22)$\big)$, resulting in placing all the decision variables into a single non-separable group. While on CEC'2013 f_8 and f_{10}, the threshold value is overestimated, resulting in some omissions of variable interactions being identified.

The threshold values estimated by the GDG and RDG methods vary significantly across the CEC'2010 and CEC'2013 benchmark problems. Therefore, it is very difficult to find a single threshold value to accurately identify variable interactions across all the problems. Although the threshold value $\epsilon = 10^{-1}$ works well for the XDG method on the CEC'2010 benchmark problems, it fails on some of the CEC'2013 benchmark problems e.g., f_5-f_8. On f_5, the XDG algorithm (with $\epsilon = 10^{-1}$) identifies all the interacting decision variables as separable. The reason for this is that the threshold value 10^{-1} is too large compared with the computational error, which is equal to 8.03×10^{-5} estimated by the RDG and GDG methods. When using the estimated computational error as the threshold value, the XDG method achieves equal decomposition accuracy with the RDG method.

To further show the efficacy of the proposed RDG method, the performance of RDG is compared against two recently published methods – DG2 and FII. The average number

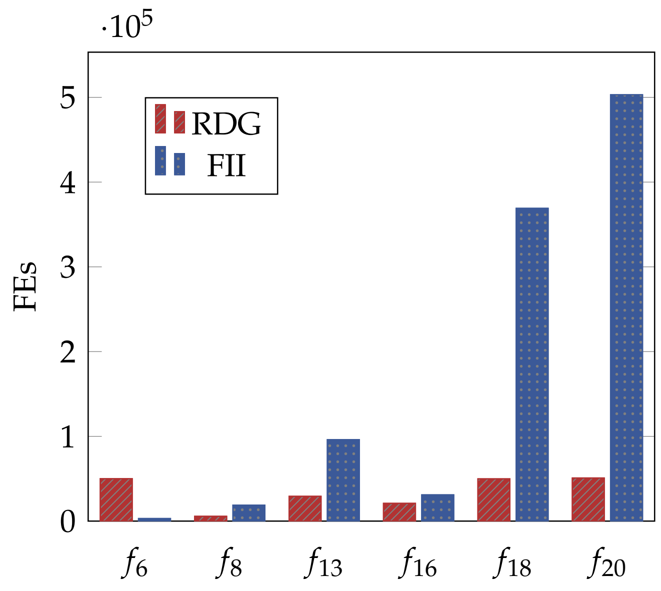


Figure 5.1: The benchmark problems from the CEC'2010 test suite on which the RDG and FII methods generate significant different results (the difference between the number of FEs used is greater than 10^4).

of FEs used by RDG to decompose the CEC'2010 and CEC'2013 benchmark problems is 1.47×10^4, which is less than that used by the DG2 and FII methods: 4.95×10^5 and 4.94×10^4 respectively. The detailed decomposition results of the DG2 and FII methods are presented in the Appendix B.

The DG2 method uses a fixed number of FEs to decompose an n-dimensional problem: $(n^2 + n + 2)/2$, which has been shown to be the lower bound of identifying the complete variable interaction matrix. With the complete variable interaction matrix being identified, it is possible to generate an effective decomposition for the problems with overlapping components, e.g., CEC'2013 f_{13} and f_{14} [126]. However, the existing automatic decomposition methods place all the linked (both directly interacting and indirectly interacting) decision variables into one component.

DG2 is a non-parametric method, which automatically estimates the threshold values in the decomposition process. The decomposition accuracy of DG2 on the CEC'2010 and CEC'2013 benchmark problems is high. It achieves 100% decomposition accuracy for the CEC'2013 f_{10} and f_{11} problems. The variable interactions in these two problems are difficult for the other decomposition methods to identify. However on CEC'2013 f_7 and

Table 5.4: The extended CEC'2010 f_{12}, f_{15} and f_{18} problems. For each problem, the number of decision variables in each non-separable component is fixed to 50, which is consistent with the original benchmark set [172].

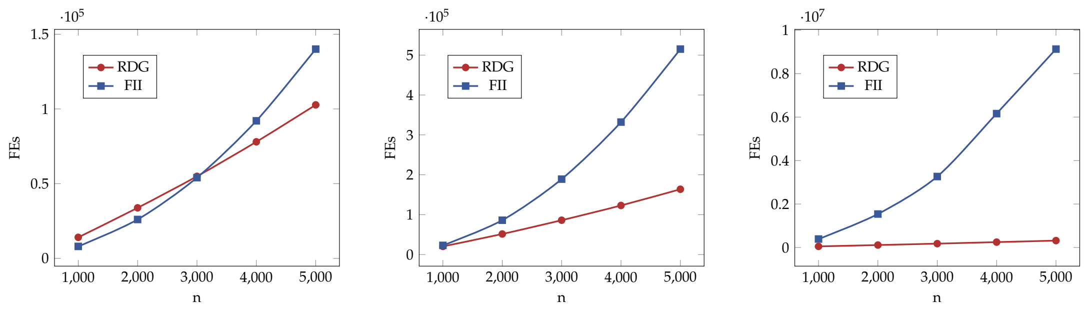
Func	Dim	Sep Variables	Non-Sep Groups
	1000	500	10
	2000	1000	20
f_{12}	3000	1500	30
	4000	2000	40
	5000	2500	50
	1000	0	20
	2000	0	40
f_{15}, f_{18}	3000	0	60
	4000	0	80
	5000	0	100

f_8, the decomposition accuracy of DG2 is less than that of RDG.

The FII method performs well when used to decompose the benchmark problems with a large portion of separable decision variables. For example on CEC'2010 f_4, where there are 950 separable and 50 non-separable decision variables, the number of FEs used by FII (3.69×10^3) is slightly less than that used by RDG (4.20×10^3). However, on some benchmark problems especially those with indirect variable interactions e.g., CEC'2010 f_{18} and f_{20}, RDG is much more efficient than FII, as shown in Figure 5.1.

In fact, the number of FEs used by the FII method to decompose the CEC'2010 f_{18} and f_{20} problems is in $\Theta(n^2)$. It has been shown that FII uses $3n + kn_n + k$ FEs when decomposing an n-dimensional problem with equally sized non-separable components, where n_n is the number of non-separable decision variables, and k is the number of non-separable components [57]. As CEC'2010 f_{18} and f_{20} are Rosenbrock's functions, n_n is equal to n and each decision variable interacts with at most two other decision variables. Therefore the total number of FEs used by FII is around $3n + n^2/3 + n/3 \in \Theta(n^2)$.

On CEC'2010 f_6, the number of FEs used by RDG (5.00×10^4) is greater than that used by FII (3.05×10^3). The reason for this is that the threshold estimated by the RDG method ($\epsilon = 2.13 \times 10^{-5}$) is too small, resulting in identifying some separable decision variables as non-separable. If RDG employs the same threshold with FII ($\epsilon = 0.01$), the FEs used by RDG will decrease to 5.06×10^3. However if FII employs the same threshold with



(a) CEC'2010 f_{12} (b) CEC'2010 f_{15} (c) CEC'2010 f_{18}

Figure 5.2: The number of FEs used by RDG and FII methods when used to decompose the extended benchmark problems (CEC'2010 f_{12}, f_{15} and f_{18}) with dimensionality equal to 1000, 2000, 3000, 4000, and 5000.

RDG, the FEs used by FII will increase to 3.12×10^5.

To test the scalability of the RDG and FII methods, I extend some of the CEC'2010 benchmark problems from 1000 dimensions to 5000 dimensions (see Table 5.4 for details). When tested on the extended benchmark problems, I observe that the FEs used by the FII method increases more quickly than that used by the RDG method as the dimensionality increases, as shown in Figure 5.2.

5.4.2 Optimization Comparison

The performances of the RDG, GDG, XDG, DG, DG2, FII, D (delta grouping), and RG methods when embedded in the DECC/CMAESCC framework to solve the CEC'2010 and CEC'2013 benchmark problems are presented in Figure 5.3 and Figure 5.4 respectively. On each benchmark problem, the eight methods are ranked from 1 to 8 (as labelled on the concentric circles in the radar charts) based on the results from 25 independent runs. The average ranking of each decomposition method across all the benchmark problems is presented in Table 5.5. The detailed optimization results from DECC-RDG and CMAESCC-RDG are presented in Table 5.6 and Table 5.7. The results from the other five methods are placed in the Appendix B.

When embedded into the DECC/CMAESCC framework, the RDG method achieves the best solution quality when used to solve 16/15 out of 23 partially separable benchmark problems (CEC'2010 f_4-f_{18} and CEC'2013 f_4-f_{11}). It obtains the smallest average

Table 5.5: The average ranking of each decomposition method when embedded into the DECC or CMAESCC framework to solve the CEC'2010 and CEC'2013 benchmark problems. The RDG method consistently achieves the smallest average ranking.

CCs	RDG	GDG	XDG	DG	DG2	FII	D	RG
DECC	2.0	4.9	4.0	4.4	3.9	2.5	4.6	4.7
CMAESCC	1.7	3.1	3.3	4.0	2.9	2.2	6.7	5.7

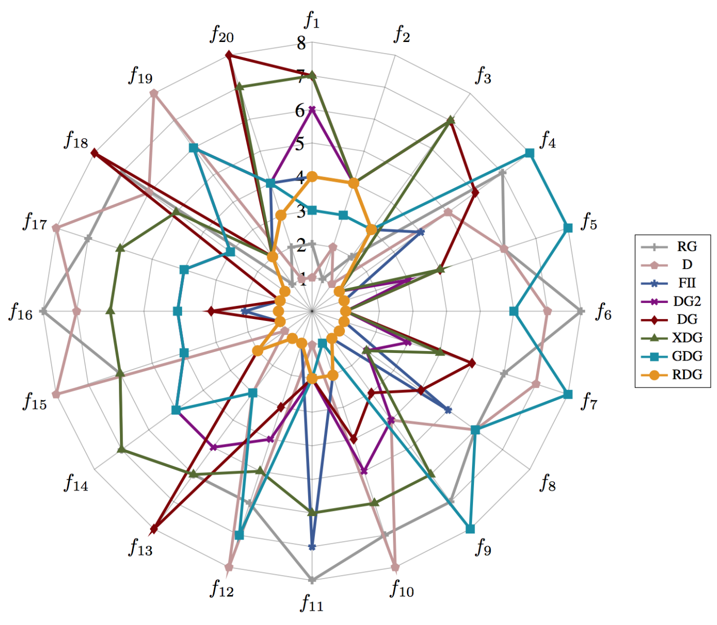
ranking across all the benchmark problems investigated. The RDG method generally uses the smallest number of FEs in the decomposition stage, assigning more computational resources to optimize the problems, when compared against the other automatic decomposition methods.

On the fully separable and some fully non-separable problems, the RDG method is outperformed by the two manual decomposition methods – D (delta grouping) and RG, as RDG does not actually perform any decomposition for these problems. However, the performance of the D (delta grouping) and RG methods deteriorates quickly on the partially separable problems.

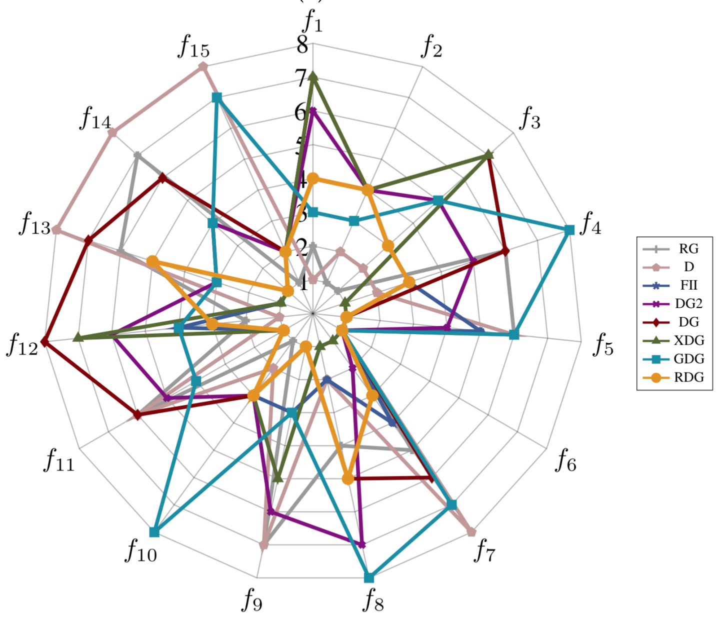
On some benchmark problems with separable decision variables, e.g., CEC'2010 f_{10}, the GDG method generates better solution quality than RDG. The reason for this is that GDG further divides the separable decision variables into several components. Interestingly, the GDG method achieves the first place when embedded into CMAESCC, however the last place when embedded into DECC, to solve the CEC'2010 f_9.

The DG method performs poorly on overlapping benchmark problems (e.g., CEC'2010 f_8, f_{13}, f_{18}, and f_{20}). The reason for this is that the DG method can not completely identify interaction between decision variables in an overlapping problem. Once all the variable interactions are identified and all the linked decision variables are placed into one component, the solution quality can be greatly improved by several magnitudes.

The DG2 method generates the best solution quality when embedded into the CMAESCC framework to solve the CEC'2013 f_{11} problem. The reason for this is that DG2 achieves 100% accuracy when used to decompose this problem, which is higher than RDG. However on the other benchmark problems, the RDG method generally obtains

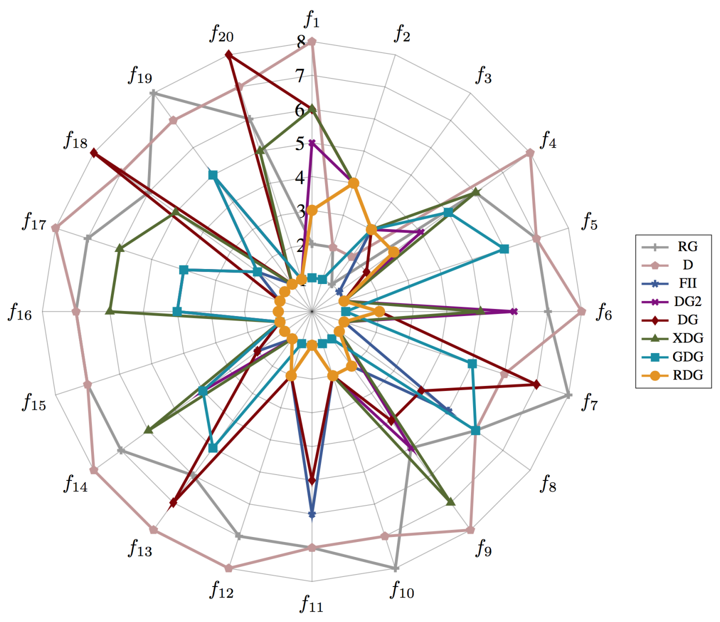


(a) CEC'2010

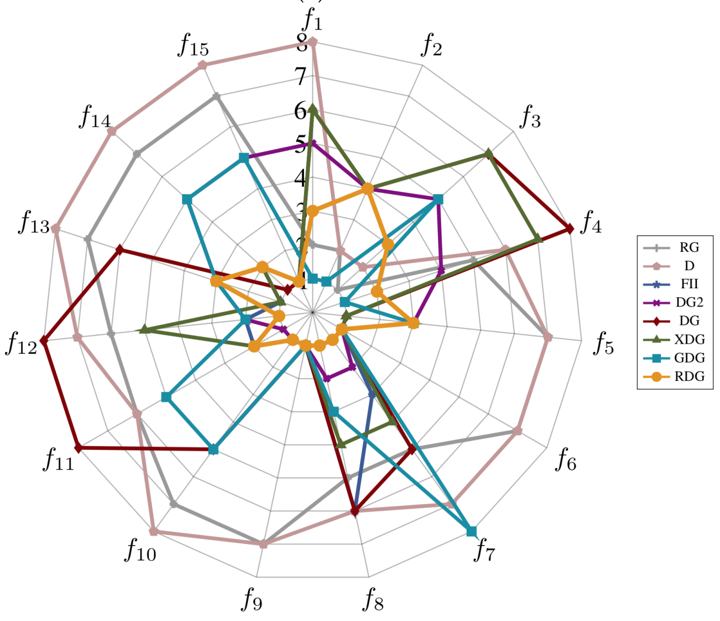


(b) CEC'2013

Figure 5.3: The radar chart of the performance of RDG, GDG, XDG, DG, DG2, FII, D (delta grouping), and RG when embedded in the DECC framework to solve the CEC'2010 and CEC'2013 benchmark problems. On each benchmark problem, the six methods are ranked from 1 to 8 (as labelled on the concentric circles) based on the results from 25 independent runs (Wilcoxon rank-sum tests (α=0.05) with Holm p-value correction).



(a) CEC'2010



(b) CEC'2013

Figure 5.4: The radar chart of the performance of RDG, GDG, XDG, DG, DG2, FII, D (delta grouping), and RG when embedded in the CMAESCC framework to solve the CEC'2010 and CEC'2013 benchmark problems. On each benchmark problem, the six methods are ranked from 1 to 8 (as labelled on the concentric circles) based on the results from 25 independent runs (Wilcoxon rank-sum tests (α=0.05) with Holm p-value correction).

equally well or statistically better solution quality than DG2.

The FII method performs well across the benchmark problems investigated. It achieves the second place according to the average ranking. However, on the benchmark problems where the decomposition by FII is less efficient e.g., CEC'2010 f_8, the RDG method can generate significantly better solution quality than FII.

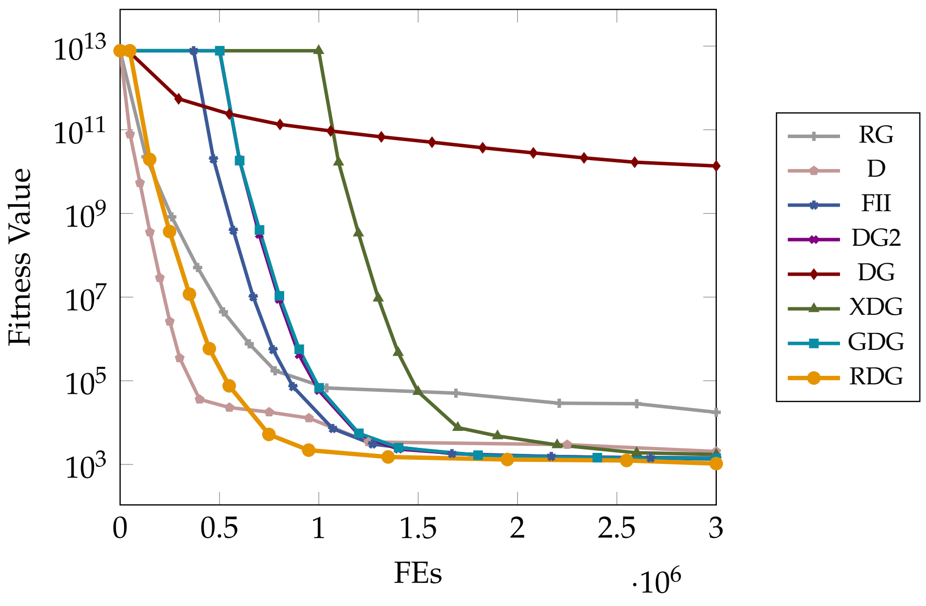
The DECC-D (with delta grouping) algorithm achieves much better results than the other DECC based algorithms when used to solve the CEC'2010 f_3 and f_{11} benchmark problems. An interesting observation is that both f_3 and f_{11} are Ackley's functions [73]. Moreover, the DECC-D (with delta grouping) algorithm also performs well when used to solve the other Ackley's functions: CEC'2010 f_6, f_{16} and CEC'2013 f_3, f_6, f_{10}. To investigate why the DECC-D (with delta grouping) algorithm performs well on the Ackley's functions is a potential direction of future work.

The convergence curves of the eight decomposition methods when embedded into the DECC/or CMAESCC framework to solve the CEC'2010 f_{18} problem are shown in Figure 5.5. The RDG method uses the smallest number of FEs in the decomposition stage, therefore can generate better solution quality when compared against the other automatic decomposition methods.

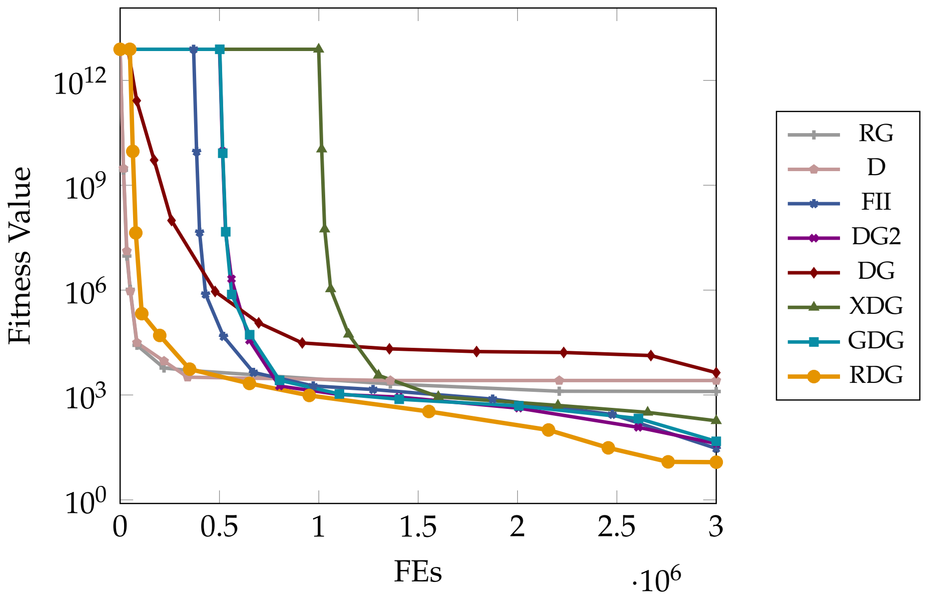
In the next phase of the experimental study, I compare the performance of DECC-RDG and CMAESCC-RDG against the performance of two state-of-the-art algorithms – MOS and MA-SW-Chains. The experimental results of each algorithm when used to solve the CEC'2010 and CEC'2013 benchmark problems are presented in Table 5.6 and Table 5.7 respectively.

The CMAESCC-RDG algorithm achieves the best solution quality on 22 out of 35 benchmark problems when compared against DECC-RDG, MOS and MA-SW-Chains. It does not perform well on the fully separable problems (CEC'2010 f_1 to f_3 and CEC'2013 f_1 to f_3). However on partially separable and fully non-separable problems, it generally achieves comparable or statistically better solution quality than the other algorithms (e.g., on CEC'2010 f_4 to f_9 and f_{11} to f_{19}).

The DECC-RDG algorithm achieves the best solution quality when used to solve the CEC'2010 f_{16} problem. On other 34 benchmark problems, DECC-RDG is outper-



(a) DECC



(b) CMAESCC

Figure 5.5: The convergence curves of the RDG, GDG, XDG, DG, DG2, FII, D (delta grouping) and RG methods when embedded into the DECC and CMAESCC frameworks to solve the CEC'2010 f_{18} problem. The horizontal axis represents the number of FEs used in the evolutionary process. The vertical axis represents the median of the best fitness found.

Table 5.6: The results of the DECC-RDG, CMAESCC-RDG, MOS and MA-SW-Chains algorithms when used to solve the CEC'2010 benchmark problems. The best performances are highlighted in bold (Wilcoxon rank-sum tests ($\alpha = 0.05$) with Holm p-value correction).

Func	Stats	DECC-RDG	CMAESCC-RDG	MOS	MA-SW-Chains
f_1	Mean	2.07e+00	2.84e+05	**1.50e-28**	3.80e-14
	Std	6.75e+00	2.28e+04	**5.55e-28**	4.91e-14
f_2	Mean	4.38e+03	4.42e+03	**0.00e+00**	8.40e+02
	Std	1.72e+02	1.76e+02	**0.00e+00**	4.88e+01
f_3	Mean	1.65e+01	1.05e+00	**0.00e+00**	5.76e-13
	Std	3.35e-01	3.49e-01	**0.00e+00**	2.73e-13
f_4	Mean	6.68e+11	**1.04e+06**	5.16e+11	2.97e+11
	Std	3.33e+11	**1.02e+05**	1.85e+11	6.19e+10
f_5	Mean	1.28e+08	**9.52e+07**	4.93e+08	2.18e+08
	Std	1.92e+07	**2.22e+07**	6.93e+07	5.75e+07
f_6	Mean	1.61e+01	**9.17e-01**	1.97e+07	1.42e+05
	Std	3.64e-01	**4.23e-01**	1.15e+05	3.96e+05
f_7	Mean	2.16e+01	**7.40e-18**	3.54e+07	1.17e+02
	Std	7.56e+01	**8.35e-19**	3.22e+07	2.37e+02
f_8	Mean	1.59e+05	**6.37e+05**	3.75e+06	6.90e+06
	Std	7.97e+05	**1.49e+06**	4.40e+06	1.90e+07
f_9	Mean	4.69e+07	**4.82e+06**	1.13e+07	1.49e+07
	Std	5.21e+06	**3.88e+05**	1.61e+06	1.61e+06
f_{10}	Mean	4.33e+03	2.79e+03	6.28e+03	**2.01e+03**
	Std	1.39e+02	1.17e+02	3.12e+02	**1.59e+02**
f_{11}	Mean	1.03e+01	**3.58e-02**	3.08e+01	3.86e+01
	Std	8.50e-01	**1.79e-01**	6.07e+00	8.06e+00
f_{12}	Mean	1.53e+03	**4.22e-22**	4.39e+03	3.24e-06
	Std	4.66e+02	**8.38e-23**	2.92e+03	5.78e-07
f_{13}	Mean	7.12e+02	**4.78e+00**	3.32e+02	9.83e+02
	Std	2.52e+02	**3.98e+00**	1.19e+02	5.66e+02
f_{14}	Mean	3.47e+08	**3.91e-20**	2.05e+07	3.25e+07
	Std	2.31e+07	**2.11e-20**	3.60e+06	2.46e+06
f_{15}	Mean	5.84e+03	**1.94e+03**	1.29e+04	2.68e+03
	Std	1.01e+02	**1.10e+02**	3.48e+02	9.95e+01
f_{16}	Mean	**2.67e-13**	8.43e-13	3.96e+02	9.95e+01
	Std	**9.81e-15**	2.10e-14	3.47e+00	1.53e+01
f_{17}	Mean	4.07e+04	**6.90e-24**	8.45e+03	1.27e+00
	Std	2.55e+03	**2.05e-25**	5.04e+03	1.24e-01
f_{18}	Mean	1.20e+03	**1.50e+01**	8.96e+02	1.57e+03
	Std	1.07e+02	**7.19e+00**	4.03e+02	6.73e+02
f_{19}	Mean	1.71e+06	**5.46e+03**	5.49e+05	3.80e+05
	Std	8.91e+04	**7.07e+02**	8.38e+04	2.34e+04
f_{20}	Mean	6.96e+03	8.26e+02	**9.23e+01**	1.06e+03
	Std	1.27e+04	6.35e+01	**8.99e+01**	9.38e+01

Table 5.7: The results of the DECC-RDG, CMAESCC-RDG, MOS and MA-SW-Chains algorithms when used to solve the CEC'2013 benchmark problems. The best performances are highlighted in bold (Wilcoxon rank-sum tests (α=0.05) with Holm p-value correction).

Func	Stats	DECC-RDG	CMAESCC-RDG	MOS	MA-SW-Chains
f_1	Mean	3.73e+01	2.89e+05	**3.10e-29**	1.34e-12
	Std	1.24e+02	3.27e+04	**4.53e-29**	2.45e-12
f_2	Mean	1.27e+04	4.68e+03	**1.83e+01**	1.25e+03
	Std	6.40e+02	1.77e+02	**4.65e+00**	1.05e+02
f_3	Mean	2.13e+01	2.03e+01	**1.65e-13**	6.85e-13
	Std	1.64e-02	4.96e-02	**1.02e-13**	2.12e-13
f_4	Mean	4.44e+10	**5.90e+06**	1.40e+10	3.81e+09
	Std	1.77e+10	**6.56e+05**	7.65e+09	2.73e+09
f_5	Mean	4.32e+06	**2.20e+06**	1.15e+07	**2.25e+06**
	Std	4.86e+05	**3.76e+05**	1.82e+06	**1.30e+06**
f_6	Mean	1.06e+06	9.95e+05	9.83e+05	**1.86e+04**
	Std	1.21e+03	2.88e+01	8.22e+03	**2.54e+04**
f_7	Mean	7.45e+07	**1.29e-17**	2.33+07	3.85e+06
	Std	2.37e+07	**2.84e-17**	3.62e+07	6.34e+05
f_8	Mean	3.95e+15	**9.74e+06**	1.65e+15	4.62e+13
	Std	1.45e+15	**5.83e+06**	1.76e+15	9.02e+12
f_9	Mean	4.82e+08	1.65e+08	9.02e+08	**1.44e+08**
	Std	3.06e+07	4.16e+07	1.04e+08	**1.55e+07**
f_{10}	Mean	9.44e+07	9.12e+07	6.66e+07	**3.72e+04**
	Std	2.06e+05	1.53e+06	3.01e+07	**6.25e+04**
f_{11}	Mean	5.38e+08	**1.62e+07**	3.87e+10	2.10e+08
	Std	1.34e+08	**6.11e+05**	1.06e+11	2.35e+07
f_{12}	Mean	4.85e+03	9.81e+02	**8.64e+01**	1.24e+03
	Std	3.06e+03	7.30e+01	**7.82e+01**	8.33e+01
f_{13}	Mean	3.06e+09	**2.47e+06**	1.09e+09	3.58e+07
	Std	6.68e+08	**3.83e+05**	7.69e+08	4.30e+07
f_{14}	Mean	1.05e+09	**2.76e+07**	6.65e+09	1.45e+08
	Std	6.88e+08	**1.49e+06**	1.62e+10	1.60e+07
f_{15}	Mean	9.75e+06	**2.19e+06**	1.33e+08	5.98e+06
	Std	1.91e+06	**2.28e+05**	6.05e+07	1.42e+06

formed by CMAESCC-RDG. It may indicate that the component optimizer used by the CMAESCC framework is more effective than that used by the DECC framework. It is important to note that on CEC'2010 f_8, CMAESCC-RDG is reported to outperform DECC-RDG, in spite of the fact that the mean of the best solution found by CMAESCC-RDG (7.97e+05) is worse than that of DECC-RDG (1.59e+05). The reason for this is that the Wilcoxon rank-sum test examines whether two samples are from continuous distributions with equal medians instead of means. The median of the best solution found by CMAESCC-RDG is 2.16e-17, which is much smaller (better) than that found by DECC-RDG (3.06e+00).

The MOS algorithm achieves the best results when used to solve the fully separable problems (CEC'2010 f_1-f_3, and CEC'2013 f_1-f_3). However, on partially separable problems and fully non-separable problems, it is generally outperformed by the CMAESCC-RDG algorithm.

The MA-SW-Chains algorithm performs well on the CEC'2013 benchmark problems. It achieves the best results when used to solve the CEC'2013 f_5, f_6, f_9, and f_{10} problems. However, on the other partially separable and fully non-separable benchmark problems, it is consistently outperformed by CMAESCC-RDG. In some cases, the best solution found by CMAESCC-RDG is much better than that found by MA-SW-Chains.

5.5 Conclusion

In this chapter, I have investigated the important role that problem decomposition has on the performance of CC algorithms when used to solve large-scale BCOPs. A robust decomposition method – RDG – was proposed, which can accurately decompose high dimensional problems using $\mathcal{O}(n\log(n))$ FEs based on a measure of non-linearity between decision variables. Significantly, RDG outperformed seven other decomposition methods when embedded into the DECC/CMAESCC framework and tested across a suite of large-scale benchmark BCOPs.

Chapter 6
Relaxed Joint Mutual Information for Feature Selection

THE AIM of a classification problem is to identify the class label of a given object based on a training data set where the class labels of objects are known. The objects are typically represented by a set of features. Therefore, the relevance of features to class labels is of key importance to the classification accuracy. Unfortunately in many real-world applications, many features are irrelevant and redundant to class labels [12, 28, 59]. Therefore it is essential to select a subset of informative features in order to improve the classification accuracy. However, feature selection is a difficult task partially due to the interactions between features – an irrelevant feature may become highly relevant to class labels when combined with other feature(s) [200, 150].

The existing information theoretic feature selection methods only consider low-order interactions between features – typically the Mutual Information (MI) between two features. A recent proposed method – Relaxed Minimum Redundancy Maximum Relevance (RelaxMRMR) [185] takes the MI between three features (three-way) into consideration when selecting a subset of informative features. However, the computational cost of the RelaxRMRM method is very high [185].

In this chapter, I further investigate the two research questions of this thesis, which are introduced in Section 1.1. However, I shift my focus from the optimization domain to the classification (more specifically feature selection) domain: $\mathfrak{RQ}6.2$ *How can feature interactions be accurately and efficiently identified in a given classification problem?* and $\mathfrak{RQ}6.1$ *How can feature interactions be used to effectively solve a given classification problem?* In fact, feature selection problems are closely related to optimization problems: the former seek high classification accuracy by selecting a subset of informative features; while the latter aim for high fitness by adjusting the values of decision variables. A large number of Evolutionary Algorithms (EAs) have been successfully used to solve feature selection problems [194, 193].

To investigate the two research questions, I relax the assumptions made on the underlying probability density function of features and class labels. A high-order feature evaluation criterion is derived which considers the class-conditional MI between three features. To reduce the computational cost, the three-way (class-conditional) MI is approximated from the two-way (class-conditional) MI between features. I assume that the percentage of (class-conditional) information of a candidate feature X_i shared with X_j is equal to the percentage of (class-conditional) MI between X_m and X_i shared with X_j. The proposed feature selection criterion is then evaluated on 16 data sets using a widely-used classification algorithm. It achieved statistically significantly better classification accuracy when compared against four other feature selection methods in most cases.

The remainder of this chapter is organized as follows. Section 6.1 describes the state-of-the-art information theoretic feature selection methods and the corresponding assumptions made on data distributions. Section 6.2 describes the proposed feature selection method in detail. Section 6.3 describes experiments to evaluate the proposed feature selection method. Section 6.4 presents and analyzes the experimental results. Section 6.5 concludes this chapter.

6.1 Related Work: Information Theoretic Feature Selection Methods

This section describes the related feature selection methods based on MI in detail. The assumptions made on the underlying probability density function of the features and the class labels are also investigated.

Entropy. *The entropy of a discrete random variable X is defined as*

$$H(X) = -\sum_{x \in \mathcal{X}} p(x) \log \big(p(x)\big), \tag{6.1}$$

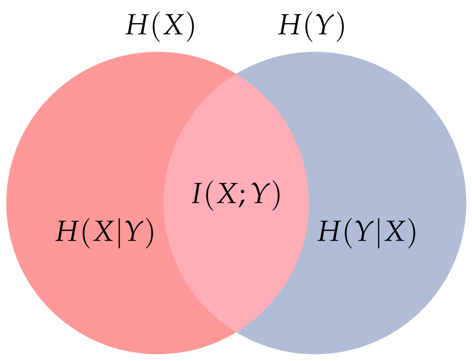


Figure 6.1: The MI between two random variables X and Y. $H(X|Y)$ is the entropy of X conditional on Y.

where x denotes a possible value of the random variable X, while $\mathcal{X}$ denotes the set of all possible values of X.

Mutual Information. *The MI between two discrete random variables X and Y is defined as*

$$I(X;Y) = -\sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x,y) \log \left(\frac{p(x,y)}{p(x)p(y)} \right). \tag{6.2}$$

The MI between X and Y indicates how much information of X (Y) can be obtained from knowing Y (X), as illustrated in Fig. 6.1. The MI has been widely used in the feature selection task. The aim of information theoretic feature selection methods is to select a subset of k features that maximizes the MI between the selected features $(\mathbf{S})$ and the class labels C:

$$\mathbf{S} = \underset{\mathbf{S} \subset \mathbf{X}, |\mathbf{S}|=k}{\arg\max}\, I(\mathbf{S};C), \tag{6.3}$$

where $\mathbf{X}$ is the set of all features. However in practice, it is unlikely to accurately estimate the high-order MI $\big(I(\mathbf{S};C)\big)$, as a): the sample size is typically limited; and b) the computational resource is limited [43, 12, 28, 59]. Therefore, many efforts have been made to approximate the high-order MI by low-order MI. In the existing literature, the high-order MI $\big(I(\mathbf{S};C)\big)$ is typically calculated from the MI between two features and/or the class labels under certain assumptions made on the data distribution [15]. Then global (e.g., quadratic programming [147, 118]) or local (e.g., sequential forward selection [67, 135]) search techniques can be used to identify the most informative feature subset.

A representative global approach is modelling the feature selection problem as a quadratic programming problem, and using the quadratic programming techniques to solve it to the global optimum [147, 118]. The global approach is usually computationally expensive. Therefore, I focus on the local search method – sequential forward selection [67, 135], which can greatly improve the search efficiency. The sequential forward selection method selects a candidate feature X_m at a time such that the MI between the selected features ($\mathbf{S}$) and the class labels C is maximized:

$$X_m = \underset{X_m \in \mathbf{X}_{\backslash \mathbf{S}}}{\arg\max}\, I(X_m, \mathbf{S}; C), \tag{6.4}$$

where $\mathbf{X}_{\backslash \mathbf{S}}$ denotes the set of candidate features not in $\mathbf{S}$ (features have not been selected). As the following equation holds true:

$$I(X_m, \mathbf{S}; C) = I(\mathbf{S}; C) + I(X_m; C|\mathbf{S}), \tag{6.5}$$

and $I(\mathbf{S}; C)$ is a constant for a fixed feature set $\mathbf{S}$, thus

$$X_m = \underset{X_m \in \mathbf{X}_{\backslash \mathbf{S}}}{\arg\max} \left\{ I(X_m; C|\mathbf{S}) \right\}. \tag{6.6}$$

Therefore, the sequential forward selection method will select a candidate feature X_m at a time to maximize the MI between X_m and the class labels C conditional on the selected feature subset $\mathbf{S}$.

In the following, I describe the assumptions made on the distribution of features and class labels, and derive the corresponding feature evaluation criteria. I use the notation $J(X_m)$ to denote an evaluation criterion of a candidate feature X_m:

$$J(X_m) = I(X_m; C|\mathbf{S}). \tag{6.7}$$

Assumption 6.1. *The features are pairwise independent:* $p(x_i, x_j) = p(x_i)p(x_j)$*, for all* $X_i, X_j \in \mathbf{X}$.

Assumption 6.2. *The features are pairwise class-conditionally independent:* $p(x_i, x_j|c) = p(x_i|c)\, p(x_j|c)$*, for all* $X_i, X_j \in \mathbf{X}$.

Theorem 6.1. *Under Assumption 6.1 and Assumption 6.2, the evaluation criterion of a candidate feature X_m, shown in Eq. (6.7), is equivalent to:*

$$J_{mim}(X_m) = I(X_m; C). \tag{6.8}$$

This method evaluates a candidate feature X_m based on the feature relevance: the MI between X_m and the class labels C. It does not consider any interaction between features. This method is known as Mutual Information Maximization (MIM) or Maximum Relevance method [70].

Assumption 6.3. *The selected features in* **S** *are conditionally independent given any unselected feature X_m in $\mathbf{X}_{\backslash \mathbf{S}}$:*

$$p(\mathbf{s}|x_m) = \prod_{X_i \in \mathbf{S}} p(x_i|x_m). \tag{6.9}$$

Theorem 6.2. *Under Assumption 6.2 and Assumption 6.3, the evaluation criterion of a candidate feature X_m, shown in Eq. (6.7), is equivalent to:*

$$J_{mifs}(X_m) = I(X_m; C) - \sum_{X_i \in \mathbf{S}} I(X_m; X_i). \tag{6.10}$$

This method takes into account the pairwise feature interactions: the two-way MI between features. It is equivalent to a version of the Mutual Information Feature Selection (MIFS) [10] method. After normalization, it becomes the well-known Minimum-Redundancy Maximum-Relevance (MRMR) [129] method:

$$J_{mrmr}(X_m) = I(X_m; C) - \frac{1}{|\mathbf{S}|} \sum_{X_i \in \mathbf{S}} I(X_m; X_i). \tag{6.11}$$

Assumption 6.4. *The selected features in* **S** *are conditionally independent given any unselected feature $X_m \in \mathbf{X}_{\backslash \mathbf{S}}$ and the class C:*

$$p(\mathbf{s}|x_m, c) = \prod_{X_i \in \mathbf{S}} p(x_i|x_m, c). \tag{6.12}$$

Theorem 6.3. *Under Assumption 6.3 and Assumption 6.4, the evaluation criterion of a candidate* *feature X_m, shown in Eq. (6.7), is equivalent to:*

$$J_{cife}(X_m) = I(X_m;C) - \sum_{X_i \in \mathbf{S}} \big(I(X_m;X_i) - I(X_m;X_i|C)\big). \tag{6.13}$$

This feature evaluation criterion is known as Conditional Informative Feature Extraction (CIFE) [76]. It takes feature relevance $\big(I(X_m;C)\big)$, redundancy $\big(I(X_m;X_i)\big)$ and complementarity $\big(I(X_m;X_i|C)\big)$ into consideration [184]. After normalization, it becomes the Joint Mutual Information (JMI) [195, 97] method:

$$J_{jmi}(X_m) = I(X_m;C) - \frac{1}{|\mathbf{S}|}\sum_{X_i \in \mathbf{S}} \big(I(X_m;X_i) - I(X_m;X_i|C)\big). \tag{6.14}$$

Assumption 6.5. *The selected features in* **S** *are conditionally independent given any unselected feature $X_m \in \mathbf{X}_{\backslash \mathbf{S}}$ and any selected feature $X_j \in \mathbf{S}$:*

$$p(\mathbf{s}|x_m) = p(x_j|x_m) \prod_{\substack{X_i \in \mathbf{S} \\ i \neq j}} p(x_i|x_m, x_j). \tag{6.15}$$

Theorem 6.4. *Under Assumption 6.4 and Assumption 6.5, the evaluation criterion of a candidate feature X_m, shown in Eq. (6.7), is equivalent to:*

$$J'_{rmrmr}(X_m) = I(X_m;C) + \sum_{X_i \in \mathbf{S}} I(X_m;X_i|C) - \frac{1}{|\mathbf{S}|}\sum_{X_i \in \mathbf{S}} \big(I(X_m;X_i) + \sum_{\substack{X_j \in \mathbf{S} \\ j \neq i}} I(X_m;X_i|X_j)\big). \tag{6.16}$$

This feature selection criterion considers the MI between three features $\big(I(X_m;X_i|X_j)\big)$. After normalization, it becomes the Relaxed Minimum-Redundancy Maximum-Relevance (RelaxMRMR) method [185]:

$$\begin{aligned} J_{rmrmr}(X_m) = I(X_m;C) &- \frac{1}{|\mathbf{S}|}\sum_{X_i \in \mathbf{S}} \big(I(X_m;X_i) - I(X_m;X_i|C)\big) \\ &- \frac{1}{|\mathbf{S}|(|\mathbf{S}|-1)} \sum_{\substack{X_i \in \mathbf{S} \\ i \neq j}} \sum_{X_j \in \mathbf{S}} I(X_m;X_i|X_j). \end{aligned} \tag{6.17}$$

The RelaxMRMR method is effective in terms of selecting a subset of informative features.

However, the computational cost of the RelaxMRMR is extremely high – more than 100 times greater than the MRMR method [185].

6.2 Relaxed Joint Mutual Information

In this section, I further relax the assumptions made on the underlying distribution of the features and class labels. Under the relaxed assumption, a new feature selection method is derived which takes the class-conditional MI between three features into consideration.

Assumption 6.6. *The selected features in* $\mathbf{S}$ *are class-conditionally independent given any unselected feature* $X_m \in \mathbf{X}_{\backslash \mathbf{S}}$ *and any selected feature* $X_j \in \mathbf{S}$*:*

$$p(\mathbf{s}|x_m, c) = p(x_j|x_m, c) \prod_{\substack{X_i \in \mathbf{S} \\ i \neq j}} p(x_i|x_m, x_j, c). \tag{6.18}$$

Theorem 6.5. *Under Assumption 6.5 and Assumption 6.6, the evaluation criterion of a candidate feature* X_m*, shown in Eq. (6.7), is equivalent to:*

$$\begin{aligned} J'_{rjmi}(X_m) = I(X_m; C) - \frac{1}{|\mathbf{S}|} \sum_{X_i \in \mathbf{S}} \big(I(X_m; X_i) \\ - I(X_m; X_i|C)\big) - \frac{1}{|\mathbf{S}|} \sum_{\substack{X_i \in \mathbf{S} \\ i \neq j}} \sum_{X_j \in \mathbf{S}} \big(I(X_m; X_i|X_j) - I(X_m; X_i|X_j, C)\big). \end{aligned} \tag{6.19}$$

Proof. As $I(A;B) - I(A;B|C) = I(A;C) - I(A;C|B)$ holds,

$$I(X_m; C|\mathbf{S}) = I(X_m; C) - I(X_m; \mathbf{S}) + I(X_m; \mathbf{S}|C). \tag{6.20}$$

The following two equations are true:

$$I(X_m; \mathbf{S}) = H(\mathbf{S}) - H(\mathbf{S}|X_m), \tag{6.21}$$

$$I(X_m; \mathbf{S}|C) = H(\mathbf{S}|C) - H(\mathbf{S}|X_m, C). \tag{6.22}$$

Therefore,

$$I(X_m;C|\mathbf{S}) = I(X_m;C) - \big(H(\mathbf{S}) - H(\mathbf{S}|X_m)\big) + \big(H(\mathbf{S}|C) - H(\mathbf{S}|X_m,C)\big). \tag{6.23}$$

As $H(\mathbf{S})$ and $H(\mathbf{S}|C)$ are constants for given C and $\mathbf{S}$, the evaluation criterion is equivalent to

$$J'_{rjmi}(X_m) = I(X_m;C) + H(\mathbf{S}|X_m) - H(\mathbf{S}|X_m,C)\big). \tag{6.24}$$

If Assumption 6.5 holds,

$$H(\mathbf{S}|X_m) = H(X_j|X_m) + \sum_{\substack{X_i\in\mathbf{S}\\ i\neq j}} H(X_i|X_m,X_j), \tag{6.25}$$

for any $1 \le j \le |\mathbf{S}|$. Therefore,

$$H(\mathbf{S}|X_m) = \frac{1}{|\mathbf{S}|}\sum_{X_j\in\mathbf{S}}\Big(H(X_j|X_m) + \sum_{\substack{X_i\in\mathbf{S}\\ i\neq j}} H(X_i|X_m,X_j)\Big). \tag{6.26}$$

Similarly, if Assumption 6.6 holds,

$$H(\mathbf{S}|X_m,C) = H(X_j|X_m,C) + \sum_{\substack{X_i\in\mathbf{S}\\ i\neq j}} H(X_i|X_m,X_j,C), \tag{6.27}$$

for any $1 \le j \le |\mathbf{S}|$. Thus,

$$H(\mathbf{S}|X_m,C) = \frac{1}{|\mathbf{S}|}\sum_{X_j\in\mathbf{S}}\big(H(X_j|X_m,C) + \sum_{\substack{X_i\in\mathbf{S}\\ i\neq j}} H(X_i|X_m,X_j,C)\big). \tag{6.28}$$

Substituting Eq. (6.26) and (6.28) into (6.24),

$$\begin{aligned} J'_{rjmi}(X_m) = I(X_m;C) &+ \frac{1}{|\mathbf{S}|}\sum_{X_j\in\mathbf{S}}\Big(H(X_j|X_m) + \sum_{\substack{X_i\in\mathbf{S}\\ i\neq j}} H(X_i|X_m,X_j)\Big) \\ &- \frac{1}{|\mathbf{S}|}\sum_{X_j\in\mathbf{S}}\Big(H(X_j|X_m,C) + \sum_{\substack{X_i\in\mathbf{S}\\ i\neq j}} H(X_i|X_m,X_j,C)\Big). \end{aligned} \tag{6.29}$$

The following equation is true:

$$H(X_j|X_m) = H(X_j) - I(X_j; X_m), \tag{6.30}$$

and $H(X_j)$ is a constant for a given X_j. Therefore

$$H(X_j|X_m) \propto -I(X_j; X_m). \tag{6.31}$$

Similarly,

$$H(X_i|X_m, X_j) \propto -I(X_i; X_m|X_j), \tag{6.32}$$

$$H(X_j|X_m, C) \propto -I(X_j; X_m|C), \tag{6.33}$$

$$H(X_i|X_m, X_j, C) \propto -I(X_i; X_m|X_j, C). \tag{6.34}$$

Therefore,

$$\begin{aligned} J'_{rjmi}(X_m) \propto I(X_m; C) &- \frac{1}{|\mathbf{S}|} \sum_{X_j \in \mathbf{S}} \Big(I(X_j; X_m) + \sum_{\substack{X_i \in \mathbf{S} \\ i \neq j}} I(X_i; X_m|X_j) \Big) \\ &+ \frac{1}{|\mathbf{S}|} \sum_{X_j \in \mathbf{S}} \Big(I(X_j; X_m|C) + \sum_{\substack{X_i \in \mathbf{S} \\ i \neq j}} I(X_i; X_m|X_j, C) \Big). \end{aligned} \tag{6.35}$$

Thus,

$$\begin{aligned} J'_{rjmi}(X_m) \propto I(X_m; C) &- \frac{1}{|\mathbf{S}|} \sum_{X_i \in \mathbf{S}} \Big(I(X_m; X_i) - I(X_m; X_i|C) \Big) \\ &- \frac{1}{|\mathbf{S}|} \sum_{\substack{X_i \in \mathbf{S} \\ i \neq j}} \sum_{X_j \in \mathbf{S}} \Big(I(X_m; X_i|X_j) - I(X_m; X_i|X_j, C) \Big). \end{aligned} \tag{6.36}$$

□

It has been empirically shown that a selection criterion with normalization is more effective than that without normalization when used to select informative features [185, 15]. Therefore, I normalize each term in Eq. (6.19) to the same scale:

$$\begin{aligned} J_{rjmi}(X_m) = I(X_m;C) &- \frac{1}{|\mathbf{S}|}\sum_{X_i\in\mathbf{S}}\big(I(X_m;X_i) - I(X_m;X_i|C)\big) \\ &- \frac{1}{|\mathbf{S}|(|\mathbf{S}|-1)}\sum_{\substack{X_i\in\mathbf{S}\\ i\neq j}}\sum_{X_j\in\mathbf{S}}\big(I(X_m;X_i|X_j) - I(X_m;X_i|X_j,C)\big). \end{aligned} \tag{6.37}$$

This feature evaluation criterion takes both the MI between three features $I(X_m;X_i|X_j)$ and the class-conditional MI between three features $I(X_m;X_i|X_j,C)$ into consideration. It can be regarded as an extension of the Joint Mutual Information (JMI) method. Therefore, I denote it as Relaxed Joint Mutual Information (RelaxJMI) method.

However, the calculation of the (class-conditional) MI between three features is computationally expensive [185]. To save computational cost, I approximate the three-way (class-conditional) MI using the two-way (class-conditional) MI under the following assumptions.

Assumption 6.7. *The percentage of information of X_i $\big(H(X_i)\big)$ carried by X_j is equal to the percentage of MI between X_m and X_i $\big(I(X_m;X_i)\big)$ carried by X_j, where $X_m \in \mathbf{X}_{\backslash\mathbf{S}}$, and $X_i, X_j \in \mathbf{S}$.*

$$\frac{I(X_i;X_j)}{H(X_i)} = \frac{I(X_m;X_j;X_i)}{I(X_m;X_i)}. \tag{6.38}$$

This assumption is illustrated in Fig. 6.2.

Assumption 6.8. *The percentage of information of X_i given C $\big(H(X_i|C)\big)$ carried by X_j is equal to the percentage of MI between X_m and X_i given C $\big(I(X_m;X_i|C)\big)$ carried by X_j, where $X_m \in \mathbf{X}_{\backslash\mathbf{S}}$, and $X_i, X_j \in \mathbf{S}$.*

$$\frac{I(X_i;X_j|C)}{H(X_i|C)} = \frac{I(X_m;X_j;X_i|C)}{I(X_m;X_i|C)}. \tag{6.39}$$

Under Assumption 6.7,

$$\begin{aligned} I(X_m;X_i|X_j) &= \frac{H(X_i|X_j)}{H(X_i)}I(X_m;X_i) \\ &= \frac{H(X_i) - I(X_i;X_j)}{H(X_i)}I(X_m;X_i) \\ &= I(X_m;X_i) - \frac{I(X_i;X_j)I(X_m;X_i)}{H(X_i)}. \end{aligned} \tag{6.40}$$

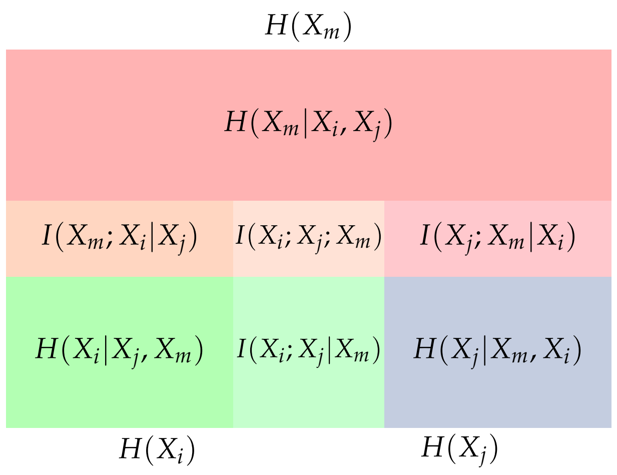


Figure 6.2: The illustration of the assumptions made to approximate the three-order MI using the two-order MI between features.

Similarly under Assumption 6.8,

$$I(X_m;X_i|X_j,C) = I(X_m;X_i|C) - \frac{I(X_i;X_j|C)I(X_m;X_i|C)}{H(X_i|C)}. \tag{6.41}$$

Substituting Eq. (6.40) and (6.41) into (6.37),

$$\begin{aligned}\tilde{J}_{rjmi}(X_m) = I(X_m;C) &- \frac{2}{|\mathbf{S}|}\sum_{X_i\in\mathbf{S}}\big(I(X_m;X_i) - I(X_m;X_i|C)\big)\\ &+ \frac{1}{|\mathbf{S}|(|\mathbf{S}|-1)}\sum_{\substack{X_i\in\mathbf{S}\\ i\neq j}}\sum_{X_j\in\mathbf{S}}\frac{I(X_i;X_j)I(X_m;X_i)}{H(X_i)}\\ &- \frac{1}{|\mathbf{S}|(|\mathbf{S}|-1)}\sum_{\substack{X_i\in\mathbf{S}\\ i\neq j}}\sum_{X_j\in\mathbf{S}}\frac{I(X_i;X_j|C)I(X_m;X_i|C)}{H(X_i|C)}.\end{aligned} \tag{6.42}$$

6.3 Experimental Methodology

To evaluate the efficacy of the proposed RelaxJMI method, detailed numerical experiments are designed to explore the following question:

Q6.1 Can the proposed RelaxJMI method $\big($Eq. (6.42)$\big)$ select more informative features when compared against other state-of-the-art methods?

Table 6.1: The description of the data sets used in the experiments. The last column denotes the ratio between the sample size (n) and the median of the selected features (m) with the number of classes (c): $n/(mc)$. The ratio is an indicator of the difficulty of a data set for a feature selection task [15].

Data	#Features	#Instances	#Classes	Ratio
Wine	13	178	3	9.88
Heart	13	270	2	22.5
Parkinsons	22	195	2	8.86
Spect	22	267	2	12.1
Ionosphere	34	351	2	10.3
Krvskp	36	3196	2	88.7
Landsat	36	6435	6	59.5
Waveform	40	5000	3	83.3
Lungcancer	56	32	3	0.42
Sonar	60	208	2	4.16
Splice	60	3175	3	42.3
Lungdiscrete	325	73	7	0.41
Lymphoma	4026	96	9	0.42
Tox_171	5748	171	4	1.71
Leukemia	7070	72	2	1.44
CLL_SUB_111	11340	111	3	1.48

To investigate Q6.1, the features selected by a method will be tested using a classification algorithm, and the effectiveness of the method is indicated by the classification accuracy.

Sixteen real-world data sets[1] are used in the empirical studies. The number of features, instances and classes of the data sets varies from 13, 32, and 2 to 11340, 5000, and 9 respectively, as shown in Table 6.1. The last column in Table 6.1 denotes the ratio between the sample size (n) and the median of the selected features (m) with the number of classes (c): $n/(mc)$. The ratio is an indicator of the difficulty of a data set for a feature selection task [15]. For data sets with continuous features, the Minimum Description Length method [30] is employed to evenly divide the continuous values into five bins.

[1]The data sets are available from the UCI Machine Learning Repository [75] `http://archive.ics.uci.edu/ml/` and the Scikit-feature Feature Selection Repository [71] `http://featureselection.asu.edu/datasets.php`.

Note that the discretization is only used for the feature selection procedure, while the classification procedure still uses the original continuous values.

For each data set with more than fifty features, the RelaxJMI method is used to incrementally select the top fifty features; while for data sets with less than fifty features, the RelaxJMI method is used to incrementally select all the features. For each subset of selected features, the K Nearest Neighbour with $K = 1$ (1NN) is employed to calculate the 10-fold cross validation error rate for each data set, following [187]. The 1NN is chosen for its simplicity and few assumptions made on the distribution of the data. This process is repeated 10 times, and the averaged cross validation error rate is recorded. Then the averaged cross-validation error rate is plotted against the number of features. For each run (10 runs in total), the cross validation error rate is averaged across the all size of feature subsets (from 1 to maximum number of features). The mean and standard deviation of the averaged cross validation rate across all the 10 runs are recorded.

The RelaxJMI method is compared with MIM [70], MRMR [129], JMI [195], and Conditional Mutual Information Maximization (CMIM) [31] methods. The Wilcoxon rank-sum test [158] (significance level α=0.05) is used to determine whether the experimental results obtained from the proposed RelaxJMI method are statistically better than the results from other feature selection methods in a pairwise fashion.

6.4 Experimental Results

The experimental results of the proposed RelaxJMI and the MIM, RMRM, JMI, CMIM methods are shown in Table 6.2. The number of Win/Tie/Loss of the RelaxJMI when compared against each of the methods is shown in the last row of Table 6.2. It shows that RelaxJMI method achieved comparable or better classification accuracy than the other feature selection methods when combined with 1NN to classify most of the 16 data sets.

The MIM method performed the worst among the five feature selection methods. It was clearly outperformed by the proposed RelaxJMI method. The reason for this is that MIM doesn't take the interaction between features into consideration. It only uses the feature relevance (MI between features and class labels) as the evaluation criterion.

Table 6.2: KNN (k=1) error rate (%) comparison between RelaxJMI and other MI based feature selection methods. '+'/'='/'-' indicates that RelaxJMI performs 'better'/'equally well'/'worse' when compared against other feature selection methods according to the Wilcoxon rank-sum test with 95% confidence interval ($\alpha = 0.05$).

Data	RelaxJMI	MIM	MRMR	JMI	CMIM
Wine	8.73 ± 0.27	$11.52 \pm 0.75(+)$	$11.11 \pm 0.16(+)$	$8.51 \pm 0.20(=)$	$8.31 \pm 0.33(-)$
Heart	31.69 ± 0.95	$30.97 \pm 0.52(=)$	$28.86 \pm 0.87(-)$	$30.45 \pm 0.46(-)$	$30.96 \pm 0.42(=)$
Parkinsons	10.80 ± 0.52	$15.06 \pm 0.77(+)$	$11.98 \pm 0.65(+)$	$11.99 \pm 0.63(+)$	$11.22 \pm 0.81(=)$
Spect	23.25 ± 0.82	$24.40 \pm 0.44(+)$	$23.06 \pm 0.61(=)$	$24.05 \pm 0.73(+)$	$24.39 \pm 0.97(+)$
Ionosphere	13.17 ± 0.34	$14.60 \pm 0.24(+)$	$13.91 \pm 0.22(+)$	$13.06 \pm 0.44(=)$	$12.85 \pm 0.30(-)$
Krvskp	8.51 ± 0.07	$10.74 \pm 0.22(+)$	$9.65 \pm 0.09(+)$	$8.92 \pm 0.12(+)$	$9.34 \pm 0.16(+)$
Landsat	19.99 ± 0.07	$22.13 \pm 0.11(+)$	$19.54 \pm 0.06(-)$	$20.14 \pm 0.07(+)$	$19.58 \pm 0.10(-)$
Waveform	27.68 ± 0.10	$28.91 \pm 0.18(+)$	$27.74 \pm 0.13(=)$	$27.66 \pm 0.13(=)$	$27.85 \pm 0.16(+)$
Lungcancer	45.73 ± 1.45	$54.69 \pm 2.07(+)$	$53.78 \pm 2.55(+)$	$47.01 \pm 3.40(+)$	$49.50 \pm 2.19(+)$
Sonar	18.02 ± 0.31	$19.65 \pm 0.85(+)$	$19.53 \pm 0.58(+)$	$17.90 \pm 0.53(=)$	$17.90 \pm 0.69(=)$
Splice	22.65 ± 0.07	$22.88 \pm 0.12(+)$	$22.55 \pm 0.16(=)$	$22.65 \pm 0.10(=)$	$22.83 \pm 0.12(+)$
Lungdiscrete	22.73 ± 2.06	$32.17 \pm 1.65(+)$	$23.60 \pm 2.04(=)$	$23.55 \pm 0.16(+)$	$21.98 \pm 1.65(=)$
Lymphoma	14.43 ± 1.40	$26.13 \pm 1.92(+)$	$14.33 \pm 1.53(=)$	$14.74 \pm 1.88(=)$	$14.18 \pm 1.91(=)$
Tox171	11.35 ± 1.55	$33.51 \pm 1.28(+)$	$23.22 \pm 1.44(+)$	$22.03 \pm 1.53(+)$	$23.77 \pm 1.59(+)$
Leukemia	4.11 ± 0.08	$4.32 \pm 0.06(=)$	$4.43 \pm 0.06(=)$	$4.00 \pm 0.06(=)$	$4.32 \pm 0.08(=)$
CLL_SUB_111	29.06 ± 2.44	$32.50 \pm 2.05(+)$	$28.13 \pm 1.75(=)$	$28.88 \pm 2.32(=)$	$27.57 \pm 1.94(-)$
Win/Tie/Loss	-	14/2/0	7/7/2	7/8/1	6/6/4

The MRMR and JMI methods performed better than the MIM method. However, both of the MRMR and JMI methods achieved overall no better results than the proposed RelaxJMI method. Noth that the main difference between the proposed RelaxJMI method with the MRMR/JMI method is that the RelaxJMI considers the interactions between three features while the MRMR/JMI method considers the interaction between two features. It implied that the proposed RelaxJMI method can select more informative features by considering high-order MI between features.

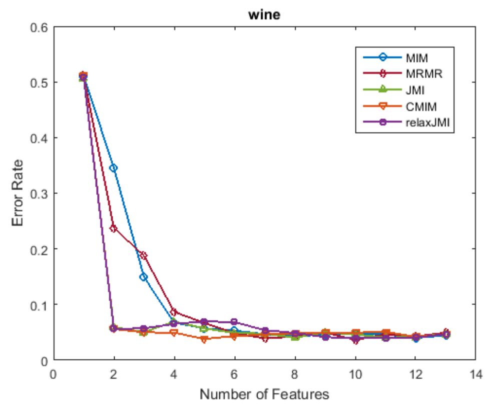
The CMIM method performed slightly worse than the proposed RelaxJMI method on the 16 data sets. The CMIM method incrementally selects the candidate features which are both individually informative and weakly interact with the selected features. Therefore, it places more emphasises on selecting the un-redundant features.

It is worth noting that the proposed RelaxJMI performed better on the data sets with low ratio value shown in Table 6.1, e.g., Lungcancer and Tox171. On some data sets with median or high ration value such as Heart, the proposed RelaxJMI achieved slightly worse results than the other methods. The explanation for this phenomenon is left for future work.

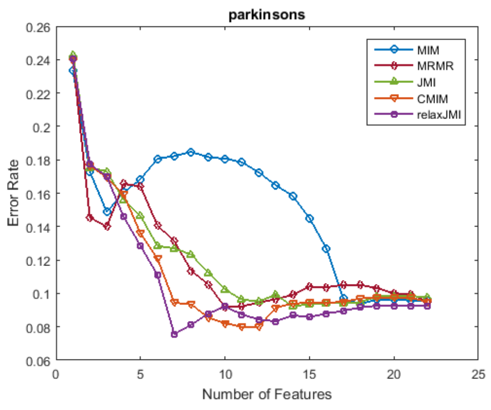
The classification error rate by the 1NN model against the number of features selected by each method on the selected data sets is plot in Fig. 6.3 and Fig. 6.4. It shows that the proposed RelaxJMI method achieved low classification error rate in most cases. For example in Fig. 6.4f, RelaxJMI consistently achieved the lowest classification error rate across all the number of selected features when compared against four other methods.

6.5 Conclusion

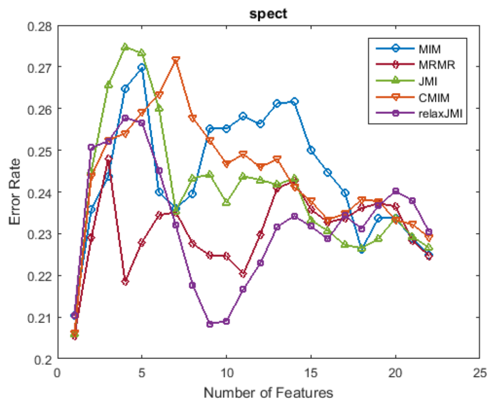
In this chapter, I have shifted my focus from the *optimization* to highly relevant *classification* domain, by investigating the effects of considering high-order feature interactions in the feature selection task. I relaxed the assumptions made on the probability density function of features and class labels, and derived a novel method – RelaxJMI – for feature selection. The RelaxJMI method takes the (class-conditional) MI between three features into consideration. To reduce computational cost, the (class-conditional) MI between



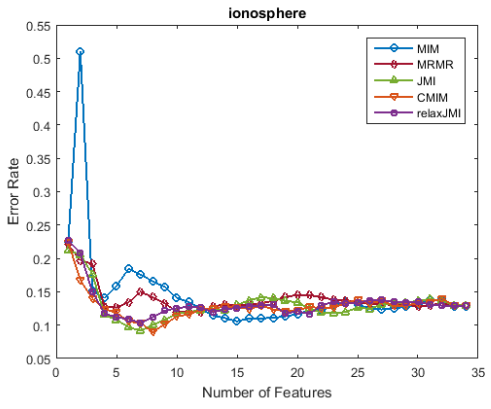
(a) Wine



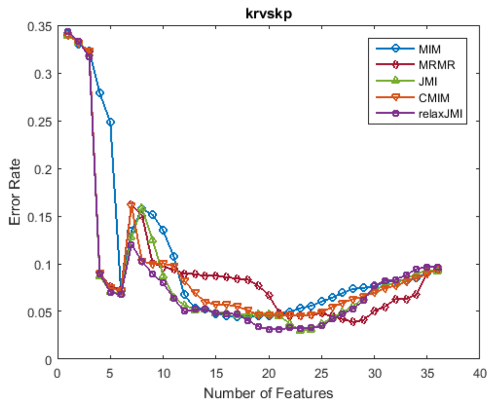
(b) Parkinsons



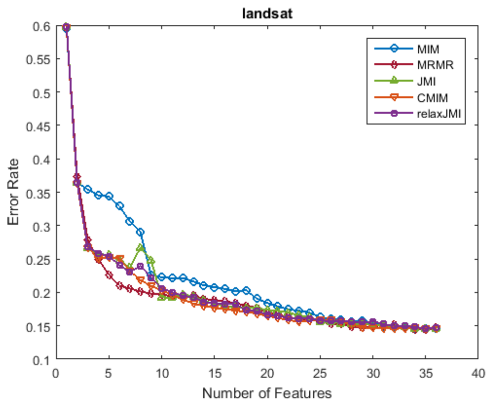
(c) Spect



(d) Ionosphere

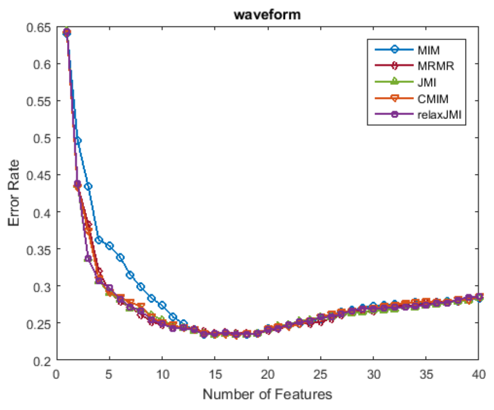


(e) Krvskp

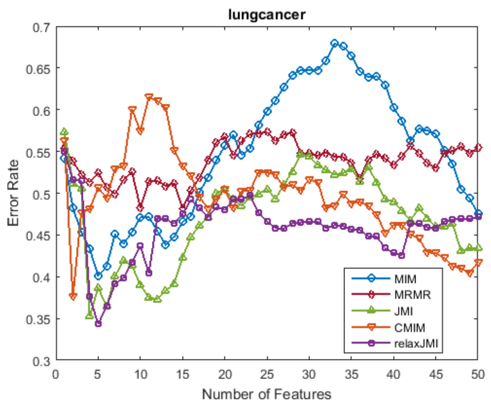


(f) Landsat

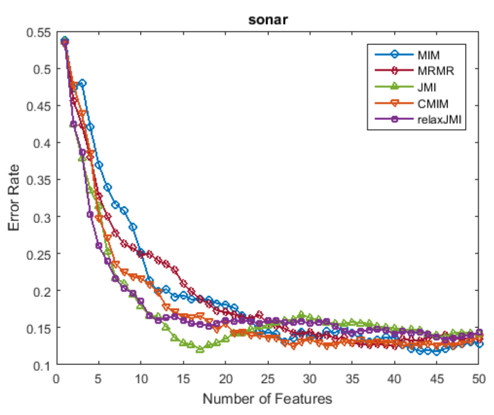
Figure 6.3: The classification error rate of the 1NN model using the subset of features selected by each feature selection method on the selected data sets. The horizontal axis represents the number of features selected by each method. The vertical axis represents the error rate when using the selected features to do the classification task.



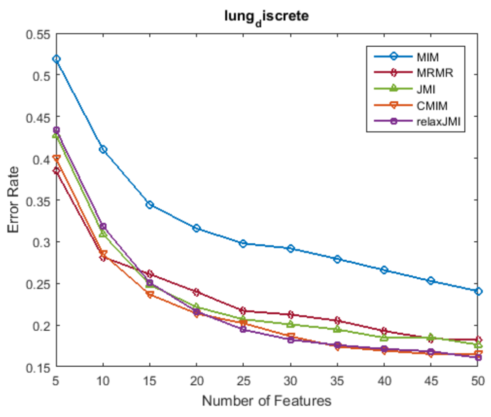
(a) Waveform



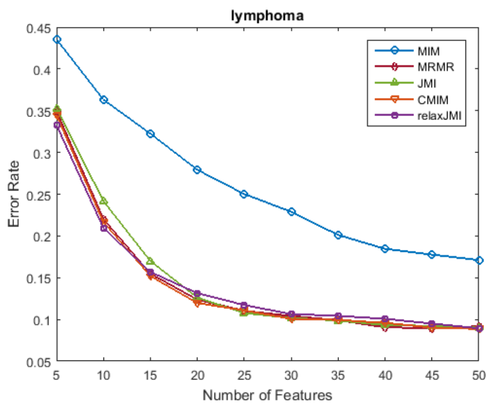
(b) Lungcancer



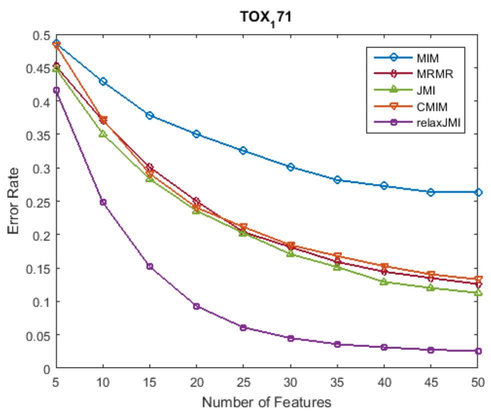
(c) Sonar



(d) Lungdiscrete



(e) lymphoma



(f) Tox171

Figure 6.4: The classification error rate of the 1NN model using the subset of features selected by each feature selection method on the selected data sets. The horizontal axis represents the number of features selected by each method. The vertical axis represents the error rate when using the selected features to do the classification task.

three features is estimated from the (class-conditional) MI between two features. Comprehensive numerical experimental results confirmed the effectiveness of the proposed method. When tested in the KNN algorithm, the proposed RelaxJMI method achieved statistically significantly higher classification accuracy than four other feature selection methods without sacrificing efficiency.

Chapter 7
Conclusion

IN THIS THESIS, I have investigated an important problem characteristic – the interaction between decision variables – in engineering design and decision-making problems. The overarching goals were to improve the accuracy and efficiency when identifying variable interactions in a given problem, and to use such information to guide the search for a promising candidate solution in the decision space. In this chapter, I summarize the findings of this thesis in Section 7.1; suggest possible directions for future work in Section 7.2; and present final remarks in Section 7.3.

7.1 Thesis Summary

In Chapter 1, I described the motivation for investigating variable interactions in engineering design and decision-making problems. The interaction between decision variables typically makes a given problem difficult to solve. By identifying variable interactions, I could possibly quantify the level of problem difficulty and design effective search algorithms.

Two research questions were presented in Chapter 1 to guide this thesis: $\mathfrak{RQ}1$ *How to accurately and efficiently identify variable interactions in a given problem?* and $\mathfrak{RQ}2$ *How can the variable interactions be used to effectively solve a given problem?* The two research questions were then explored in the *optimization* domain (Chapter 2, 3, 4 and 5) and the *classification* (feature selection) domain (Chapter 6). It is important to note that feature selection problems are highly relevant to optimization problems.

7.1.1 Optimization

In Chapter 2, I have reviewed the related work of variable interactions in the *optimization* domain. I first introduced fitness landscapes, and described the Exploratory Landscape Analysis (ELA) measures that can be used to explore fitness landscapes and quantify the level of variable interactions. Then the algorithms that guide the search process using variable interactions were reviewed, including Genetic Algorithms (GAs), model building algorithms as well as Cooperative Co-evolution (CC). Lastly, I suggested a taxonomy of the methods that can be used to identify variable interactions. The similarities and differences as well as the merits and drawbacks of the methods in each category were carefully discussed.

In Chapter 3, I have investigated the effects of interactions between decision variables in the Black-box Continuous Optimization Problems (BCOPs). Here, the goal was to examine whether it was possible to improve the performance of an optimization algorithm by guiding the search based on the level of variable interactions. I have described two alternative types of variable interactions – *direct interaction* and *indirect interaction* – that may appear in the decision variable space of BCOPs.

A robust ELA measure – *Maximum Entropic Epistasis* (MEE) – was introduced based on the Maximal Information Coefficient (MIC). The MIC can accurately identify a wide range of functional relationships with limited sample size. MEE identifies the *interaction matrix* of decision variables in a continuous optimization problem, which is then used to generate an accurate measure of the degree of variable interactions (encapsulating both direct and indirect variable interactions) spanning a suite of benchmark problems.

I have shown that the solution quality found by an EA is correlated with the level of variable interactions in a given problem. This observation suggests that fitness landscape characteristics captured by MEE can be used as a source of information to predict algorithm performance.

Significantly, when the MEE measure was embedded into an optimzation algorithm design framework, the performance of my model was at least as good, and in many cases outperformed, the Separability Prototype for Automatic Memes model on the suite of benchmark problems. These results confirmed my hypothesis that quantifying the level of variable interactions can be used to guide the search towards a better region of a fitness landscape.

In Chapter 4, I have investigated the effects of decision variable interactions and decomposition methods within the CC framework when solving large-scale BCOPs. I adopted the technique used in the Differential Grouping (DG) method to identify variable interactions based on a measure of non-linearity between decision variables. Importantly, I have shown that DG can not capture *indirect interaction* (described in Chapter 3), and this in turn can be used to explain relatively poor performance when decomposing some of the large-scale benchmark problems into components.

As a result, I have introduced an eXtended Differential Grouping (XDG) method to address this limitation. Results from comprehensive numerical simulation experiments clearly illustrated that XDG can achieve perfect decomposition on all of the benchmark functions investigated. When XDG was embedded within the Differential Evolution Cooperative Co-evolution (DECC) framework, it achieved significantly better performance than DECC-DG and DECC-D (with Delta Grouping) on the benchmark problems with indirect interaction. It also achieved comparable performance with DECC-DG on the benchmark problems without indirect interaction.

In Chapter 5, a Recursive Differential Grouping (RDG) method was proposed to improve the efficiency of problem decomposition. The existing decomposition methods (including XDG) typically need to check pairwise interactions between decision variables, resulting in use of $\mathcal{O}(n^2)$ Function Evaluations (FEs) when decomposing an n-dimensional large-scale BCOP into components.

The RDG method recursively examines the interaction between a selected decision variable and the remaining variables, placing all interacting decision variables into the same component. I have shown theoretically that the RDG method can decompose an n-dimensional problem using $\mathcal{O}\big(n\log(n)\big)$ FEs.

I evaluated the efficacy of the RDG method using large-scale benchmark BCOPs. Numerical simulation experiments showed that RDG greatly improved the efficiency of problem decomposition in terms of time complexity. Significantly, when RDG was embedded in the DECC / CMAESCC (uses CMAES as the component solver) framework, the optimization results were better than results from seven other decomposition methods when tested across a suite of benchmark large-scale BCOPs.

7.1.2 Classification

In the next stage of this thesis, I shifted my focus from the *optimization* to closely related *classification* domain, by investigating the interactions between features to improve classification accuracy. In Chapter 6, I have taken high-order interactions between features into consideration when selecting a subset of informative features for classification.

I first briefly described the existing assumptions made on the distribution of features and class labels, and summarized the well-known feature selection methods in the literature. Then I relaxed the assumptions made on the feature / class distribution and derived a novel feature evaluation criterion which considers the class-conditional Mutual Information (MI) between three features. To reduce computational cost, the three-way (class-conditional) MI was estimated from the two-way (class-conditional) MI between features.

The proposed feature selection method was evaluated on a suite of 16 real-world data sets using a widely used classification algorithm – K Nearest Neighbour. It achieved statistically significantly better classification accuracy when compared against four other feature selection methods in most cases.

7.2 Future Work

In this section, I present several possible directions for future work. Some of the future work is an extension or application of the proposed techniques described in this thesis. Some are inspired in the process of conducting this thesis, and may require extensive work.

This section is organized as follows. Section 7.2.1 presents a possible application of the MEE measure in the field of algorithm selection. Section 7.2.2 describes an avenue of designing other search techniques based on the MEE measure. Section 7.2.3 presents a possible extension of the XDG and RDG methods by considering the level of variable interactions. Section 7.2.4 describes an avenue of future work to investigate variable interactions in multi-objective optimization problems. Section 7.2.5 plans to develop a unified framework for feature selection based on MI.

7.2.1 Algorithm Selection Based on the MEE Measure

Selecting an appropriate algorithm to solve a given optimization problem has attracted much attention in recent years. It is a challenging task due to the large number of existing (candidate) algorithms and high complexity of problems. An algorithm selection framework typically trains a machine learning model based on the problem characteristics and the algorithm performances. When a new problem is given, the trained machine learning model can hopefully recommend an appropriate algorithm to solve it based on its characteristics. The success of the algorithm selection model highly relies on the characteristics extracted from the problems. In recent years, a large number of ELA measures have been proposed to identify certain problem characteristics (see Table 2.2 for some representative measures). The MEE described in Chapter 3 can also be used as an ELA measure.

In future work, I plan to develop an algorithm selection model, where a range of ELA measures are combined with MEE. In [166], I have proposed two criteria to follow when selecting ELA measures to build the algorithm selection model: (1) the selected ELA measures can capture all the important problem characteristics (e.g., separability, modality, basins of attraction, ruggedness, smoothness, neutrality) related to problem difficulty; and (2) the computational cost of the selected ELA measures is minimal. I have attempted to map ELA measures used to capture the problem characteristics. However, other ELA measures such as Fitness Distance Correlation and Dispersion Metric should be examined to paint a more coherent and complete picture.

7.2.2 Search Techniques Based on the MEE Measure

In Chapter 3, I presented an algorithm design framework where the MEE measure was used to guide the selection of appropriate search operators (S or R) to solve a given problem. When the given problem has low (high) level of variable interactions, S (R) operator is more likely to be selected to solve the problem. The S operator searches along each decision variable, which is efficient to solve separable problems. The R operator perturbs all the variables simultaneously to follow the gradient of the fitness landscape. It is important to note that this is just one possible application of the MEE measure when designing search techniques. In future work, I plan to complete further analysis of the MEE measure and explore additional avenues to guide the search using the degree of variable interactions.

7.2.3 Problem Decomposition Based on the Degree of Variable Interactions

The application of the existing automatic decomposition methods (including XDG and RDG) is limited to partially separable problems. The solution quality found by a CC algorithm can be greatly improved by fully considering the underlying interaction structure between decision variables when decomposing the problem. However some of the real-world design and decision-making problems are fully non-separable – all the decision variables directly or indirectly interact with each other. The automatic decomposition methods are ineffective when used to decompose such problems, as all the decision variables will be assigned to one group.

In [167], I have presented some preliminary results indicating that Random Grouping outperformed some sophisticated decomposition methods when tested across a suite of benchmark large-scale fully non-separable problems. A possible improvement of the existing automatic decomposition methods is to take the level of variable interactions into consideration. The level of variable interactions may vary significantly in a fully non-separable problem. Therefore it is possible to assign strongly interacting decision variables into the same component while keeping weakly interacting ones separate when decomposing a large-scale BCOP.

7.2.4 Variable Interaction in Multi-objective Problems

This thesis focuses on the interaction between decision variables in single-objective problems. A possible extension of this work is to explore variable interactions in multi-objective problems. There have been some initial results shown that identifying variable interaction structures for some well-known multi-objective benchmark problems is non-trivial [72]. However more theoretical and empirical work is required in this relatively new and interesting field.

Another direction worth pursuing is to apply the CC framework to solve large-scale multi-objective optimization problems. It is challenging to divide a multi-objective problem into several components, as the interaction structure of the decision variables in each objective function may be different. In future work, I plan to design an effective method based on variable interactions to decompose multi-objective problems.

Exploring the interaction between objectives is another interesting topic in many-objective optimization problems. Such problems are difficult to solve as many candidate solutions in each generation tend to be non-dominated, resulting in a low selection pressure. There has been some work investigating the interaction between objectives, based on which the non-redundant objectives are selected to evaluate candidate solutions. The MIC used in Chapter 3 can capture a wide range of function relationships. In future work, I plan to apply MIC for the objective reduction task.

7.2.5 Unified Feature Selection based on Information Theory

The existing information theoretic feature selection methods typically require normalization, as described in Chapter 6. This may indicate that the assumptions made on the underlying distribution of the features and class labels are not appropriate. In future work, I plan to further investigate the assumptions made on the data distribution and derive the normalized feature selection methods directly.

Most of the existing feature selection methods focus on the low-order interaction between features: typically the MI between two or three features. It is not clear yet which criterion should be used when considering interactions between four or more features, and which assumptions should be made in order to derive such criterion. These questions are worth pursuing in the future.

A large number of feature selection methods based on MI has been proposed in recent years. Is there any relation between these methods? In future, it is worth investigating this question and provide a generic feature selection framework based on MI. The generic framework should include the existing low-order feature selection methods and can provide new selection criterion when considering high-order feature interactions.

7.3 Final Remarks

In this thesis, I have explored the interaction between decision variables in the *optimization* and *classification* domains. Novel methods have been proposed which can be used to efficiently and accurately identify variable interactions in a given problem. I have shown that the identified variable interactions can provide insights into problem difficulty, and can guide the design of effective search and optimization techniques.

I recognize that the scope of this thesis is limited to the continuous optimization and to a limited extent the feature selection domains, and the techniques presented in this thesis can be further improved in terms of efficiency, accuracy and generality (see Section 7.2 for some possible extensions). However I acknowledge that it should be the community effort. By presenting this thesis, I hope more attention will be drawn to the investigation of variable interactions in the future.

Bibliography

[1] Khalid Abdulla, Julian De Hoog, Valentin Muenzel, Frank Suits, Kent Steer, Andrew Wirth, and Saman Halgamuge. Optimal operation of energy storage systems considering forecasts and battery degradation. *IEEE Transactions on Smart Grid*, In Press.

[2] Davide Albanese, Michele Filosi, Roberto Visintainer, Samantha Riccadonna, Giuseppe Jurman, and Cesare Furlanello. minerva and minepy: a c engine for the mine suite and its r, python and matlab wrappers. *Bioinformatics*, 29(3):407–408, 2013.

[3] Mostafa Z Ali, Noor H Awad, and Ponnuthurai N Suganthan. Multi-population differential evolution with balanced ensemble of mutation strategies for large-scale global optimization. *Applied Soft Computing*, 33:304–327, 2015.

[4] Lee Altenberg. NK fitness landscapes. In *The Handbook of Evolutionary Computation*, 1997.

[5] Anne Auger and Nikolaus Hansen. A restart CMA evolution strategy with increasing population size. In *Evolutionary Computation, 2005 IEEE Congress on*, volume 2, pages 1769–1776. IEEE, 2005.

[6] Anne Auger, Marc Schoenauer, and Nicolas Vanhaecke. LS-CMA-ES: A second-order algorithm for covariance matrix adaptation. In *International Conference on Parallel Problem Solving from Nature*, pages 182–191. Springer, 2004.

[7] Shumeet Baluja. Population-based incremental learning. A method for integrating genetic search based function optimization and competitive learning. Technical report, Carnegie-Mellon Univ Pittsburgh Pa Dept Of Computer Science, 1994.

[8] Shumeet Baluja and Scott Davies. Using optimal dependency-trees for combinatorial optimization: Learning the structure of the search space. Technical Report CMU-CS-97-107, January 1997.

[9] William Bateson. Mendel's principles of heredity. Cambridge University Press, 1909.

[10] Roberto Battiti. Using mutual information for selecting features in supervised neural net learning. *IEEE Transactions on Neural Networks*, 5(4):537–550, 1994.

[11] Peter Benner. Solving large-scale control problems. *Control Systems, IEEE*, 24(1):44–59, 2004.

[12] Verónica Bolón-Canedo, Noelia Sánchez-Marono, Amparo Alonso-Betanzos, José Manuel Benítez, and Francisco Herrera. A review of microarray datasets and applied feature selection methods. *Information Sciences*, 282:111–135, 2014.

[13] Terry Bossomaier, Lionel Barnett, and Michael Harré. Information and phase transitions in socio-economic systems. *Complex Adaptive Systems Modeling*, 1(1):9–34, 2013.

[14] Janez Brest and Mirjam Sepesy Maučec. Self-adaptive differential evolution algorithm using population size reduction and three strategies. *Soft Computing*, 15(11):2157–2174, 2011.

[15] Gavin Brown, Adam Pocock, Ming-Jie Zhao, and Mikel Luján. Conditional likelihood maximisation: a unifying framework for information theoretic feature selection. *Journal of Machine Learning Research*, 13(Jan):27–66, 2012.

[16] Darren J Burgess. Genetic screens: A global map of genetic interactions. *Nature Reviews Genetics*, 17(11):659–659, 2016.

[17] Pilar CaamañO, Francisco Bellas, Jose A Becerra, and Richard J Duro. Evolutionary algorithm characterization in real parameter optimization problems. *Applied Soft Computing*, 13(4):1902–1921, 2013.

[18] Fabio Caraffini, Ferrante Neri, and Lorenzo Picinali. An analysis on separability for memetic computing automatic design. *Information Sciences*, 265:1–22, 2014.

[19] Wenxiang Chen and Ke Tang. Impact of problem decomposition on cooperative coevolution. In *Evolutionary Computation, 2013 IEEE Congress on*, pages 733–740. IEEE, 2013.

[20] Wenxiang Chen, Thomas Weise, Zhenyu Yang, and Ke Tang. Large-scale global optimization using cooperative coevolution with variable interaction learning. In *Parallel Problem Solving from Nature, PPSN XI*, pages 300–309. Springer, 2010.

[21] YP Chen, Tian-Li Yu, Kumara Sastry, and David E Goldberg. A survey of linkage learning techniques in genetic and evolutionary algorithms. *IlliGAL report*, 2007014, 2007.

[22] Ran Cheng and Yaochu Jin. A competitive swarm optimizer for large scale optimization. *Cybernetics, IEEE Transactions on*, 45(2):191–204, 2015.

[23] C Chow and Cong Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14(3):462–467, 1968.

[24] Heather J Cordell. Detecting gene–gene interactions that underlie human diseases. *Nature Reviews Genetics*, 10(6):392–404, 2009.

[25] Yuval Davidor. Epistasis variance: Suitability of a representation to genetic algorithms. *Complex Systems*, 4(4):369–383, 1990.

[26] Yuval Davidor. Epistasis variance: A viewpoint on GA-hardness. *Foundations of Genetic Algorithms*, 1:23–35, 1991.

[27] Jeremy S De Bonet, Charles Lee Isbell Jr, and Paul A Viola. MIMIC: Finding optima by estimating probability densities. In *Advances in Neural Information Processing Systems*, pages 424–430, 1997.

[28] Ramón Díaz-Uriarte and Sara Alvarez De Andres. Gene selection and classification of microarray data using random forest. *BMC Bioinformatics*, 7(1):3, 2006.

[29] Weishan Dong, Tianshi Chen, Peter Tino, and Xin Yao. Scaling up estimation of distribution algorithms for continuous optimization. *IEEE Transactions on Evolutionary Computation*, 17(6):797–822, 2013.

[30] Usama Fayyad and Keki Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of International Joint Conference on Artificial Intelligence*, pages 1022–1029, 1993.

[31] François Fleuret. Fast binary feature selection with conditional mutual information. *Journal of Machine Learning Research*, 5(Nov):1531–1555, 2004.

[32] Cyril Fonlupt, Denis Robilliard, and Philippe Preux. A bit-wise epistasis measure for binary search spaces. In *Parallel Problem Solving from NaturePPSN V*, pages 47–56. Springer, 1998.

[33] S Forrest and M Mitchell. Relative building-block fitness and the building block hypothesis. *Foundations of Genetic Algorithms*, 2:109–126, 1993.

[34] Marcus Gallagher, Marcus Frean, and Tom Downs. Real-valued evolutionary optimization using a flexible probability density estimator. In *Proceedings of the first Annual Conference on Genetic and Evolutionary Computation*, pages 840–846. Morgan Kaufmann Publishers Inc., 1999.

[35] Hongwei Ge, Liang Sun, Xin Yang, Shinichi Yoshida, and Yanchun Liang. Cooperative differential evolution with fast variable interdependence learning and cross-cluster mutation. *Applied Soft Computing*, 36:300–314, 2015.

[36] ChiKeong Goh and Kay Chen Tan. A competitive-cooperative coevolutionary paradigm for dynamic multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 13(1):103–127, 2009.

[37] D E Goldberg, K Deb, H Kargupta, and G Harik. Rapid accurate optimization of difficult problems using fast messy genetic algorithms. *Proceedings of the Fifth International Conference on Genetic Algorithm*, pages 56–64, 1993.

[38] David Goldberg, K. Deb, and B. Korb. Messy genetic algorithms: Motivation, analysis, and first results. *Complex Systems*, (3):493–530, 1989.

[39] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1989.

[40] David E Goldberg, DEB Kalyanmoy, and Dirk Thierens. Toward a better understanding of mixing in genetic algorithms. *Journal of the Society of Instrument and Control Engineers*, 32(1):10–16, 1993.

[41] David E Goldberg, Robert Lingle, et al. Alleles, loci, and the traveling salesman problem. In *Proceedings of an International Conference on Genetic Algorithms and Their Applications*, volume 154, pages 154–159. Lawrence Erlbaum, Hillsdale, NJ, 1985.

[42] Samuel B Green. How many subjects does it take to do a regression analysis. *Multivariate behavioral research*, 26(3):499–510, 1991.

[43] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3(Mar):1157–1182, 2003.

[44] Pathima Nusrath Hameed, Karin Verspoor, Snezana Kusljic, and Saman Halgamuge. Positive-unlabeled learning for inferring drug interactions based on heterogeneous attributes. *BMC Bioinformatics*, 18(1):140, 2017.

[45] Nikolaus Hansen, Sibylle D Müller, and Petros Koumoutsakos. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). *Evolutionary Computation*, 11(1):1–18, 2003.

[46] Nikolaus Hansen and Andreas Ostermeier. Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation. In *Evolutionary Computation, 1996 IEEE Congress on*, pages 312–317. IEEE, 1996.

[47] Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary computation*, 9(2):159–195, 2001.

[48] Nikolaus Hansen, Raymond Ros, Nikolas Mauny, Marc Schoenauer, and Anne Auger. Impacts of invariance in search: When CMA-ES and PSO face ill-conditioned and non-separable problems. *Applied Soft Computing*, 11(8):5755–5769, 2011.

[49] Georges R Harik and David E Goldberg. Learning linkage. In *FOGA*, volume 4, pages 247–262, 1996.

[50] Georges R Harik, Fernando G Lobo, and David E Goldberg. The compact genetic algorithm. *IEEE Transactions on Evolutionary Computation*, 3(4):287–297, 1999.

[51] Georges Raif Harik. Learning gene linkage to efficiently solve problems of bounded difficulty using genetic algorithms. In *Doctoral dissertation, The University of Michigan*, 1997.

[52] Mark Hauschild and Martin Pelikan. An introduction and survey of estimation of distribution algorithms. *Swarm and Evolutionary Computation*, 1(3):111–128, 2011.

[53] Jun He, Tianshi Chen, and Xin Yao. On the easiest and hardest fitness functions. *IEEE Transactions on Evolutionary Computation*, 19(2):295–305, 2015.

[54] Jun He, Colin Reeves, Carsten Witt, and Xin Yao. A note on problem difficulty measures in black-box optimization: Classification, realizations and predictability. *Evolutionary Computation*, 15(4):435–443, 2007.

[55] Gunawan Herman, Bang Zhang, Yang Wang, Getian Ye, and Fang Chen. Mutual information-based method for selecting informative feature sets. *Pattern Recognition*, 46(12):3315–3327, 2013.

[56] John H. Holland. *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control and artificial intelligence*. Ann Arbor, University of Michigan Press, 1975.

[57] Xiao-Min Hu, Fei-Long He, Wei-Neng Chen, and Jun Zhang. Cooperation coevolution with fast interdependency identification for large scale optimization. *Information Sciences*, 381:142–160, 2017.

[58] Walters Williams Janett and Yan Li. Estimation of mutual information: A survey. In *Rough Sets and Knowledge Technology*, pages 389–396. Springer Berlin Heidelberg, 2009.

[59] Thanyaluk Jirapech-Umpai and Stuart Aitken. Feature selection and classification for microarray data analysis: Evolutionary methods for identifying predictive genes. *BMC Bioinformatics*, 6(1):148, 2005.

[60] Terry Jones and Stephanie Forrest. Fitness distance correlation as a measure of problem didculty for genetic algorithms. *Proceedings of the 6th International Conference on Genetic Algorithms*, 95:184–192, 1995.

[61] Ata Kabán, Jakramate Bootkrajang, and Robert John Durrant. Toward Large-Scale Continuous EDA: A Random Matrix Theory Perspective. *Evolutionary Computation*, 2015.

[62] Leila Kallel, Bart Naudts, and Colin R Reeves. Properties of fitness functions and search landscapes. In *Theoretical Aspects of Evolutionary Computing*, pages 175–206. Springer, 2001.

[63] Hillol Kargupta. The performance of the gene expression messy genetic algorithm on real test functions. In *Evolutionary Computation, 1996 IEEE Congress on*, pages 631–636. IEEE, 1996.

[64] Stuart A Kauffman and Edward D Weinberger. The NK model of rugged fitness landscapes and its application to maturation of the immune response. *Journal of Theoretical Biology*, 141(2):211–245, 1989.

[65] Borhan Kazimipour, Mohammad Nabi Omidvar, Xiaodong Li, and AK Qin. A sensitivity analysis of contribution-based cooperative co-evolutionary algorithms. In *Evolutionary Computation, 2015 IEEE Congress on*, pages 417–424. IEEE, 2015.

[66] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Neural Networks, IEEE International Conference on*, pages 1942–1948, 1995.

[67] Josef Kittler. Feature selection and extraction. *Handbook of Pattern Recognition and Image Processing*, pages 59–83, 1986.

[68] Miguel Lastra, Daniel Molina, and José M Benítez. A high performance memetic algorithm for extremely high-dimensional problems. *Information Sciences*, 293:35–58, 2015.

[69] Antonio LaTorre, Santiago Muelas, and José-María Peña. A MOS-based dynamic memetic differential evolution algorithm for continuous optimization: a scalability test. *Soft Computing*, 15(11):2187–2199, 2011.

[70] David D Lewis. Feature selection and feature extraction for text categorization. In *Proceedings of the Workshop on Speech and Natural Language*, pages 212–217. Association for Computational Linguistics, 1992.

[71] Jundong Li, Kewei Cheng, Suhang Wang, Fred Morstatter, Robert P Trevino, Jiliang Tang, and Huan Liu. Feature selection: A data perspective. *arXiv preprint arXiv:1601.07996*, 2016.

[72] Ke Li, Mohammad Nabi Omidvar, Kalyanmoy Deb, and Xin Yao. Variable interaction in multi-objective optimization problems. In *International Conference on Parallel Problem Solving from Nature*, pages 399–409. Springer, 2016.

[73] Xiaodong Li, Ke Tang, Mohammad N Omidvar, Zhenyu Yang, and Kai Qin. Benchmark functions for the CEC 2013 special session and competition on large-scale global optimization. *Gene*, 7(33):8, 2013.

[74] Xiaodong Li and Xin Yao. Cooperatively coevolving particle swarms for large scale optimization. *IEEE Transactions on Evolutionary Computation*, 16(2):210–224, 2012.

[75] M. Lichman. UCI machine learning repository. *University of California, Irvine, School of Information and Computer Sciences,* http://archive.ics.uci.edu/ml, 2013.

[76] Dahua Lin and Xiaoou Tang. Conditional infomax learning: an integrated framework for feature extraction and fusion. *Computer Vision–ECCV 2006*, pages 68–82, 2006.

[77] Yingbiao Ling, Haijian Li, and Bin Cao. Cooperative co-evolution with graph-based differential grouping for large scale global optimization. In *Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD), 2016 12th International Conference on*, pages 95–102. IEEE, 2016.

[78] Haiyan Liu, Yuping Wang, Xuyan Liu, and Shiwei Guan. Empirical study of effect of grouping strategies for large scale optimization. In *Neural Networks (IJCNN), 2016 International Joint Conference on*, pages 3433–3439. IEEE, 2016.

[79] Huan Liu and Rudy Setiono. Chi2: Feature selection and discretization of numeric attributes. In *Tools with Artificial Intelligence, 1995. Proceedings., Seventh International Conference on*, pages 388–391. IEEE, 1995.

[80] Huan Liu and Lei Yu. Toward integrating feature selection algorithms for classification and clustering. *Knowledge and Data Engineering, IEEE Transactions on*, 17(4):491–502, 2005.

[81] Jinpeng Liu and Ke Tang. Scaling up covariance matrix adaptation evolution strategy using cooperative coevolution. In *International Conference on Intelligent Data Engineering and Automated Learning*, pages 350–357. Springer, 2013.

[82] Monte Lunacek and Darrell Whitley. The dispersion metric and the CMA evolution strategy. In *Proceedings of the 2006 Annual Conference on Genetic and Evolutionary Computation*, pages 477–484. ACM, 2006.

[83] Sedigheh Mahdavi, Shahryar Rahnamayan, and Mohammad Ebrahim Shiri. Multilevel framework for large-scale global optimization. *Soft Computing*, pages 1–30, 2016.

[84] Sedigheh Mahdavi, Mohammad Ebrahim Shiri, and Shahryar Rahnamayan. Cooperative co-evolution with a new decomposition method for large-scale optimiza-

tion. In *Evolutionary Computation, 2014 IEEE Congress on*, pages 1285–1292. IEEE, 2014.

[85] Sedigheh Mahdavi, Mohammad Ebrahim Shiri, and Shahryar Rahnamayan. Metaheuristics in large-scale global continues optimization: A survey. *Information Sciences*, 295:407–428, 2015.

[86] Katherine M Malan and Andries P Engelbrecht. A survey of techniques for characterising fitness landscapes and some possible ways forward. *Information Sciences*, 241:148–163, 2013.

[87] M Manela and J A Campbell. Harmonic analysis, epistasis and genetic algorithms. In *Proceedings of the Second Conference on Parallel Problem Solving from Nature*, pages 57–64. Elsevier, 1992.

[88] Narine Manukyan, Margaret J Eppstein, and Jeffrey S Buzas. Tunably rugged landscapes with known maximum and minimum. *IEEE Transactions on Evolutionary Computation*, 20(2):263–274, 2016.

[89] Jesús Marín. How landscape ruggedness influences the performance of real-coded algorithms: a comparative study. *Soft Computing*, 16(4):683–698, 2012.

[90] Joaquim RRA Martins and Andrew B Lambe. Multidisciplinary design optimization: a survey of architectures. *AIAA journal*, 51(9):2049–2075, 2013.

[91] David M McCandlish. Visualizing fitness landscapes. *Evolution*, 65(6):1544–1558, 2011.

[92] Yi Mei, Xiaodong Li, and Xin Yao. Cooperative coevolution with route distance grouping for large-scale capacitated arc routing problems. *IEEE Transactions on Evolutionary Computation*, 18(3):435–449, 2014.

[93] Yi Mei, Mohammad Nabi Omidvar, Xiaodong Li, and Xin Yao. A competitive divide-and-conquer algorithm for unconstrained large-scale black-box optimization. *ACM Transactions on Mathematical Software (TOMS)*, 42(2):13, 2016.

[94] Olaf Mersmann, Bernd Bischl, Heike Trautmann, Mike Preuss, Claus Weihs, and Günter Rudolph. Exploratory landscape analysis. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pages 829–836. ACM, 2011.

[95] Olaf Mersmann, Mike Preuss, and Heike Trautmann. Benchmarking evolutionary algorithms: Towards exploratory landscape analysis. In *International Conference on Parallel Problem Solving from Nature*, pages 73–82. Springer, 2010.

[96] Olaf Mersmann, Mike Preuss, Heike Trautmann, Bernd Bischl, and Claus Weihs. Analyzing the bbob results by means of benchmarking concepts. *Evolutionary Computation*, 23(1):161–185, 2015.

[97] Patrick Emmanuel Meyer, Colas Schretter, and Gianluca Bontempi. Information-theoretic feature selection in microarray data using variable complementarity. *IEEE Journal of Selected Topics in Signal Processing*, 2(3):261–274, 2008.

[98] Melanie Mitchell. *An introduction to genetic algorithms*. MIT press, 1998.

[99] Daniel Molina, Manuel Lozano, and Francisco Herrera. MA-SW-Chains: Memetic algorithm based on local search chains for large scale continuous global optimization. In *Evolutionary Computation, 2010 IEEE Congress on*, pages 1–8. IEEE, 2010.

[100] Daniel Molina, Manuel Lozano, Ana M Sánchez, and Francisco Herrera. Memetic algorithms based on local search chains for large scale continuous optimisation problems: MA-SSW-Chains. *Soft Computing*, 15(11):2201–2220, 2011.

[101] Jason H Moore. A global view of epistasis. *Nature Genetics*, 37(1):13–14, 2005.

[102] Boris S Mordukhovich, Nguyen Mau Nam, and Nguyen Thi Yen Nhi. Partial second-order subdifferentials in variational analysis and optimization. *Numerical Functional Analysis and Optimization*, 35(7-9):1113–1151, 2014.

[103] Rachael Morgan and Marcus Gallagher. Length scale for characterising continuous optimization problems. *Parallel Problem Solving from Nature-PPSN XII*, pages 407–416, 2012.

[104] Heinz Mühlenbein. The equation for response to selection and its use for prediction. *Evolutionary Computation*, 5(3):303–346, 1997.

[105] Heinz Mühlenbein, Jürgen Bendisch, and H Voigt. From recombination of genes to the estimation of distributions II. Continuous parameters. *Parallel Problem Solving from NaturePPSN IV*, pages 188–197, 1996.

[106] Heinz Mühlenbein, Thilo Mahnig, and Alberto Ochoa Rodriguez. Schemata, distributions and graphical models in evolutionary optimization. *Journal of Heuristics*, 5(2):215–247, 1999.

[107] Heinz Mühlenbein and Gerhard Paass. From recombination of genes to the estimation of distributions I. Binary parameters. *Parallel Problem Solving from NaturePPSN IV*, pages 178–187, 1996.

[108] Masaharu Munetomo. Linkage identification based on epistasis measures to realize efficient genetic algorithms. In *Evolutionary Computation, 2002 IEEE Congress on*, volume 2, pages 1332–1337. IEEE, 2002.

[109] Masaharu Munetomo and David E Goldberg. Designing a genetic algorithm using the linkage identification by nonlinearity check. *IlliGAL Report*, (98014), 1998.

[110] Masaharu Munetomo and David E Goldberg. Identifying linkage by nonlinearity check. *IlliGAL Report*, 98012, 1998.

[111] Masaharu Munetomo and David E Goldberg. A genetic algorithm using linkage identification by nonlinearity check. In *Systems, Man, and Cybernetics, 1999. IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on*, volume 1, pages 595–600. IEEE, 1999.

[112] Masaharu Munetomo and David E Goldberg. Identifying linkage groups by nonlinearity/non-monotonicity detection. In *Proceedings of the genetic and evolutionary computation conference*, volume 1, pages 433–440, 1999.

[113] Masaharu Munetomo and David E Goldberg. Linkage identification by non-monotonicity detection for overlapping functions. *Evolutionary Computation*, 7(4):377–398, 1999.

[114] Mario A Munoz, Michael Kirley, and Saman K Halgamuge. The algorithm selection problem on the continuous optimization domain. In *Computational Intelligence in Intelligent Data Analysis*, pages 75–89. Springer, 2013.

[115] Mario A Muñoz, Michael Kirley, and Saman K Halgamuge. Exploratory landscape analysis of continuous space optimization problems using information content. *IEEE Transactions on Evolutionary Computation*, 19(1):74–87, 2015.

[116] Mario A Muñoz, Yuan Sun, Michael Kirley, and Saman K Halgamuge. Algorithm selection for black-box continuous optimization problems: A survey on methods and challenges. *Information Sciences*, 317:224–245, 2015.

[117] Bart Naudts and Leila Kallel. A comparison of predictive measures of problem difficulty in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 4(1):1–15, 2000.

[118] Xuan Vinh Nguyen, Jeffrey Chan, Simone Romano, and James Bailey. Effective global approaches for mutual information based feature selection. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 512–521. ACM, 2014.

[119] Olusegun Olorunda and Andries Petrus Engelbrecht. An analysis of heterogeneous cooperative algorithms. In *Evolutionary Computation, 2009 IEEE Congress on*, pages 1562–1569. IEEE, 2009.

[120] Mohammad Nabi Omidvar, Xiaodong Li, Yi Mei, and Xin Yao. Cooperative co-evolution with differential grouping for large scale optimization. *IEEE Transactions on Evolutionary Computation*, 18(3):378–393, 2014.

[121] Mohammad Nabi Omidvar, Xiaodong Li, and Ke Tang. Designing benchmark problems for large-scale continuous optimization. *Information Sciences*, 316:419–436, 2015.

[122] Mohammad Nabi Omidvar, Xiaodong Li, Zhenyu Yang, and Xin Yao. Cooperative co-evolution for large scale optimization through more frequent random grouping. In *Evolutionary Computation, 2010 IEEE Congress on*, pages 1–8. IEEE, 2010.

[123] Mohammad Nabi Omidvar, Xiaodong Li, and Xin Yao. Cooperative co-evolution with delta grouping for large scale non-separable function optimization. In *Evolutionary Computation, 2010 IEEE Congress on*, pages 1–8. IEEE, 2010.

[124] Mohammad Nabi Omidvar, Xiaodong Li, and Xin Yao. Smart use of computational resources based on contribution for cooperative co-evolutionary algorithms. In *Proceedings of the 2011 Annual Conference on Genetic and Evolutionary Computation*, pages 1115–1122. ACM, 2011.

[125] Mohammad Nabi Omidvar, Yi Mei, and Xiaodong Li. Effective decomposition of large-scale separable continuous functions for cooperative co-evolutionary algorithms. In *Evolutionary Computation, 2014 IEEE Congress on*, pages 1305–1312. IEEE, 2014.

[126] Mohammad Nabi Omidvar, Ming Yang, Yi Mei, Xiaodong Li, and Xin Yao. DG2: A faster and more accurate differential grouping for large-scale black-box optimization. *IEEE Transactions on Evolutionary Computation*, 21(6):929–942, 2017.

[127] Martin Pelikan, David E Goldberg, and Erick Cantú-Paz. BOA: The Bayesian optimization algorithm. In *Proceedings of the first Annual Conference on Genetic and Evolutionary Computation*, pages 525–532. Morgan Kaufmann Publishers Inc., 1999.

[128] Martin Pelikan and Heinz Mühlenbein. The bivariate marginal distribution algorithm. In *Advances in Soft Computing*, pages 521–535. Springer, 1999.

[129] Hanchuan Peng, Fuhui Long, and Chris Ding. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8):1226–1238, 2005.

[130] Patrick C Phillips. Epistasisthe essential role of gene interactions in the structure and evolution of genetic systems. *Nature Reviews. Genetics*, 9(11):855, 2008.

[131] Erik Pitzer and Michael Affenzeller. A comprehensive survey on fitness landscape analysis. *Recent Advances in Intelligent Engineering Systems*, 378:161–191, 2012.

[132] Frank J Poelwijk, Daniel J Kiviet, Daniel M Weinreich, and Sander J Tans. Empirical fitness landscapes reveal accessible evolutionary paths. *Nature*, 445(7126):383–386, 2007.

[133] Mitchell A Potter and Kenneth A De Jong. A cooperative coevolutionary approach to function optimization. In *Parallel problem solving from nature PPSN III*, pages 249–257. Springer, 1994.

[134] Upeka Premaratne, Saman K Halgamuge, and Iven MY Mareels. Event triggered adaptive differential modulation: a new method for traffic reduction in networked control systems. *IEEE Transactions on Automatic Control*, 58(7):1696–1706, 2013.

[135] Pavel Pudil, Jana Novovičová, and Josef Kittler. Floating search methods in feature selection. *Pattern Recognition Letters*, 15(11):1119–1125, 1994.

[136] R. Cheng and Y. Jin. A social learning particle swarm optimization algorithm for scalable optimization. *Information Sciences*, 291:43–60, 2015.

[137] Tapabrata Ray and Xin Yao. A cooperative coevolutionary algorithm with correlation based adaptive variable partitioning. In *Evolutionary Computation, 2009 IEEE Congress on*, pages 983–989. IEEE, 2009.

[138] Colin Reeves and Christine Wright. An experimental design perspective on genetic algorithms. *Foundations of Genetic Algorithms*, 3:7–22, 1995.

[139] Colin R Reeves. Predictive measures for problem difficulty. In *Evolutionary Computation, 1999 IEEE Congress on*, volume 1, pages 736–743. IEEE, 1999.

[140] Yuanfang Ren and Yan Wu. An efficient algorithm for high-dimensional function optimization. *Soft Computing*, 17(6):995–1004, 2013.

[141] D. N. Reshef, Y. A. Reshef, H. K. Finucane, and et al. Detecting novel associations in large data sets. *Science*, 334(6062):1518–1524, 2011.

[142] John R Rice. The algorithm selection problem. *Advances in Computers*, 15:65–118, 1976.

[143] Marko Robnik-Šikonja and Igor Kononenko. Theoretical and empirical analysis of relieff and rrelieff. *Machine Learning*, 53(1-2):23–69, 2003.

[144] Sophie Rochet. Epistasis in genetic algorithms revisited. *Information Sciences*, 102(1-4):133–155, 1997.

[145] Sophie Rochet, Gilles Venturini, Mohamed Slimane, and E El Kharoubi. A critical and empirical study of epistasis measures for predicting ga performances: a summary. In *Artificial Evolution*, pages 275–285. Springer, 1998.

[146] R Tyrrell Rockafellar and Roger J-B Wets. *Variational analysis*, volume 317. Springer Science & Business Media, 2009.

[147] Irene Rodriguez-Lujan, Ramon Huerta, Charles Elkan, and Carlos Santa Cruz. Quadratic programming feature selection. *Journal of Machine Learning Research*, 11(Apr):1491–1516, 2010.

[148] Helge Rosé, Werner Ebeling, and Torsten Asselmeyer. The density of statesa measure of the difficulty of optimisation problems. *Parallel Problem Solving from NaturePPSN IV*, pages 208–217, 1996.

[149] HoHo Rosenbrock. An automatic method for finding the greatest or least value of a function. *The Computer Journal*, 3(3):175–184, 1960.

[150] Yvan Saeys, Iñaki Inza, and Pedro Larrañaga. A review of feature selection techniques in bioinformatics. *Bioinformatics*, 23(19):2507–2517, 2007.

[151] Ralf Salomon. Re-evaluating genetic algorithm performance under coordinate rotation of benchmark functions. A survey of some theoretical and practical aspects of genetic algorithms. *BioSystems*, 39(3):263–278, 1996.

[152] Eman Sayed, Daryl Essam, Ruhul Sarker, and Saber Elsayed. Decomposition-based evolutionary algorithm for large scale constrained problems. *Information Sciences*, 316:457–486, 2015.

[153] Michèle Sebag and Antoine Ducoulombier. Extending population-based incremental learning to continuous search spaces. In *International Conference on Parallel Problem Solving from Nature*, pages 418–427. Springer, 1998.

[154] Dong-Il Seo and Byung-Ro Moon. An information-theoretic analysis on the interactions of variables in combinatorial optimization problems. *Evolutionary Computation*, 15(2):169–198, 2007.

[155] Songqing Shan and G Gary Wang. Metamodeling for high dimensional simulation-based design problems. *Journal of Mechanical Design*, 132(5):051009, 2010.

[156] Songqing Shan and G Gary Wang. Survey of modeling and optimization strategies to solve high-dimensional design problems with computationally-expensive black-box functions. *Structural and Multidisciplinary Optimization*, 41(2):219–241, 2010.

[157] Claude E Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423, 1948.

[158] David J Sheskin. *Handbook of parametric and nonparametric statistical procedures*. CRC Press, 2003.

[159] Yan-jun Shi, Hong-fei Teng, and Zi-qiang Li. Cooperative co-evolutionary differential evolution for function optimization. *Advances in Natural Computation*, pages 428–428, 2005.

[160] Hemant Kumar Singh and Tapabrata Ray. Divide and conquer in coevolution: A difficult balancing act. In *Agent-Based Evolutionary Search*, pages 117–138. Springer, 2010.

[161] T. Speed. A correlation for the 21st century. *Science*, 334:1502–1503, 2011.

[162] Peter F Stadler. Fitness landscapes. *Biological Evolution and Statistical Physics*, pages 187–207, 2002.

[163] Rainer Storn and Kenneth Price. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359, 1997.

[164] Dan Stowell and Mark D Plumbley. Fast multidimensional entropy estimation by k-d partitioning. *IEEE Signal Processing Letters*, 16:537–540, 2009.

[165] Liang Sun, Shinichi Yoshida, Xiaochun Cheng, and Yanchun Liang. A cooperative particle swarm optimizer with statistical variable interdependence learning. *Information Sciences*, 186(1):20–39, 2012.

[166] Yuan Sun, Saman K Halgamuge, Michael Kirley, and Mario A Munoz. On the selection of fitness landscape analysis metrics for continuous optimization problems. In *Information and Automation for Sustainability (ICIAfS), 2014 7th International Conference on*, pages 1–6. IEEE, 2014.

[167] Yuan Sun, Michael Kirley, and Saman Kumara Halgamuge. On the selection of decomposition methods for large scale fully non-separable problems. In *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*, pages 1213–1216. ACM, 2015.

[168] Yuan Sun, Mohammad Nabi Omidvar, Michael Kirley, and Xiaodong Li. Adaptive threshold parameter estimation with recursive differential grouping for problem decomposition. In *Proceedings of the 2018 Annual Conference on Genetic and Evolutionary Computation*. ACM, accepted March 2018.

[169] David T Suzuki, Anthony JF Griffiths, et al. *An introduction to genetic analysis.* WH Freeman and Company., 1976.

[170] Kay Chen Tan, YJ Yang, and Chi Keong Goh. A distributed cooperative coevolutionary algorithm for multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 10(5):527–549, 2006.

[171] Jiliang Tang, Salem Alelyani, and Huan Liu. Feature selection for classification: A review. *Data Classification: Algorithms and Applications*, pages 37–64, 2014.

[172] Ke Tang, X Yao, and Pn Suganthan. Benchmark functions for the CEC'2010 special session and competition on large scale global optimization. *Technique Report, USTC, Natrue Inspired Computation and Applications Laboratory*, (1):1–23, 2010.

[173] RuoLi Tang, Zhou Wu, and YanJun Fang. Adaptive multi-context cooperatively coevolving particle swarm optimization for large-scale problems. *Soft Computing*, pages 1–20, 2016.

[174] Cara Tannenbaum and Nancy L Sheehan. Understanding and preventing drug–drug and drug–gene interactions. *Expert Review of Clinical Pharmacology*, 7(4):533–544, 2014.

[175] Mohammad-H Tayarani-N, Xin Yao, and Hongming Xu. Meta-heuristic algorithms in car engine design: A literature survey. *IEEE Transactions on Evolutionary Computation*, 19(5):609–629, 2015.

[176] Masaru Tezuka, Masaharu Munetomo, and Kiyoshi Akama. Linkage identification by nonlinearity check for real-coded genetic algorithms. In *Proceedings of the 2004 Annual Conference on Genetic and Evolutionary Computation*, pages 222–233. Springer, 2004.

[177] LinYu Tseng and Chun Chen. Multiple trajectory search for large scale global optimization. In *Evolutionary Computation, 2008 IEEE Congress on*, pages 3052–3059. IEEE, 2008.

[178] Miwako Tsuji, Masaharu Munetomo, and Kiyoshi Akama. Modeling dependencies of loci with string classification according to fitness differences. In *Proceedings of*

the 2004 Annual Conference on Genetic and Evolutionary Computation, pages 246–257. Springer, 2004.

[179] Frans Van den Bergh and Andries P Engelbrecht. A cooperative approach to particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, 8(3):225–239, 2004.

[180] Leonardo Vanneschi, Manuel Clergue, Philippe Collard, Marco Tomassini, and Sébastien Vérel. Fitness clouds and problem hardness in genetic programming. In *Proceedings of the 2004 Annual Conference on Genetic and Evolutionary Computation*, pages 690–701. Springer, 2004.

[181] Vesselin K Vassilev, Terence C Fogarty, and Julian F Miller. Information characteristics and the structure of landscapes. *Evolutionary Computation*, 8(1):31–60, 2000.

[182] Sébastien Verel, Philippe Collard, and Manuel Clergue. Where are bottlenecks in nk fitness landscapes? In *Evolutionary Computation, 2003 IEEE Congress on*, volume 1, pages 273–280. IEEE, 2003.

[183] Sébastien Verel, Gabriela Ochoa, and Marco Tomassini. Local optima networks of NK landscapes with neutrality. *IEEE Transactions on Evolutionary Computation*, 15(6):783–797, 2011.

[184] Jorge R Vergara and Pablo A Estévez. A review of feature selection methods based on mutual information. *Neural Computing and Applications*, 24(1):175–186, 2014.

[185] Nguyen Xuan Vinh, Shuo Zhou, Jeffrey Chan, and James Bailey. Can high-order dependencies improve mutual information based feature selection? *Pattern Recognition*, 53:46–58, 2016.

[186] Hui Wang, Zhijian Wu, and Shahryar Rahnamayan. Enhanced opposition-based differential evolution for solving high-dimensional continuous optimization problems. *Soft Computing*, 15(11):2127–2140, 2011.

[187] Jun Wang, Jin-Mao Wei, Zhenglu Yang, and Shu-Qin Wang. Feature selection by maximizing independent classification information. *IEEE Transactions on Knowledge and Data Engineering*, 29(4):828–841, 2017.

[188] Edward Weinberger. Correlated and uncorrelated fitness landscapes and how to tell the difference. *Biological cybernetics*, 63(5):325–336, 1990.

[189] Edward D Weinberger. Local properties of Kauffmans NK model: A tunably rugged energy landscape. *Physical Review A*, 44(10):6399, 1991.

[190] Daniel M Weinreich, Richard A Watson, and Lin Chao. Perspective: sign epistasis and genetic constraint on evolutionary trajectories. *Evolution*, 59(6):1165–1174, 2005.

[191] Thomas Weise, Raymond Chiong, and Ke Tang. Evolutionary optimization: Pitfalls and booby traps. *Journal of Computer Science and Technology*, 27(5):907–936, 2012.

[192] Sewall Wright. The roles of mutation, inbreeding, crossbreeding, and selection in evolution. In *Proceedings of the Sixth International Congress of Genetics*, volume 1, pages 356–366, 1932.

[193] Bing Xue, Mengjie Zhang, and Will N Browne. Particle swarm optimization for feature selection in classification: A multi-objective approach. *IEEE Transactions on Cybernetics*, 43(6):1656–1671, 2013.

[194] Bing Xue, Mengjie Zhang, Will N Browne, and Xin Yao. A survey on evolutionary computation approaches to feature selection. *IEEE Transactions on Evolutionary Computation*, 20(4):606–626, 2016.

[195] Howard Hua Yang and John E Moody. Data visualization and feature selection: New algorithms for nongaussian data. In *Advances in Neural Information Processing Systems*, volume 12, pages 687–693. MIT Press, 1999.

[196] Ming Yang, Mohammad Nabi Omidvar, Changhe Li, Xiaodong Li, Zhihua Cai, Borhan Kazimipour, and Xin Yao. Efficient resource allocation in cooperative co-

evolution for large-scale global optimization. *IEEE Transactions on Evolutionary Computation*, 2016.

[197] Zhenyu Yang, Ke Tang, and Xin Yao. Large scale evolutionary optimization using cooperative coevolution. *Information Sciences*, 178(15):2985–2999, 2008.

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

[199] Zhenyu Yang, Ke Tang, and Xin Yao. Self-adaptive differential evolution with neighborhood search. In *Evolutionary Computation, 2008 IEEE Congress on*, pages 1110–1116. IEEE, 2008.

[200] Zheng Zhao and Huan Liu. Searching for interacting features. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, pages 1156–1161. Morgan Kaufmann Publishers Inc., 2007.

[201] Xiangwei Zheng, Dianjie Lu, Xiaoguang Wang, and Hong Liu. A cooperative coevolutionary biogeography-based optimizer. *Applied Intelligence*, 43(1):95–111, 2015.

Appendix A
Benchmark Functions of MEE

Table A.1 presents the extended benchmark problems that can be used to evaluate the MEE measure proposed in Chapter 3. The base functions that used to generate the benchmark problems are presented in Table A.2. The extended benchmark suite consists of 24 benchmark problems with different level of variable interactions. Operators such as rotating, shifting, symmetry breaking and adding ill-condition are also available in this extended benchmark suite. The letter R represents the rotation operator, which is employed to generate interactions between decision variables. The condition number for all of the benchmark problems is 100 except for f_1. The optima of all the benchmark problems are shifted to $f(\mathbf{0}) = 0$.

Table A.1: The extended benchmark suite with 6 categories and 24 benchmark functions. The base functions are presented in Table A.2. R represents rotation. The global minima for all benchmark functions are $f(\mathbf{0}) = 0$.

Type	Func	Equation
I	f_1	$f(x) = sphere(x)$
	f_2	$f(x) = elli(x)$
	f_3	$f(x) = rast(x)$
	f_4	$f(x) = alpi(x)$
II	f_5	$f(x) = sphe(x(1 : d/2)) + ackl(Rx(d/2 + 1 : d))$
	f_6	$f(x) = elli(x(1 : d/2)) + schw(x(d/2 + 1 : d))$
	f_7	$f(x) = rast(x(1 : d/2)) + rast(Rx(d/2 + 1 : d))$
	f_8	$f(x) = alpi(x(1 : d/2)) + whit(Rx(d/2 + 1 : d))$
III	f_9	$f(x) = \sum_{i=1}^{d/2} scha(x(2i - 1 : 2i))$
	f_{10}	$f(x) = \sum_{i=1}^{d/2} levi(x(2i - 1 : 2i))$
	f_{11}	$f(x) = \sum_{i=1}^{d/2} thr\_hum(x(2i - 1 : 2i))$
	f_{12}	$f(x) = \sum_{i=1}^{d/2} rose(x(2i - 1 : 2i))$
IV	f_{13}	$f(x) = whit(x(1 : d/2)) + whit(x(d/2 + 1 : d))$
	f_{14}	$f(x) = grie(x(1 : d/2)) + grie(Rx(x/2 + 1 : d))$
	f_{15}	$f(x) = rast(Rx(1 : d/2)) + schw(Rx(d/2 + 1 : d))$
	f_{16}	$f(x) = elli(Rx(1 : d/2)) + alpi(Rx(d/2 + 1 : d))$
V	f_{17}	$f(x) = rose(x)$
	f_{18}	$f(x) = \sum_{i=1}^{d-1} grie(x(i : i + 1))$
	f_{19}	$f(x) = \sum_{i=1}^{d/2+1} ackl(Rx(i : i + d/2 - 1))$
	f_{20}	$f(x) = rose(x(1 : d/2)) + rose(x(d/2 + 1 : d))$
VI	f_{21}	$f(x) = rast(Rx)$
	f_{22}	$f(x) = whit(Rx)$
	f_{23}	$f(x) = schw(x) + schw(Rx)$
	f_{24}	$f(x) = rast(Rx) + rose(Rx)$

Table A.2: Base Functions: The dimensionality of the first 9 bases functions (from Sphere to Whitely) are tunable, while the last three functions (Schaffer, Levi, ThreeHump) are two-dimensional.

Name	Equation
Sphere	$sphe(x) = \sum_{i=1}^{d} x_i^2$
Elliptic	$elli(x) = \sum_{i=1}^{d} 10^{3\frac{i-1}{d-1}} x_i^2$
Rastrigin	$rast(x) = 10d - 10 \sum_{i=1}^{d} \cos(2\pi x_i) + \sum_{i=1}^{d} x_i^2$
Alpine	$alpi(x) = \sum_{i=1}^{d} \lvert x_i \sin x_i + 0.1 x_i \rvert$
Ackley	$ackl(x) = 20 - 20 \exp\Bigg(-0.2\sqrt{\frac{\sum_{i=1}^{d} x_i^2}{d}}\Bigg) - \exp\Bigg(\frac{\sum_{i=1}^{d} 2\pi x_i}{d}\Bigg) + e$
Schwefel	$schw(x) = \sum_{i=1}^{d} \big(\sum_{j=1}^{i} x_j \big)^2$
Rosenbrock	$rose(x) = \sum_{i=1}^{d-1} 100(x_i^2 - x_{i+1})^2 + \sum_{i=1}^{d-1} (x_i - 1)^2$
Griewank	$grie(x) = 1 + \frac{1}{4000} \sum_{i=1}^{d} x_i^2 - \prod_{i=1}^{d} \cos \frac{x_i}{\sqrt{i}}$
Whitley	$whit(x) = \sum_{i=1}^{d} \sum_{j=1}^{d} \frac{1}{4000} (100(x_i^2 - x_j)^2 + (1 - x_j)^2)^2 - \cos(100(x_i^2 - x_j)^2 + (1 - x_j)^2) + 1$
Schaffer	$scha(x) = 0.5 + \frac{\sin(x_1^2 - x_2^2)^2 - 0.5}{1 + 0.001(x_1^2 + x_2^2)^2}$
Levi	$levi(x) = \sin(3\pi x_1)^2 + (x_1 - 1)^2(1 + \sin(3\pi x_2)^2) + (x_2 - 1)^2(1 + \sin(2\pi x_2))^2$
ThreeHump	$thr\_hum(x) = 2x_1^2 - 1.05x_1^4 + \frac{x_1^6}{6} + x_1 x_2 + x_2^2$

Appendix B
Experimental Results of RDG

B.1 Detailed Decomposition Results of DG2 and FII

The detailed results of the RDG (with $\alpha = 10^{-12}$), DG2, and FII (with $\epsilon = 10^{-2}$) methods when used to decompose the CEC'2010 and CEC'2013 benchmark problems are presented in Table B.1. DG2 is a non-parametric method. RDG approximates the threshold values based on the magnitude of the objective space. To test the sensitivity of RDG to the control coefficient α, we also present the decomposition results from RDG with $\alpha = 10^{-10}$ in the table. The FII method uses a fix threshold $\epsilon = 10^{-2}$ in the original paper, which is not sufficient to identify variable interactions in all of the benchmark problems. For a fair comparison, we present the decomposition results from FII which uses the same threshold as RDG (with $\alpha = 10^{-12}$). The experimental results show that the average number of FEs used by RDG is less than that used by DG2 and FII.

B.2 Detailed Results from DECC and CMAESCC Comparisons

The detailed optimization results of the eight decomposition methods when embedded into the DECC and CMAESCC frameworks to solve the CEC'2010 and CEC'2013 benchmark problems are presented in Table B.2, Table B.3, Table B.4 and Table B.5. The RDG method achieves overall the best solution quality when compared against the other seven decomposition methods.

Table B.1: The experimental results of the RDG (with $\alpha = 10^{-12}$ or $\alpha = 10^{-10}$), DG2, and FII (with $\epsilon = 10^{-2}$ or $\alpha = 10^{-12}$) methods when used to decompose the CEC'2010 and CEC'2013 benchmark problems. "DA" is the decomposition accuracy; "FEs" is the function evaluations used; "ϵ" is the threshold. DG2 is a non-parametric method.

Bench-	Func	RDG ($\alpha = 10^{-12}$)			RDG ($\alpha = 10^{-10}$)			DG2		FII ($\epsilon = 10^{-2}$)		FII ($\alpha = 10^{-12}$)	
marks	Num	DA	FEs	ϵ	DA	FEs	ϵ	DA	FEs	DA	FEs	DA	FEs
	f_1	–	3.00e+03	4.11e-01	–	3.00e+03	4.11e+01	–	5.00e+05	–	3.00e+03	–	3.01e+03
	f_2	–	3.00e+03	2.49e-08	–	3.00e+03	2.49e-06	–	5.00e+05	–	3.00e+03	–	3.01e+03
	f_3	–	6.00e+03	2.15e-11	–	6.02e+03	2.15e-09	–	5.00e+05	–	3.00e+03	–	4.01e+03
	f_4	100%	4.20e+03	1.03e+04	100%	4.20e+03	1.03e+06	100%	5.00e+05	100%	3.69e+03	100%	3.06e+03
	f_5	100%	4.15e+03	1.14e-03	100%	4.15e+03	1.14e-01	100%	5.00e+05	100%	3.05e+03	100%	3.06e+03
	f_6	100%	5.00e+04	2.13e-05	100%	5.06e+03	2.13e-03	100%	5.00e+05	100%	3.05e+03	100%	3.12e+05
	f_7	100%	4.23e+03	5.17e+00	100%	4.23e+03	5.17e+02	100%	5.00e+05	100%	3.05e+03	100%	3.06e+03
	f_8	100%	5.60e+03	2.62e+05	100%	5.60e+03	2.62e+07	100%	5.00e+05	100%	1.88e+04	100%	3.66e+03
	f_9	100%	1.40e+04	4.88e-01	100%	1.40e+04	4.88e+01	100%	5.00e+05	100%	8.01e+03	100%	8.02e+03
CEC'2010	f_{10}	100%	1.40e+04	2.52e-08	100%	1.40e+04	2.52e-06	100%	5.00e+05	100%	8.01e+03	100%	8.02e+03
	f_{11}	100%	1.36e+04	2.36e-10	100%	1.39e+04	2.36e-08	100%	5.00e+05	97.2%	9.59e+03	100%	1.40e+04
	f_{12}	100%	1.43e+04	4.26e-05	100%	1.43e+04	4.26e-03	100%	5.00e+05	100%	8.01e+03	100%	8.02e+03
	f_{13}	100%	2.92e+04	3.71e+00	100%	2.92e+04	3.71e+02	100%	5.00e+05	100%	9.61e+04	100%	9.61e+04
	f_{14}	100%	2.05e+04	4.15e-01	100%	2.05e+04	4.15e+01	100%	5.00e+05	100%	2.30e+04	100%	2.30e+04
	f_{15}	100%	2.05e+04	2.53e-08	100%	2.05e+04	2.53e-06	100%	5.00e+05	100%	2.30e+04	100%	2.30e+04
	f_{16}	100%	2.09e+04	4.30e-10	100%	2.09e+04	4.30e-08	100%	5.00e+05	96.1%	3.09e+04	100%	2.30e+04
	f_{17}	100%	2.07e+04	1.10e-04	100%	2.07e+04	1.10e-02	100%	5.00e+05	100%	2.30e+04	100%	2.30e+04
	f_{18}	100%	4.98e+04	8.19e+00	100%	4.98e+04	8.19e+02	100%	5.00e+05	100%	3.69e+05	100%	3.69e+05
	f_{19}	100%	6.00e+03	6.14e-04	100%	6.00e+03	6.14e-02	100%	5.00e+05	100%	4.00e+03	100%	4.01e+03
	f_{20}	100%	5.08e+04	8.53e+00	100%	5.08e+04	8.53e+02	100%	5.00e+05	100%	5.03e+05	100%	5.03e+05
	f_1	–	3.00e+03	4.20e-01	–	3.00e+03	4.20e-+01	–	5.00e+05	–	3.00e+03	–	3.01e+03
	f_2	–	3.00e+03	1.31e-07	–	3.00e+03	1.31e-05	–	5.00e+05	–	3.00e+03	–	3.01e+03
	f_3	–	6.00e+03	2.16e-11	–	6.05e+03	2.16e-09	–	5.00e+05	–	3.00e+03	–	4.01e+03
	f_4	100%	9.84e+03	7.22e+01	100%	9.84e+03	7.22e+03	100%	5.00e+05	100%	4.55e+03	100%	4.56e+03
	f_5	100%	1.01e+04	8.03e-05	51.6%	9.02e+03	9.08e-03	100%	5.00e+05	66.3%	4.16e+03	98.3%	4.97e+03
	f_6	100%	1.32e+04	1.07e-06	83.3%	8.55e+03	1.07e-04	100%	5.00e+05	59.6%	3.68e+03	99.3%	6.32e+03
	f_7	100%	9.82e+03	5.82e+05	100%	9.82e+03	5.82e+07	83.3%	5.00e+05	33.3%	7.52e+03	100%	5.06e+03
CEC'2013	f_8	80.0%	1.95e+04	1.20e+06	46.8%	1.54e+04	4.35e+08	78.5%	5.00e+05	16.9%	4.92e+03	48.2%	1.16e+04
	f_9	100%	1.92e+04	6.07e-03	97.3%	2.02e+04	6.07e-01	100%	5.00e+05	99.6%	2.11e+04	99.9%	2.11e+04
	f_{10}	82.7%	1.91e+04	9.80e-05	76.6%	1.96e+04	9.80e-03	100%	5.00e+05	75.8%	1.53e+04	87.8%	1.92e+04
	f_{11}	10.0%	1.06e+04	1.52e+06	83.3%	1.10e+04	1.52e+08	100%	5.00e+05	10.0%	4.77e+03	97.0%	2.04e+04
	f_{12}	100%	5.08e+04	8.57e+00	100%	5.08e+04	8.57e+02	100%	5.00e+05	100%	5.03e+05	100%	5.03e+05
	f_{13}	–	8.39e+03	1.83e+06	–	8.31e+03	1.83e+08	–	4.10e+05	–	4.38e+03	–	8.89e+03
	f_{14}	–	1.61e+04	5.45e+06	–	1.60e+04	5.45e+08	–	4.10e+05	–	4.29e+03	–	9.50e+03
	f_{15}	100%	6.16e+03	2.70e+06	100%	8.11e+03	2.70e+08	100%	5.00e+05	100%	4.00e+03	100%	4.69e+03
Average	-	-	1.47e+04	-	-	1.45e+04	-	-	4.95e+05	-	4.94e+04	-	5.91e+04

Table B.2: The results of the proposed RDG method when embedded into the DECC framework to solve the CEC'2010 benchmark problems. The RDG method is compared with GDG, XDG, DG, DG2, FII, D (delta grouping), and RG methods. The best performances are highlighted in bold (Wilcoxon rank-sum tests (α=0.05) with Holm p-value correction).

Func	Stats	RDG	GDG	XDG	DG	DG2	FII	D	RG
f_1	Median	1.50e-01	3.78e-10	5.58e+02	5.58e+02	4.02e+00	1.50e-01	**0.00e+00**	6.06e-14
	Mean	2.07e+00	3.96e-10	1.37e+04	1.37e+04	8.46e+02	2.07e+00	1.35e-26	9.15e-14
	Std	6.76e+00	1.21e-10	4.12e+04	4.12e+04	3.98e+03	6.76e+00	4.75e-26	7.88e-14
f_2	Median	4.46e+03	5.02e+02	4.46e+03	4.46e+03	4.36e+03	4.46e+03	2.89e+02	**1.17e+02**
	Mean	4.41e+03	4.99e+02	4.44e+03	4.44e+03	4.42e+03	4.41e+03	2.89e+02	1.14e+02
	Std	1.68e+02	2.03e+01	1.70e+02	1.70e+02	1.75e+02	1.68e+02	2.24e+01	2.65e+01
f_3	Median	1.67e+01	1.66e+01	1.68e+01	1.68e+01	1.67e+01	1.67e+01	**1.21e-13**	1.79e+00
	Mean	1.66e+01	1.67e+01	1.68e+01	1.68e+01	1.67e+01	1.66e+01	1.23e-13	1.77e+00
	Std	3.05e-01	3.33e-01	3.36e-01	3.36e-01	3.34e-01	3.37e-01	5.24e-15	3.15e-01
f_4	Median	**6.10e+11**	6.03e+13	**7.16e+11**	5.14e+12	**7.97e+11**	7.44e+11	3.00e+12	1.18e+13
	Mean	6.74e+11	6.53e+13	7.92e+11	5.54e+12	8.16e+11	8.86e+11	3.36e+12	1.10e+13
	Std	3.19e+11	2.01e+13	2.11e+11	2.11e+12	3.41e+11	3.50e+11	1.37e+12	2.84e+12
f_5	Median	**1.32e+08**	3.69e+08	1.61e+08	1.63e+08	1.44e+08	**1.24e+08**	2.62e+08	2.26e+08
	Mean	1.28e+08	3.66e+08	1.63e+08	1.61e+08	1.44e+08	1.27e+08	2.58e+08	2.46e+08
	Std	1.92e+07	2.20e+07	2.33e+07	2.10e+07	2.13e+07	2.08e+07	6.83e+07	5.41e+07
f_6	Median	**1.64e+01**	3.70e+02	**1.62e+01**	**1.64e+01**	**1.53e+01**	**1.64e+01**	3.55e-09	4.95e+06
	Mean	1.63e+01	3.63e+02	1.63e+01	1.64e+01	1.51e+01	1.64e+01	2.84e+06	5.04e+06
	Std	3.70e-01	8.81e+01	3.55e-01	3.61e-01	5.71e-01	3.68e-01	1.42e+07	8.78e+05
f_7	Median	**2.46e+00**	1.64e+10	4.32e+02	8.74e+03	2.33e+01	**8.84e+00**	3.28e+08	4.40e+06
	Mean	2.16e+01	1.71e+10	8.00e+02	1.51e+04	3.36e+02	6.76e+01	3.31e+08	5.13e+06
	Std	7.57e+01	3.17e+09	9.59e+02	1.37e+04	1.01e+03	1.28e+02	1.48e+08	3.70e+06
f_8	Median	**3.06e+00**	4.81e+07	1.59e+01	1.82e+07	5.95e+01	3.37e+07	8.31e+07	8.71e+07
	Mean	1.59e+05	8.82e+07	7.97e+05	2.73e+07	3.19e+05	4.16e+07	1.06e+08	7.34e+07
	Std	7.97e+05	6.85e+07	1.63e+06	2.19e+07	1.10e+06	2.17e+07	8.10e+07	3.17e+07
f_9	Median	**4.65e+07**	3.98e+08	1.10e+08	5.78e+07	6.63e+07	**4.53e+07**	6.30e+07	2.44e+08
	Mean	4.70e+07	4.03e+08	1.12e+08	5.85e+07	6.80e+07	4.64e+07	6.27e+07	2.42e+08
	Std	5.22e+06	2.89e+07	1.17e+07	7.43e+06	9.73e+06	4.92e+06	5.74e+06	2.68e+07
f_{10}	Median	4.33e+03	**3.44e+03**	5.27e+03	4.48e+03	4.64e+03	4.35e+03	1.31e+04	9.48e+03
	Mean	4.33e+03	3.43e+03	5.30e+03	4.50e+03	4.67e+03	4.34e+03	1.31e+04	9.29e+03
	Std	1.39e+02	5.34e+01	1.85e+02	1.17e+02	1.39e+02	1.40e+02	2.27e+02	1.30e+03
f_{11}	Median	1.01e+01	1.07e+01	1.07e+01	1.05e+01	1.03e+01	1.17e+01	**1.49e-13**	2.53e+01
	Mean	9.96e+00	1.04e+01	1.08e+01	1.04e+01	1.04e+01	1.16e+01	1.52e-13	2.51e+01
	Std	7.85e-01	1.05e+00	1.08e+00	1.09e+00	1.05e+00	1.16e+00	1.03e-14	1.46e+00
f_{12}	Median	**1.39e+03**	1.34e+05	1.16e+04	2.36e+03	3.94e+03	**1.36e+03**	4.33e+06	4.50e+04
	Mean	1.53e+03	1.34e+05	1.21e+04	2.38e+03	4.05e+03	1.62e+03	4.36e+06	4.47e+04
	Std	4.66e+02	6.92e+03	2.12e+03	3.01e+02	5.86e+02	6.77e+02	2.24e+05	5.11e+03
f_{13}	Median	**6.80e+02**	9.16e+02	2.98e+03	4.50e+03	1.28e+03	**7.42e+02**	1.03e+03	3.13e+03
	Mean	7.41e+02	9.40e+02	3.65e+03	5.22e+03	1.42e+03	8.86e+02	1.21e+03	4.00e+03
	Std	2.57e+02	2.00e+02	1.80e+03	2.93e+03	6.50e+02	5.34e+02	5.01e+02	2.53e+03
f_{14}	Median	3.48e+08	4.46e+08	5.82e+08	3.32e+08	4.61e+08	3.45e+08	**2.02e+08**	5.89e+08
	Mean	3.47e+08	4.53e+08	5.83e+08	3.42e+08	4.61e+08	3.50e+08	2.02e+08	5.86e+08
	Std	2.31e+07	3.98e+07	3.29e+07	2.30e+07	2.58e+07	2.15e+07	1.54e+07	4.45e+07
f_{15}	Median	**5.83e+03**	6.09e+03	6.35e+03	**5.84e+03**	6.11e+03	**5.86e+03**	1.58e+04	6.64e+03
	Mean	5.84e+03	6.09e+03	6.35e+03	5.84e+03	6.10e+03	5.85e+03	1.59e+04	8.61e+03
	Std	1.01e+02	8.91e+01	8.19e+01	7.48e+01	8.84e+01	7.60e+01	2.37e+02	3.23e+03
f_{16}	Median	**2.66e-13**	5.30e-11	1.73e-08	7.25e-13	5.31e-11	4.76e-13	2.20e-13	7.89e+01
	Mean	2.67e-13	5.47e-11	1.77e-08	7.42e-13	5.41e-11	4.67e-13	9.40e-02	7.77e+01
	Std	9.81e-15	5.61e-12	1.43e-09	6.10e-14	4.83e-12	3.17e-14	3.36e-01	1.47e+01
f_{17}	Median	**4.08e+04**	7.31e+04	1.28e+05	**4.11e+04**	7.25e+04	**4.14e+04**	7.49e+06	1.78e+05
	Mean	4.08e+04	7.43e+04	1.29e+05	4.07e+04	7.31e+04	4.07e+04	7.57e+06	1.76e+05
	Std	2.56e+03	4.37e+03	7.39e+03	2.49e+03	4.28e+03	3.08e+03	3.77e+05	1.02e+04
f_{18}	Median	**1.21e+03**	1.26e+03	1.38e+03	1.50e+10	1.28e+03	**1.23e+03**	1.81e+03	2.35e+04
	Mean	1.19e+03	1.28e+03	1.40e+03	1.48e+10	1.32e+03	1.21e+03	1.81e+03	2.35e+04
	Std	1.69e+02	1.57e+02	1.64e+02	2.34e+09	1.64e+02	1.47e+02	0.00e+00	0.00e+00
f_{19}	Median	1.73e+06	1.87e+06	1.73e+06	1.72e+06	1.86e+06	1.70e+06	1.85e+07	**7.88e+05**
	Mean	1.73e+06	1.88e+06	1.73e+06	1.72e+06	1.85e+06	1.72e+06	1.88e+07	7.74e+05
	Std	7.52e+04	7.34e+04	1.02e+05	9.27e+04	8.50e+04	1.06e+05	1.75e+06	3.94e+04
f_{20}	Median	4.09e+03	6.23e+03	3.64e+04	6.50e+10	6.25e+03	5.52e+03	**1.16e+03**	3.36e+03
	Mean	5.05e+03	2.85e+04	2.07e+05	6.55e+10	2.21e+04	9.83e+03	1.15e+03	3.39e+03
	Std	3.29e+03	7.50e+04	5.08e+05	8.03e+09	7.13e+04	1.13e+04	8.39e+01	3.04e+02

Table B.3: The results of the proposed RDG method when embedded into the DECC framework to solve the CEC'2013 benchmark problems. The RDG method is compared with GDG, XDG, DG, DG2, FII, D (delta grouping), and RG methods. The best performances are highlighted in bold (Wilcoxon rank-sum tests (α=0.05) with Holm p-value correction).

Func	Stats	RDG	GDG	XDG	DG	DG2	FII	D	RG
f_1	Median	4.75e-01	5.15e-10	4.84e+02	4.84e+02	1.12e+01	4.75e-01	**0.00e+00**	1.32e-11
	Mean	3.83e+00	5.21e-10	1.31e+05	1.31e+05	1.56e+02	3.83e+00	4.31e-28	2.59e-11
	Std	6.38e+00	9.61e-11	5.40e+05	5.40e+05	4.78e+02	6.38e+00	1.46e-27	3.83e-11
f_2	Median	1.25e+04	4.61e+02	1.27e+04	1.27e+04	1.23e+04	1.25e+04	2.95e+02	**8.24e+01**
	Mean	1.26e+04	4.62e+02	1.27e+04	1.27e+04	1.25e+04	1.26e+04	2.94e+02	8.53e+01
	Std	5.78e+02	2.23e+01	6.62e+02	6.62e+02	5.44e+02	5.78e+02	1.83e+01	2.71e+01
f_3	Median	2.14e+01	2.14e+01	2.14e+01	2.14e+01	2.14e+01	2.14e+01	2.07e+01	**2.01e+01**
	Mean	2.14e+01	2.14e+01	2.14e+01	2.14e+01	2.14e+01	2.14e+01	2.07e+01	2.01e+01
	Std	1.34e-02	1.73e-02	1.32e-02	1.32e-02	1.30e-02	1.46e-02	2.22e-02	3.11e-03
f_4	Median	4.29e+10	2.85e+11	**8.78e+09**	8.02e+10	5.92e+10	4.42e+10	1.88e+10	8.49e+10
	Mean	4.17e+10	2.83e+11	8.73e+09	8.62e+10	5.79e+10	4.12e+10	1.86e+10	9.00e+10
	Std	1.51e+10	9.40e+10	2.13e+09	5.01e+10	2.21e+10	1.15e+10	8.94e+09	3.64e+10
f_5	Median	**5.09e+06**	7.72e+06	**4.78e+06**	**4.78e+06**	5.37e+06	5.39e+06	7.81e+06	8.61e+06
	Mean	5.09e+06	7.57e+06	4.89e+06	4.89e+06	5.37e+06	5.33e+06	7.65e+06	8.28e+06
	Std	4.81e+05	4.17e+05	5.80e+05	5.80e+05	4.90e+05	3.48e+05	1.93e+06	1.32e+06
f_6	Median	1.06e+06	1.06e+06	1.06e+06	1.06e+06	1.06e+06	1.06e+06	1.06e+06	1.06e+06
	Mean	1.06e+06	1.06e+06	1.06e+06	1.06e+06	1.06e+06	1.06e+06	1.06e+06	1.06e+06
	Std	1.10e+03	1.48e+03	1.62e+03	1.27e+03	1.60e+03	1.09e+03	2.10e+03	1.44e+03
f_7	Median	5.41e+07	2.29e+09	**1.51e+07**	3.88e+08	4.30e+07	1.43e+08	3.33e+09	2.83e+08
	Mean	6.42e+07	2.44e+09	1.57e+07	4.77e+08	4.37e+07	1.57e+08	3.45e+09	3.54e+08
	Std	2.97e+07	7.77e+08	5.46e+06	2.59e+08	1.73e+07	6.53e+07	1.33e+09	2.36e+08
f_8	Median	4.74e+15	7.87e+15	**2.39e+14**	3.74e+15	5.25e+15	7.73e+14	7.94e+14	2.51e+15
	Mean	5.04e+15	8.08e+15	3.11e+14	4.29e+15	5.20e+15	9.56e+14	9.01e+14	2.90e+15
	Std	1.86e+15	3.34e+15	1.86e+14	2.48e+15	1.59e+15	6.87e+14	3.43e+14	1.32e+15
f_9	Median	**4.73e+08**	4.99e+08	5.14e+08	**4.89e+08**	5.11e+08	4.98e+08	6.16e+08	5.68e+08
	Mean	4.78e+08	5.00e+08	5.08e+08	4.87e+08	5.13e+08	4.93e+08	5.90e+08	5.95e+08
	Std	2.43e+07	3.53e+07	3.54e+07	2.42e+07	3.58e+07	3.03e+07	7.72e+07	1.37e+08
f_{10}	Median	9.45e+07	9.47e+07	9.45e+07	9.46e+07	9.46e+07	9.45e+07	9.33e+07	**9.29e+07**
	Mean	9.45e+07	9.46e+07	9.46e+07	9.45e+07	9.46e+07	9.44e+07	9.33e+07	9.29e+07
	Std	2.61e+05	3.67e+05	2.14e+05	2.63e+05	2.65e+05	2.94e+05	4.48e+05	5.89e+05
f_{11}	Median	**5.18e+08**	7.27e+08	**4.83e+08**	2.60e+10	2.40e+09	**5.31e+08**	2.31e+10	5.19e+10
	Mean	5.56e+08	8.35e+08	5.23e+08	4.85e+10	4.95e+09	5.75e+08	6.02e+10	5.93e+10
	Std	1.97e+08	3.57e+08	1.30e+08	5.84e+10	6.88e+09	1.77e+08	7.18e+10	4.24e+10
f_{12}	Median	3.87e+03	5.63e+03	4.23e+04	1.65e+11	6.35e+03	6.00e+03	**1.26e+03**	3.36e+03
	Mean	4.02e+03	6.78e+03	5.80e+05	1.67e+11	6.97e+07	4.97e+04	1.25e+03	3.42e+03
	Std	6.64e+02	3.64e+03	1.70e+06	1.17e+10	3.48e+08	1.66e+05	1.09e+02	2.86e+02
f_{13}	Median	3.16e+09	1.47e+09	**1.11e+09**	1.86e+10	1.36e+09	**1.04e+09**	5.65e+10	5.56e+09
	Mean	3.06e+09	1.49e+09	1.14e+09	2.07e+10	1.45e+09	1.19e+09	5.58e+10	5.75e+09
	Std	6.68e+08	5.50e+08	4.00e+08	5.83e+09	3.54e+08	4.73e+08	9.85e+09	2.38e+09
f_{14}	Median	**2.50e+09**	6.71e+09	**3.13e+09**	1.66e+10	6.07e+09	**2.59e+09**	8.08e+11	6.35e+10
	Mean	2.87e+09	6.94e+09	4.27e+09	1.87e+10	6.64e+09	3.31e+09	7.96e+11	7.68e+10
	Std	1.73e+09	3.84e+09	3.19e+09	1.20e+10	3.31e+09	1.86e+09	2.75e+11	4.97e+10
f_{15}	Median	9.75e+06	1.05e+07	9.92e+06	9.28e+06	1.02e+07	8.97e+06	6.20e+07	**5.03e+06**
	Mean	1.10e+07	1.09e+07	1.01e+07	9.97e+06	1.04e+07	9.73e+06	6.26e+07	5.14e+06
	Std	3.76e+06	2.25e+06	1.80e+06	2.35e+06	2.45e+06	2.17e+06	7.92e+06	4.37e+05

Table B.4: The results of the proposed RDG method when embedded into the CMAESCC framework to solve the CEC'2010 benchmark problems. The RDG method is compared with GDG, XDG, DG, DG2, FII, D (delta grouping), and RG methods. The best performances are highlighted in bold (Wilcoxon rank-sum tests (α=0.05) with Holm p-value correction).

Func	Stats	RDG	GDG	XDG	DG	DG2	FII	D	RG
f_1	Median	2.86e+05	**8.44e-21**	9.72e+05	9.72e+05	5.18e+05	2.86e+05	1.30e+06	1.56e-01
	Mean	2.84e+05	8.71e-21	9.71e+05	9.71e+05	5.23e+05	2.84e+05	1.34e+06	5.58e+01
	Std	2.28e+04	1.02e-21	5.92e+04	5.92e+04	4.31e+04	2.28e+04	4.23e+05	2.34e+02
f_2	Median	4.44e+03	**1.50e+03**	4.40e+03	4.40e+03	4.53e+03	4.44e+03	2.55e+03	2.57e+03
	Mean	4.43e+03	1.50e+03	4.43e+03	4.43e+03	4.51e+03	4.43e+03	2.52e+03	2.56e+03
	Std	1.77e+02	6.38e+01	2.03e+02	2.03e+02	1.73e+02	1.77e+02	9.58e+01	1.21e+02
f_3	Median	1.12e+00	1.09e+00	1.09e+00	1.09e+00	1.14e+00	1.09e+00	3.94e-13	**2.88e-13**
	Mean	1.06e+00	1.03e+00	9.48e-01	9.48e-01	1.05e+00	1.05e+00	3.96e-13	2.83e-13
	Std	3.49e-01	3.37e-01	4.37e-01	4.37e-01	4.06e-01	3.43e-01	1.73e-14	7.30e-15
f_4	Median	9.97e+05	2.40e+09	1.25e+11	8.57e+05	1.61e+06	**5.52e+05**	2.36e+11	1.44e+11
	Mean	1.01e+06	2.41e+09	1.24e+11	8.50e+05	1.60e+06	5.45e+05	2.58e+11	1.37e+11
	Std	9.37e+04	1.45e+09	1.09e+10	9.61e+04	1.33e+05	5.70e+04	1.12e+11	3.18e+10
f_5	Median	**9.05e+07**	1.04e+08	**9.27e+07**	**9.85e+07**	**9.45e+07**	**1.00e+08**	3.08e+08	2.79e+08
	Mean	9.52e+07	1.06e+08	9.33e+07	9.66e+07	9.13e+07	9.88e+07	3.27e+08	2.91e+08
	Std	2.23e+07	2.04e+07	1.78e+07	1.82e+07	2.07e+07	2.28e+07	9.26e+07	7.16e+07
f_6	Median	1.05e+00	**6.04e-08**	1.16e+00	1.03e+00	1.58e+00	1.06e+00	3.91e+06	2.58e+06
	Mean	9.17e-01	6.10e-08	1.12e+00	9.09e-01	1.57e+00	9.50e-01	3.93e+06	2.81e+06
	Std	4.23e-01	8.68e-09	1.10e-01	4.22e-01	9.62e-02	3.77e-01	9.55e+05	7.41e+05
f_7	Median	**7.41e-19**	3.05e-18	**7.41e-19**	4.66e+05	**7.59e-19**	**7.84e-19**	1.77e+05	4.18e+06
	Mean	7.41e-19	2.81e-18	7.47e-19	4.55e+05	7.59e-19	7.56e-19	1.87e+05	4.20e+06
	Std	8.35e-20	7.86e-19	9.28e-20	9.15e+04	9.22e-20	8.11e-20	6.85e+04	2.16e+05
f_8	Median	**2.16e-17**	2.11e+07	**1.90e-17**	6.26e-01	**2.12e-17**	1.57e+06	7.70e+06	9.12e+06
	Mean	7.97e+05	3.20e+07	4.78e+05	6.38e+05	4.78e+05	1.52e+06	2.36e+07	2.46e+07
	Std	1.63e+06	3.75e+07	1.32e+06	1.49e+06	1.32e+06	2.95e+05	2.97e+07	2.44e+07
f_9	Median	4.75e+06	**6.64e+02**	1.00e+07	5.82e+06	6.62e+06	4.62e+06	2.97e+07	6.57e+06
	Mean	4.82e+06	7.62e+02	1.02e+07	5.82e+06	6.62e+06	4.64e+06	2.91e+07	6.67e+06
	Std	5.25e+05	3.19e+02	9.80e+05	5.49e+05	4.33e+05	3.95e+05	3.94e+06	8.65e+05
f_{10}	Median	2.89e+03	**1.73e+03**	2.84e+03	2.76e+03	2.84e+03	2.83e+03	4.01e+03	4.17e+03
	Mean	2.88e+03	1.72e+03	2.82e+03	2.81e+03	2.84e+03	2.84e+03	4.03e+03	4.16e+03
	Std	1.29e+02	7.73e+01	1.22e+02	1.30e+02	1.38e+02	1.27e+02	1.93e+02	1.82e+02
f_{11}	Median	**1.51e-12**	**1.52e-12**	**1.52e-12**	2.10e+01	**1.52e-12**	2.97e+01	1.15e+02	1.17e+02
	Mean	3.58e-02	7.64e-02	7.70e-02	2.09e+01	3.52e-02	2.99e+01	1.12e+02	1.16e+02
	Std	1.79e-01	2.65e-01	2.67e-01	3.83e-01	1.76e-01	2.19e+00	1.49e+01	2.02e+01
f_{12}	Median	4.31e-22	**5.51e-24**	4.37e-22	4.09e-22	4.38e-22	4.27e-22	4.12e+04	3.64e-04
	Mean	4.23e-22	5.56e-24	4.37e-22	3.86e-22	4.26e-22	4.17e-22	3.87e+04	3.83e-04
	Std	8.39e-23	3.77e-25	3.60e-23	1.34e-22	8.96e-23	8.20e-23	1.05e+04	1.59e-04
f_{13}	Median	**3.99e+00**	1.42e+02	**3.99e+00**	6.80e+02	**3.99e+00**	**3.99e+00**	1.03e+03	3.08e+02
	Mean	5.90e+00	1.85e+02	5.26e+00	6.98e+02	5.90e+00	7.02e+00	1.32e+03	3.62e+02
	Std	4.01e+00	8.07e+01	3.77e+00	2.92e+02	4.32e+00	4.92e+00	6.99e+02	1.86e+02
f_{14}	Median	**3.91e-20**	2.03e-19	4.73e-01	4.07e-20	1.89e-19	4.07e-20	6.91e+07	1.85e+07
	Mean	3.91e-20	2.04e-19	1.34e+00	4.06e-20	1.98e-19	4.04e-20	7.11e+07	1.85e+07
	Std	2.12e-21	4.17e-20	2.42e+00	1.89e-21	3.35e-20	1.44e-21	1.01e+07	1.88e+06
f_{15}	Median	**1.93e+03**	**1.93e+03**	**1.93e+03**	**1.91e+03**	**1.92e+03**	**1.94e+03**	4.34e+03	4.29e+03
	Mean	1.95e+03	1.92e+03	1.91e+03	1.93e+03	1.92e+03	1.94e+03	4.35e+03	4.37e+03
	Std	1.11e+02	9.80e+01	9.95e+01	9.41e+01	6.82e+01	6.68e+01	2.17e+02	2.97e+02
f_{16}	Median	**8.42e-13**	8.70e-13	9.45e-13	**8.53e-13**	8.70e-13	**8.14e-13**	2.19e+02	2.23e+02
	Mean	8.44e-13	8.72e-13	9.43e-13	8.48e-13	8.73e-13	8.12e-13	2.12e+02	2.16e+02
	Std	2.10e-14	2.11e-14	2.73e-14	2.16e-14	2.34e-14	2.15e-14	2.00e+01	2.96e+01
f_{17}	Median	**6.90e-24**	7.21e-24	8.29e-24	**6.95e-24**	7.33e-24	**6.94e-24**	1.18e+05	9.77e+00
	Mean	6.91e-24	7.29e-24	8.29e-24	6.93e-24	7.36e-24	6.97e-24	1.19e+05	9.95e+00
	Std	2.06e-25	3.06e-25	3.14e-25	2.38e-25	2.48e-25	2.53e-25	1.31e+04	1.71e+00
f_{18}	Median	**1.55e+01**	3.01e+01	1.78e+02	4.33e+03	3.22e+01	2.33e+01	2.50e+03	1.27e+03
	Mean	1.50e+01	4.97e+01	2.00e+02	7.35e+04	4.33e+01	3.27e+01	2.47e+03	1.37e+03
	Std	7.20e+00	4.67e+01	7.68e+01	1.14e+05	3.01e+01	2.53e+01	8.60e+02	4.26e+02
f_{19}	Median	**5.64e+03**	2.00e+04	**5.30e+03**	**5.33e+03**	1.99e+04	**5.30e+03**	6.29e+06	9.69e+06
	Mean	5.47e+03	1.98e+04	5.29e+03	5.23e+03	2.00e+04	5.29e+03	6.45e+06	9.79e+06
	Std	7.08e+02	1.89e+03	6.35e+02	7.31e+02	2.38e+03	6.35e+02	8.17e+05	6.80e+05
f_{20}	Median	**8.55e+02**	**8.29e+02**	8.65e+02	1.09e+03	**8.30e+02**	**8.27e+02**	9.75e+02	9.67e+02
	Mean	8.27e+02	8.34e+02	8.61e+02	1.12e+03	8.36e+02	8.24e+02	9.90e+02	9.72e+02
	Std	6.35e+01	5.79e+01	4.79e+01	1.05e+02	5.07e+01	5.32e+01	2.68e+01	1.20e+01

Table B.5: The results of the proposed RDG method when embedded into the CMAESCC framework to solve the CEC'2013 benchmark problems. The RDG method is compared with GDG, XDG, DG, DG2, FII, D (delta grouping), and RG methods. The best performances are highlighted in bold (Wilcoxon rank-sum tests (α=0.05) with Holm p-value correction).

Func	Stats	RDG	GDG	XDG	DG	DG2	FII	D	RG
	Median	2.85e+05	**1.02e-20**	1.02e+06	1.02e+06	5.48e+05	2.85e+05	1.61e+06	8.26e+02
f_1	Mean	2.90e+05	1.04e-20	1.04e+06	1.04e+06	5.52e+05	2.90e+05	1.54e+06	1.64e+03
	Std	3.28e+04	9.90e-22	1.13e+05	1.13e+05	5.88e+04	3.28e+04	5.19e+05	1.94e+03
	Median	4.67e+03	**1.55e+03**	4.72e+03	4.72e+03	4.69e+03	4.67e+03	2.64e+03	2.72e+03
f_2	Mean	4.69e+03	1.54e+03	4.74e+03	4.74e+03	4.69e+03	4.69e+03	2.66e+03	2.69e+03
	Std	1.78e+02	7.52e+01	2.24e+02	2.24e+02	1.81e+02	1.78e+02	1.70e+02	1.02e+02
	Median	2.04e+01	2.04e+01	2.05e+01	2.05e+01	2.04e+01	2.04e+01	2.02e+01	**2.00e+01**
f_3	Mean	2.04e+01	2.04e+01	2.05e+01	2.05e+01	2.04e+01	2.04e+01	2.02e+01	2.00e+01
	Std	4.96e-02	4.28e-02	5.25e-02	5.25e-02	5.21e-02	5.29e-02	1.90e-02	7.86e-03
	Median	5.81e+06	**6.35e+04**	3.55e+09	1.45e+10	8.43e+06	5.91e+06	2.37e+09	8.59e+08
f_4	Mean	5.83e+06	7.31e+04	3.51e+09	1.49e+10	8.52e+06	5.88e+06	2.45e+09	9.48e+08
	Std	6.32e+05	3.71e+04	3.02e+08	2.21e+09	8.54e+05	5.44e+05	9.95e+08	3.78e+08
	Median	2.35e+06	2.23e+06	**1.60e+06**	**1.60e+06**	2.17e+06	2.15e+06	5.90e+06	5.63e+06
f_5	Mean	2.40e+06	2.23e+06	1.59e+06	1.59e+06	2.19e+06	2.22e+06	5.82e+06	5.66e+06
	Std	4.36e+05	4.24e+05	2.38e+05	2.38e+05	3.51e+05	3.88e+05	1.10e+06	1.28e+06
	Median	**9.96e+05**	**9.96e+05**	**9.96e+05**	**9.96e+05**	**9.96e+05**	**9.96e+05**	1.06e+06	1.06e+06
f_6	Mean	9.96e+05	9.96e+05	9.96e+05	9.96e+05	9.96e+05	9.96e+05	1.06e+06	1.06e+06
	Std	1.48e+02	1.70e+03	8.83e+00	5.57e+02	3.31e+02	5.93e+01	1.49e+03	1.29e+03
	Median	**2.94e-20**	3.61e+07	9.22e+05	1.36e+06	1.00e+03	1.56e+05	6.27e+06	1.31e+06
f_7	Mean	8.12e-17	3.73e+07	9.20e+05	1.40e+06	1.05e+03	1.53e+05	6.90e+06	1.33e+06
	Std	2.17e-16	1.30e+07	7.47e+04	1.81e+05	2.79e+02	2.53e+04	3.55e+06	1.35e+05
	Median	**8.26e+06**	1.14e+08	2.67e+13	5.68e+13	3.57e+07	5.57e+13	5.47e+13	3.98e+13
f_8	Mean	8.51e+06	1.28e+08	2.73e+13	5.58e+13	3.85e+07	5.73e+13	5.35e+13	4.25e+13
	Std	2.92e+06	3.52e+07	8.01e+12	2.04e+13	1.09e+07	1.28e+13	1.65e+13	1.84e+13
	Median	**1.58e+08**	**1.58e+08**	**1.51e+08**	**1.61e+08**	**1.52e+08**	**1.68e+08**	4.41e+08	4.74e+08
f_9	Mean	1.65e+08	1.67e+08	1.66e+08	1.56e+08	1.51e+08	1.71e+08	4.54e+08	4.96e+08
	Std	4.16e+07	3.88e+07	3.49e+07	2.75e+07	2.87e+07	3.29e+07	9.83e+07	1.17e+08
	Median	**9.05e+07**	9.06e+07	**9.05e+07**	9.05e+07	**9.05e+07**	**9.05e+07**	9.32e+07	9.29e+07
f_{10}	Mean	9.10e+07	9.11e+07	9.07e+07	9.14e+07	9.13e+07	9.07e+07	9.33e+07	9.30e+07
	Std	1.29e+06	1.20e+06	3.09e+05	1.64e+06	1.51e+06	8.45e+05	3.65e+05	5.71e+05
	Median	1.68e+07	2.57e+07	1.71e+07	2.90e+08	**1.56e+05**	1.72e+07	1.32e+08	1.32e+08
f_{11}	Mean	1.67e+07	2.53e+07	1.73e+07	4.67e+08	2.47e+05	1.67e+07	1.63e+08	1.63e+08
	Std	1.62e+06	2.69e+06	1.62e+06	3.29e+08	2.37e+05	1.67e+06	9.02e+07	9.02e+07
	Median	**1.01e+03**	1.02e+03	1.03e+03	1.24e+03	1.02e+03	1.02e+03	1.03e+03	1.03e+03
f_{12}	Mean	9.81e+02	1.00e+03	1.02e+03	1.25e+03	1.01e+03	1.02e+03	1.05e+03	1.03e+03
	Std	7.30e+01	3.91e+01	4.67e+01	1.36e+02	5.81e+01	4.52e+01	2.43e+01	1.76e+01
	Median	2.49e+06	2.29e+06	**1.47e+06**	3.40e+07	2.28e+06	**1.59e+06**	1.93e+09	1.31e+08
f_{13}	Mean	2.47e+06	2.36e+06	1.53e+06	3.40e+07	2.43e+06	1.55e+06	1.96e+09	1.49e+08
	Std	3.83e+05	3.38e+05	2.00e+05	1.10e+07	3.70e+05	1.85e+05	1.02e+09	7.64e+07
	Median	2.74e+07	3.60e+07	2.75e+07	**6.22e+06**	3.66e+07	2.73e+07	1.58e+09	1.85e+08
f_{14}	Mean	2.77e+07	3.63e+07	2.81e+07	7.26e+06	3.59e+07	2.74e+07	2.79e+09	1.84e+08
	Std	1.80e+06	3.18e+06	2.25e+06	2.12e+06	2.85e+06	2.40e+06	3.47e+09	3.22e+07
	Median	**2.19e+06**	3.04e+06	**2.36e+06**	**2.21e+06**	2.93e+06	**2.36e+06**	1.72e+07	3.68e+06
f_{15}	Mean	2.19e+06	3.05e+06	2.33e+06	2.25e+06	3.02e+06	2.33e+06	1.69e+07	6.69e+06
	Std	2.28e+05	3.35e+05	2.92e+05	1.55e+05	3.30e+05	2.92e+05	1.80e+06	7.49e+06