采用协同演化算法解决
大规模数值优化问题的研究
Large-Scale Numerical
Optimization using Cooperative
Coevolutionary Algorithms
计算机科学与技术学院

陈文祥

PB07210425

唐珂 教授 二〇一一年六月

中国科学技术大学

University of Science and Technology of China

本科毕业论文

A Dissertation Submitted for the Bachelor's Degree

采用协同演化算法解决大规模数值优化问题的研究
Large-Scale Numerical Optimization using Cooperative
Coevolutionary Algorithms

姓 名	陈文祥
B.S. Candidate	Wenxiang Chen
导 师	唐珂 教授
Supervisor	Prof. Ke Tang

2011年6月

June, 2010

致 谢

随着本科毕业论文的完成,我在科大四年的学习接近尾声,而在自然 计算与应用实验室已经学习研究了两年零两个月。衷心感谢自然计算与应 用实验室的各位老师及师兄师姐们在两年多来给予我的热情鼓励和耐心帮 助,是你们帮助我发掘兴趣所在,让我自由而愉快的徜徉于科学的海洋, 决定了我的人生轨迹!

首先感谢唐珂老师两年来点点滴滴对我科研上的指导和生活上的关心。感谢您提供给我如同博士生般的待遇:从台式计算机购买到服务器使用、从鼓励我参加最前沿的尖端课题到全力资助我参加国际学术会议,是您让我感觉到,一个本科生也同样能够做一点真正的研究。感谢您总是在我需要帮助的时候,百忙之中抽出时间,解除了我诸多科研上的疑惑和指明了前方的道路。感谢您一直把学生的发展和利益放在首位,给予我许多关于未来职业规划的指导和建议。感谢您让彭飞师兄、杨振宇师兄和Thomas Weise 博士直接帮助我解决工作中众多琐碎的细节问题,他们亦师亦友,扫除了我科研道路上无数的绊脚石。

感谢彭飞师兄的引荐。如果没有师兄当年在数据结构实验课上的一番 关于 NICAL 颇具感染力的介绍,我可能错过 NICAL 这个让我蜕变的熔 炉。虽然当时选择 NICAL 前并没有进行对多个实验室的理性比较,而更 多的是出于对师兄的信任。现在回头看来,如果再让我选择一次,我肯定 还会成为一个 NICALer! 感谢师兄在我刚刚进入实验室时,耐心解答了关 于论文的阅读、各种研究相关软件的使用等细节问题,你乐于助人的精神 永远值得我学习。

感谢杨振宇师兄领我进入了大规模数值优化这个前沿课题。师兄对大规模数值优化问题的独到见解,就像是一把进入密室的钥匙。是师兄让我快速登堂入室,从 2010 年 1 月底开始接触具体的课题,到同年 4 月 20 日投出第一篇论文并被 PPSN 接受,如果没有师兄的帮助,这是绝对不可能完成的。我对科研具体流程的把握,正是通过和师兄一次次讨论逐步形成的。

感谢姚新老师高屋建瓴的指导。和姚老师的交流机会并不是太多,不过每次交流都是极具启发性的,是我产生新思路的最重要源泉。PPSN会议期间您热情邀请我到你所下榻的旅馆就餐,中途还遇见进化策略的发明者 H.P. Schwefel 教授,还获得了和他握手的机会,让我感受到演化计算社区的温暖,有了继续在这个领域探索的勇气。PPSN会议和两次由姚老

师发起的 workshop,让我看到演化计算的研究当前蓬勃的发展趋势,使 我坚定了在这个领域继续研究的信心。

感谢 Thomas Weise 博士就科技论文写作方面提供的悉心指导,你严谨的写作风格将使我受用终生。感谢你在对我工作进展孜孜不倦的关心,在失落时给我持续不断的鼓励。

感谢科大所有有教过我的和没有教过我老师,你们是科大真正的脊梁!我坚信,科大在你们兢兢业业的不懈努力下,一定能够重振往日的雄风!你们在传授给我知识的同时,踏实的作风更是给我留下了深刻的印象,你们永远是我精神丰碑。

感谢三位可爱的室友,司成可、李啸海和周金红,是你们让小小的寝 室有了家的温馨。感谢你们一直以来对我科研工作的鼓励和支持。

感谢 07 级计算机学院全体同学,和你们一起度过的四年欢乐时光我 将永生难忘!

感谢生我养我的、最可敬的父母亲,虽然由于种种原因你们都没有完成正常的教育,但你们在很多生活问题中体现出的睿智、对我人生积极向上态度的培养、对我任何选择的信任和尊重,你们是所知道的最好的家长!

Table of Contents

致	谢		i
摘	要		vi
${f Abstr}$	act		viii
Chapt	ter 1 I	ntroduction	1
1.1	Motiv	ation	2
1.2	Objec	tives	4
Chapt	ter 2 B	Background	6
2.1	Evolu	tionary Computation	6
2.2	Coope	erative Coevolution	7
Chapt	ter 3 F	formulating Interdependencies among Subproblems	11
3.1	Backg	round and Related Works	11
	3.1.1	Classical Benchmark Problems	11
	3.1.2	Interdependencies among Subcomponents	11
	3.1.3	Limitations for previous Formulation of Interdependency	
		in Cooperative Coevolution	12
3.2	New I	Formation that Never Commits the Type-II Error	14
Chapt	ter 4 I	mpact of Problem Decomposition Strategy on Cooper-	-
ati	ve Coe	volution	17
4.1	Interp	retation of Problem Decomposition Strategy	17
4.2	How t	o Handle Unknown Elements in \vec{A}_{intr}	18
4.3	Exper	imental Studies	18
	4.3.1	Experimental Setup	18
	4.3.2	Experimental Results with P_{prior} varying from 0% to 100%	19
	4.3.3	Experimental Results with P_{prior} varying from 1% to 9% .	19
	434	Insights into the Underlying Reasons	22

Chapte	er 5 E	nable Variable Interaction Learning in Cooperative	;
Coe	evoluti	onary Algorithms	28
5.1	Backg	round	28
	5.1.1	Discovering Decision Variable Interactions	29
	5.1.2	Related Works	31
5.2	Coope	erative Coevolution with Variable Interaction Learning	32
	5.2.1	Learning Stage	33
	5.2.2	Optimization Stage	35
5.3	Exper	imental Studies	36
	5.3.1	Experimental Setup	36
	5.3.2	Benchmark Functions and Learned Groups	36
	5.3.3	Comparison with other CC-based algorithms and JADE $$.	38
5.4	Summ	nary	41
Chapte	er 6 E	Companies the Effectiveness of Variable Interaction Learn-	•
ing	Mecha	anism	42
6.1	Simpli	ify Learning Process by Employing Occam's Razor	42
	6.1.1	Random Sampling Learning Process (RS-LP)	43
	6.1.2	Discussion on Parameter Termination Condition	44
	6.1.3	Experimental Study	45
6.2	Inspir	ations From Linkage Learning Techniques	46
	6.2.1	Background	46
	6.2.2	Generalizing the Definition of Variable Interaction	48
	6.2.3	Will Local Search be Helpful for Learning Variable Interaction?	49
	6.2.4	Reducing the Runtime Complexity of Learning Process By	
		Binary Search	49
	6.2.5	Experimental Studies	50

中国科学技术大学本科毕业论文

Chapter	7 C	onclusions	56
7.1	Summa	ary	56
7.2	Future	Works	57
	7.2.1	Verification of the Efficacy of CCVIL in various Real-World	
		Applications	58
	7.2.2	The Scalability of CCVIL	58
	7.2.3	$\label{eq:multiple optimization} \mbox{Multiple Optimization Strategies in the Optimization Stage}$	58
	7.2.4	More Sophisticated Approaches for Separable yet Not Ad-	
		ditively Separable Functions	59
	7.2.5	Evolving Variable Partitioning	59
Bibliogr	aphy		60

摘 要

数值优化存在广泛的应用背景,很多科学研究与工程实践问题都可抽象为数值优化模型加以求解。演化算法作为一类基于群体搜索的启发式方法,因具有使用简单、全局搜索能力强等特点,在数值优化领域受到越来越多的关注。然而,因为"维数灾难"问题的存在,这类方法仍然存在无法有效求解大规模问题等不足,严重影响了算法的求解效率与可扩展性,从而限制了其在相关应用领域的发展。

基于"分而治之"思想的"合作型协同演化算法"在变量相关性信息已知的情况下,被认为是解决大规模数值优化难题有潜力的方法。然而在实际生产生活中,所面对的问题往往无从事先知道变量相关性信息(即黑盒难题),从而无法确保合理的将原问题分解成若干个相互独立的子问题。而不合理的问题分解策略使得分解之后的子问题存在相互依赖关系,这将是极大的限制协同演化算法全局搜索能力,从而使算法陷入局部最优解而停滞不前。针对合作性协同演化算法目前存在的不足,本论文主要从以下几个方面开展研究工作并提出一系列性能国际领先的新方法:

- 1. 以二进制串编码的遗传算法中的连系学习一直是过去 20 年来的研究 热点。然而在以实数编码的数值优化领域,目前几乎没有工作讨论变 量相关性学习的问题,本论文的出现填补这一空白。为了描述合作型 协同演化算法中子问题之间的相互依赖关系,本论文在国际上首次 提出了变量相关性的数学定义。
- 2. 变量相关性信息由少到多的条件下,合作型协同演化算法性能变化的实验研究结果表明,(即使是部分的)相关性信息的获取将显著提升算法的性能。本论文以此阐明黑盒问题的变量相关性学习是一个有重要意义而亟待解决的问题。
- 3. 基于第1项变量相关性的定义,在合作性协同演化算法中成功引入系统的变量相关性学习机制。基于公认测试函数集的研究结果表明,本论文提出的变量相关性定义及基于此的学习机制较好表达了子问题之间的相互依赖关系,提出的算法成为性能国际领先的合作性协同演化算法。
- 4. 根据奥卡姆剃刀原理,本论文将学习机制中复杂的操作除去,直接使用随机采样考察问题相关性特征。新算法使用更少的计算代价却能

够学习到更多的相关性,算法的整体性能又有了进一步的提升,已经超过测试函数集的世界冠军,成为目前性能最好的算法。

5. 本论文受到遗传算法中连系定义的启发,进一步提出推广之后的变量相关性定义。使用基于推广后定义的合作性协同演化算法在保证正确检测子问题之间相互倚赖关系的前提下,相关性学习效率统计显著的增强。更进一步,本论文还将遗传算法中的一种先进的连系学习方法推广到连续空间的数值优化领域。这一推广将原变量相关性学习机制的时间复杂度从 $O(N^2)$ 降至 $O(N \times log N)$,成为目前时间复杂度最低的实数域变量相关性学习方法。

关键字: 大规模数值优化,高维数值优化,合作型协同演化,变量相关性学习,演化算法,遗传算法,连系学习

ABSTRACT

Numerical optimization, which is the basic mathematical model for various problems emerging from scientific researches and industrial applications, is a critical research topic and widely used technique. Evolutionary algorithms, are a set of heuristic approaches based on behaviors of searching agents. Evolutionary algorithms are becoming increasingly popular in the domain of numerical optimization, since they require no sophisticated knowledge to employ and are robust in performing global search. However, "the Curse of Dimensionality", dramatically degrades the efficacy and scalability of evolutionary algorithms, which in return, limits the development of their application in real world.

Cooperative Cevolutionary Algorithms are proposed for tackling complex problems in a "divide-and-conquer" fashion. With prior knowledge given on the problem's separability, Cooperative Coevolutionary algorithms have been proved to be more efficient in high-dimensional optimization compared with conventional approaches. Nevertheless, for black-box optimization, which is usually the case in practice, inappropriate problems decomposition leads to interdependencies among subproblems. The interdependencies trap algorithms into local optima, yielding solutions far from satisfactory. Towards solving the decomposition issue in Cooperative Coevolution, this thesis follows the steps as listed below and proposes several highly competent approaches:

- 1. Linkage learning is a popular topic in domain of genetic algorithms, whose candidate solutions are encoded into bit strings. However, there are little research conducted on analysis of variable Interdependency with representation of real values. Towards filling up this gap, this thesis formulates the interdependencies among subproblems by introducing the definition of variable interaction.
- 2. Conducting comprehensive experimental studies on the dynamics of Cooperative Coevlutionary algorithms, when the portion of prior information on variable interaction varies from none to full. The study demonstrates that incorporating the (even partial) knowledge on vari-

- able interaction is very helpful in enhancing the searching capacity of cooperative coevolutionary algorithms. Consequently, it is critical to examine the issue on how to identify the variable interaction within a black-box problem.
- 3. Next, based on the definition proposed in Item 1, we introduce a systematic variable interaction learning mechanism, yielding a two-staged cooperative coevolutionary algorithm. The simulation result, which is on the basis of widely used benchmark set for large-scale numerical optimization, shows the learning mechanism address the interdependencies among subproblems very well. The newly designed cooperative coevolutionary algorithms outperforms both state-of-the-art cooperative coevolutionary algorithms and the internal optimizer remarkably.
- 4. Motivated by the principle of the Occam's Razor, we remove all complex operations in the learning mechanism. Only the most basic search algorithm, random sampling, is directly employed in exploring the characteristics of problems. The new algorithms consumes less fitness evaluation, yet results in discovering more variable interactions. The overall algorithms found statistically significantly better solution than the original one, and even outperforms the champion of the CEC'2010 competition on large-scale global optimization.
- 5. Finally, inspired by linkage learning techniques in genetic algorithms, this thesis further generalizes the definition of variable interaction to include more cases. The experimental study indicates that the generalized definition significantly reinforce the efficiency of learning mechanism, while maintaining the correctness of all detected variable interactions. We also extend a competent linkage learning technique in the field of genetic algorithms to the domain of numerical optimization on continuous space. This advancement reduces the runtime complexity of identifying the interaction between a pair of variables from O(N) to $O(N \times log N)$. It is currently the fastest method of variable interaction learning in terms of runtime complexity.

Keywords: Large-Scale Numerical Optimization, High-Dimensional Optimization,

Cooperative Coevolution, Variable Interaction Learning, Evolutionary Computation, Evolutionary Algorithms, Genetic Algorithms, Linkage Learning, Epistasis, Variable Interdependency

1 Introduction

Optimization deals with the problem of minimizing/maximizing a certain objective subject to some set of side constraints [1]. Such problems appear in everyone's daily life. For instance, when one tries to go from A to B by plane, train, or car as fast as possible or when one tries to buy some special goods available at different stores as cheap as possible. Optimization problems are mathematically modeled by introducing variables reflecting the options/quantities to be determined and by expressing the objective and the side constraints by functions defined on the domains of the variables. In particular, numerical optimization deal with real-value variables on continuous domain.

As pointed out by Jorge Moré [2],

...interesting optimization problems that arise in applications, for example, fluid dynamics, medicine, elasticity, combustion, molecular design, nondestructive testing, chemical kinetics, lubrication, optimal design, and superconductivity.

numerical optimization problems arise in almost every aspect of science, technology, and business. It is therefore an essential request on solving numerical optimization problems effectively.

Computational models of biological evolution have a number of advantages over other problem-solving methods [3]:

1. They can be applied when little knowledge is given about the problem being solved. For example, unlike some other problem-solving methods, the application of an evolutionary algorithm to a function optimization problem would not require knowledge of first or second derivatives, discontinuities, etc. The minimal requirement is the ability to evaluate the relative worth of candidate solutions approximately. In the domain of evolutionary computation, we use the term *fitness* to describe the relative worth of a solution, which is defined by some objective function.

- 2. Evolutionary computation is more robust and less likely to trap in local optima. This is because evolutionary algorithms maintain a population of alternative solutions and strike a balance between exploiting regions of the search space that have previously discovered fit individuals and continuing to explore uncharted territory.
- 3. Evolutionary computation can be applied in the context of noisy or dynamic objective functions. This advantage makes the model of evolutionary computation attractive for solving problems in a wide range of domains, particularly when the goal is to construct a system that manifests some of the characteristics of biology, such as intelligence or the ability to adapt to change.

Since many of numerical problems cannot be solved analytically, Evolutionary Algorithms (EAs) become one of the most popular methods to address them [4]. EAs are a family of stochastic search algorithms that take the inspiration from natural selection and survival. EAs maintain a number of individuals and explore distributed regions of the search space with their multiple search agents (individuals). Cooperation between search agents guide them to exploit the most promising regions, which accelerates the search process remarkably [5, 6, 7].

In the past few decades, various EAs and numbers of their variants have been developed and employed in optimization problems [8], e.g., Genetic Algorithm (GA) [9, 10], Evolutionary Programming (EP) [11, 12], Evolution Strategies (ES) [13], Differential Evolution (DE) [14], Estimation of Distribution Algorithms (EDA) [15], etc.

1.1 Motivation

Most of the studies demonstrate remarkable performance of EAs, but these studies are mostly limited to small-scale problems, whose numbers of decision variables range from 30 to 100. Their performance on even larger problems have rarely been tested. Some researchers report that conventional EAs suffer from "curse of dimensionality" [16, 17], which describes the phenomenon where

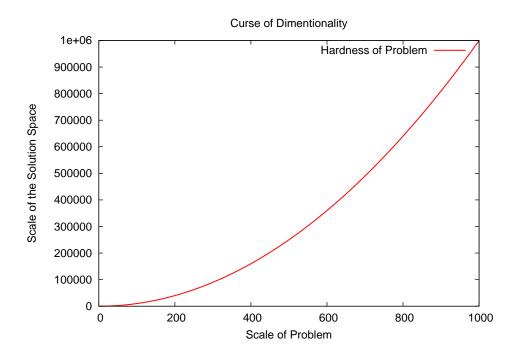


Figure 1.1: Curse of Dimensionality: The volume of search space grows exponentially as the dimension increases. In most cases, the volume of search space determine the hardness of solving the problem. Thus, the difficulty of solving a problem increase exponentially as the dimension raise.

the performance of optimization algorithms *degrade* rapidly as the dimensions of problems rise up [18] (See Figure 1.1).

However, quite a few real-world applications, such as evolving large-scale artificial neural networks, high-dimensional scheduling problems, vehicle routing in large-scale traffic network and designing large-scale electronic systems, face large-scale (usually means having more than 1000 decision variables) and hard problems. Consequently, it is highly urgent to develop effective large-scale optimization algorithms.

It is an intuitive consideration to decompose the original large-scale problem into several moderate-scale subproblems. As the subproblems contain moderate amount of decision variables, they can be solved by conventional approaches. Ideally, solving all subproblems directly leads to the optimal solution to the original large-scale problem. Yet interdependencies may exist among subproblems, these render it impossible to consider each subproblem separably. For example, in some corporation, the personal goals of stuff are likely to contradict from one to another. In this case, the cooperation is required among employees and the interests of several stuff members should be considered jointly, so that the corporation as a whole will be able to gain the most benefit.

In addition, it is also impossible that all decision variables are strongly interacting with each other. In a certain university, a student from physics department tend to have rather weak connection to a cleaner working in the art building, even though both of them are factors influencing the university entity.

To sum up briefly, there are two assumptions underlying this thesis: 1) In a wide variety of real-world problems, there exist interdependencies among certain variables. 2) In most problems, it is highly unlikely that all variables are closely interdependent. Under the two assumptions, it would be both crucial and attractive to understand the importance of proper decomposition strategy in cooperative coevolutionary algorithms, design effective approaches for problem decomposition, and thus solve large-scale numerical problems effectively.

1.2 Objectives

The primary goal of this thesis is to understand the true difficulties of solving large-scale numerical optimization, design effective algorithms that scale up well on high-dimensional problems, initiate the studies on variable interaction learning with representation of real values, and provide general guideline on how to solve complex problems where thousands of decision variables are involved. This thesis is going to achieve this goal by following path listed below:

1. To propose a definition, variable interaction, for depicting the interdependencies among subproblems formally, so that the definition can be the principal basis for analyzing the separability of problems in numerical optimization.

- 2. To verify the importance of properly decomposing problems, by comprehensive empirical study.
- 3. To design a mechanism for identifying the variable interactions, and hopefully, with the knowledge of variable interactions, the problem decomposition method will be very close to the ideal one.
- 4. To simplify the learning mechanism according to principle of Occam's Razor, and to investigate towards what directions can the learning process be accelerated.
- 5. To learn from the recent advancement of linkage learning, and propose a more general definition of variable definition, which is supposed to accelerate the learning process without bringing in any side effort like committing type-II errors.
- 6. To further reduce the runtime complexity of learning mechanism by employing binary search on discovering interactions.

2 Background

2.1 Evolutionary Computation

The basic idea of evolutionary computation derives from the evolutionary process of biological systems. Conventional evolutionary computation is with the seminal Darwinian evolutionary theory, the essential components are listed below [5]:

- one or more populations of individuals competing for limited resources,
- the notion of dynamically changing populations due to the birth and death of individuals,
- a concept of fitness which reflects the ability of an individual to survive and reproduce,
- and a concept of variational inheritance: offspring closely resemble their parents, yet are likely to be not identical.

Evolutionary computation is mostly considered as a problem-solver [5]. Conventional Evolutionary Algorithms (EAs) include:

- Genetic Algorithms: Genetic algorithms (GAs) [6, 9, 19, 20] model the evolutionary processes at the level of genome, which is usually represented as bit strings. Standard GA firstly initializes the population over the search space, computes the fitness of each individual, and select the several individuals as the parents. Those parents are used to reproduced with respect to their fitness, i.e, parents with better fitness earn higher chance to reproduce offspring. The reproduction is accompanied with mutation and crossover. See Figure 2.1 for an illustrated example.
- Evolution Strategies: While genetic algorithms simulate evolution at the level of the genome, Evolution Strategies (ES) [13, 21, 22], as well as many other classes of evolutionary algorithms currently in use, directly evolve

Figure 2.1: 2-point Crossover and Mutation in GAs

phenotypes. The original ES consists one parent and one offspring only. A parent is mutated to create an offspring, and the more highly fit of the two individuals survives into the next generation.

• Evolutionary Programming: Evolutionary Programming (EP) [23] initially used finite-state machines as the underlying structure to be evolved, it was extended in the mid-1980s to handle essentially arbitrary data structures, such as applications involving continuous parameters and combinatorial optimization problems.

2.2 Cooperative Coevolution

Potter and De Jong invents the Cooperative Coevolution in 1990s, for the purpose of handling increasingly complex problem by evolving interacting co-adapted subcomponents. It yields a promising performance both in benchmark function and real-world application [24],[25]. In this thesis, we mainly focus on the particular potential of CC, i.e., numerical optimization. Many cooperative coevolutionary algorithms for numerical optimization consist of three basic ingredients: [17]:

1. A decomposition method used to divide the N-dimensional decision vector into groups $G_1 \dots G_m$ of variables. Each such group is optimized with a separate subpopulation of the corresponding dimension $|G_i| < N$.

- 2. In order to evaluate the fitness of the individuals from a certain subpopulation, a representative element from each of the other subpopulations is selected. In this *cooperation* step, a population of complete N-dimensional candidate solutions is constructed by concatenating the representatives to each element of the current subpopulation.
- 3. An optimizer is applied to the population for only *optimizing* the decision variables in the current group.

Figure 2.2 provides an illustrated example showing how cooperative coevolutionary algorithms works.

In the conventional CC framework, optimizing a group with the corresponding subpopulation is called a *phase*. After finishing a *phase*, CC will turn to optimize the next group and start a new *phase*. One iteration over all groups constitutes a *cycle*. A CC algorithm performs several *cycles*. The basic paradigm is sketched out in Figure 2.3. In Algorithms 1 and 2, we sketch the flow of a simple CC algorithm that treats the problem as completely separable [24].

Algorithm 1: $(subpop, pop, b\widetilde{e}st) \leftarrow initializeCC(NP)$

- 1 for $i \leftarrow 1$ to N do
- $subpop_i \longleftarrow NP_i \ random \ one-dimensional \ samples \ from \ a \ given \ interval;$
- $s pop \leftarrow (subpop_1, \ldots, subpop_N);$
- 4 $\vec{best} \leftarrow \arg\min f(pop)$;
- 5 return (subpop, pop, best);

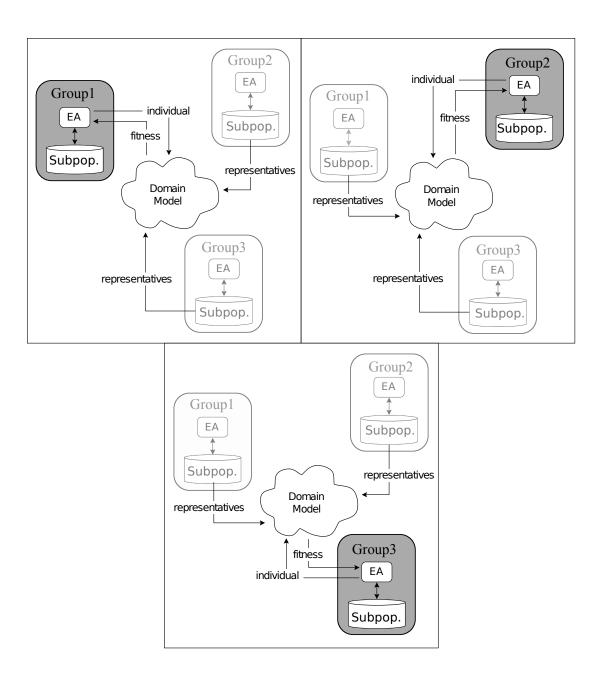


Figure 2.2: Demonstration for the Relationship between Phase and Cycle

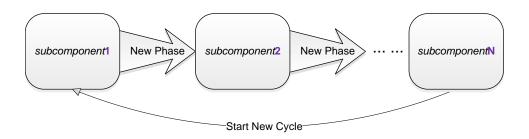


Figure 2.3: Demonstration for the Relationship between Phase and Cycle

```
Algorithm 2: \vec{best} \leftarrow \mathbf{basicCC}(NP)
                                                                                  (as introduced in [24])
1 (subpop, pop, best) \leftarrow initializeCC(NP);
  // Decomposition: implicitly performed based on separability
       assumption
2 while stoping criterion not met do // Optimization: Start a new cycle
      \mathbf{for}\ i \longleftarrow 1\,\mathbf{to}\,N\ \mathbf{do}\ /\!/\ \mathtt{Start}\ \mathtt{a}\ \mathtt{new}\ \mathtt{phase}
3
          for j \leftarrow 1 \text{ to } NP_i \text{ do // Collaboration}
4
             popcc_j \longleftarrow (best_1, \dots, best_{i-1}, subpop_{i,j}, best_{i+1}, \dots, best_N)
5
          (popcc, n\vec{e}w) \leftarrow \mathbf{optimizer}(popcc, i) / / \mathbf{Optimize} the i^{th}
6
               subcomponent
          subpop_i \longleftarrow popcc_i;
7
           best_i \leftarrow new_i;
9 return best;
```

3 Formulating Interdependencies among Subproblems

In this chapter, we revisit previous studies on the formulation of interdependency among subcomponent problem, show the crucial yet uncovered territory of research, and introduce our manner of formulation.

3.1 Background and Related Works

3.1.1 Classical Benchmark Problems

We firstly would like to review some classical problems for benchmarking the interdependencies identification process. One prominent example of this sort of problems is N-K landscapes. Finding the global optimum of such functions has been shown to be an NP-complete problem for $K \geq 2$ [26].

Another widely used problem class is k-ordered additively separable functions, defined by

$$f(s) = \sum_{i=1}^{m} f_i(s)$$
 (3-1)

where each subfunction $f_i(s)$ depends on at most k string positions, and the number of subfunctions that depend on a particular position j is at most one. [27]

3.1.2 Interdependencies among Subcomponents

In a variety of research fields related with evolutionary computation, subcomponents' interdependencies with respect to the contributions to fitness are essential issues. Lots of studies have done on the identification of such interdependencies:

1. In biological systems, which is the origin for the ideas of evolutionary computation, epistasis is the phenomenon that the effects of one gene can be modified by one or several other genes [3].

2. In the domain of binary-encoded genetic algorithms, linkage is commonly referred to situation where the contribution of a bit to the fitness function relies on other bits [27, 28, 29].

In practice, without prior knowledge linkage information, it is difficult to ensure that the coding scheme defined by the user always provides tight building blocks, although it is a key to the success of genetic algorithms. Therefore, it is highly critical to identify the linkage effectively.

However, we observe in numerical optimization, where the decision variables are encoded into real values, little has been reported on the independence analysis. Since it is an important issue across many subjects, this thesis would like to initialize the exploration on this topic. Especially, the driven force of our studies comes from large-scale numerical optimization. As stated before in Abstract part "divide-and-conquer" is favored as promising approach towards tackling large-scale and complex problem [17]. And for the "divide-and-conquer" approach, how to decompose the problem has a great impact on the cooperative coevolutionary algorithms. In black box scenarios, which are often the cases in practice, identification of interdependencies appears to be helpful as well. In order to achieve the identification, we need to formulate the interdependency in the first place.

3.1.3 Limitations for previous Formulation of Interdependency in Cooperative Coevolution

The first attempt of formulating interdependency dates back to Weicker and Weicker's paper [30]. In paper [30], Weicker and Weicker proposes an approach for capturing interdependencies between variables. That is, when a CC-based algorithm is optimizing on dimension i, its fitness value $newval_i$ is evaluated by two different candidate solutions $cand_{best} = \langle d_1, ..., d_N \rangle$ and $cand_{rand} = \langle d'_1, ..., d'_N \rangle$. They can be defined by

$$d_{j} = \begin{cases} newval_{i}, & if j = i \\ bestval_{j}, & otherwise \end{cases}$$
 (3-2)

$$d'_{j} = \begin{cases} newval_{i}, & if j = i \\ randval_{k}, & if j = k \\ bestval_{j}, & otherwise \end{cases}$$

$$(3-3)$$

where $bestval_j$ is the vector value of the best individual for dimension j, $randval_k$ is the vector value of a random chosen individual for dimension k, k is distinct from i and the random chosen individual should be distinct from the best individual. If $cand_{random}$ possesses a better fitness than candbest, a counter for the link(k,j) is increased accordingly. At end of each cycle, two subpopulations are merged where the corresponding counter achieve the maximal value of all links under consideration.

However, paper [30] reports a couple of the weaknesses of the foregoing interdependency learning mechanism:

- 1. Learning process is not supposed to start too early, as this would lead to the merging of subpopulations with non-interacting dimensions. In other words, their mechanism for learning variable interdependencies is likely to commit the type-II error, i.e., reports interdependency that is actually non-existing in reality. Also, when to start the learning process is hard to determine beforehand.
- 2. The maximal variables of a subpopulation allowed to include is limited to be 2. As a result, the learning process is incapable to handle problems with the interdependencies of the order higher than 2.
- 3. The calibration of the acceptance condition for the variable interdependencies is very sensible.

Some following works [31, 32, 33] intend to tackle the merging of the non-interacting sub-population by allowing the splitting merged subpopulation. They are, however, just a remedy for the type-II error that the learning process commits. The internal cause for committing type-II error is still left to uncover.

New Formation that Never Commits the Type-II Error

All weaknesses described in Subsection 3.1.3 render the learn process in [30] ineffective in practice. Towards uncovering the underlying reasons and overcoming these weaknesses, we propose a new definition that: 1) never commits the type-II error for any additively separable function; 2) is capable to deal with functions containing any number of decision variables; 3) requires no particular acceptance condition which might be problem-specific.

De Jong et al. [34] explain that "two variables in a problem are interdependent if the fitness contribution or optimal setting for one variable depends on the setting of the other variable". Here in this thesis, we refer such relationship between variables on real-value encoded numerical optimization on continuous space as Variable Interaction from now on.

In the first place, we provide the reasons why Weicker and Weicker's learning process is likely to commit the type-II error. To begin with, we observe that the presentation of Equation 3-2 and 3-3 are inherently flawed. As "bestval_i is the vector value of the best individual for dimension j," the vector "best val" is not necessarily to be fixed between the optimization on the i^{th} and the k^{th} variables. Thus, we should use distinct presentation for the two vectors. The revised ones for describing the two candidate solutions are listed below in Equation 3-4 and Equation 3-5:

$$d_{j} = \begin{cases} newval_{i}, & if j = i \\ bestval_{j}, & otherwise \end{cases}$$
 (3-4)

$$d_{j} = \begin{cases} newval_{i}, & if j = i \\ bestval_{j}, & otherwise \end{cases}$$

$$d'_{j} = \begin{cases} newval_{i}, & if j = i \\ randval_{k}, & if j = k \\ bestval'_{j}, & otherwise \end{cases}$$

$$(3-4)$$

Proposition 3.2.1. There exists chance when Weicker and Weicker's learning process [30] reports non-existing variable interactions.

Proof. Let 1^{th} and 3^{th} variables be the two under consideration for interaction. And without loss of generality, assume that only the 1^{th} and the 2^{th} variables are interacting. Suppose that dimensions are optimized with sequence $\{1,2,3\}$, and the learning process is now working on the 3^{th} variable. The two corresponding candidate solutions can be given in Equation 3-6 and Equation 3-7:

$$\vec{cand}_{best} = \langle bestval_1, bestval_2, newval_3 \rangle$$
 (3-6)

$$\vec{cand}_{random} = \langle randval_1, bestval'_2, newval_3 \rangle$$
 (3-7)

Since the 1^{th} and the 2^{th} variables are interacting, it is likely that Equation 3-8 and Equation 3-9 hold true simultaneously.

$$f(bestval_1, bestval_2, bestval_3) < f(randval_1, bestval_2, bestval_3)$$
 (3-8)

$$f(bestval_1, bestval'_2, bestval_3) > f(randval_1, bestval'_2, bestval_3)$$
 (3-9)

On the other hand, the value setting of independent 3^{th} variable has no impact on the relationship given in 3-9, hence Equation 3-10 holds.

$$\vec{cand}_{best} = f(bestval_1, bestval_2', newval_3) >$$

$$\vec{cand}_{random} = f(randval_1, bestval_2', newval_3)$$
(3-10)

which leads Weicker and Weicker's learning process to report the non-existing interaction between the 1^{th} and the 3^{th} variables.

Next, let's take a look at the definition of separable function for separable function given by CEC'2010 Special Session on Large-Scale Global Optimization [8]. A function f is separable according to [8] if Equation 3-11 holds, i.e., if its global optimum can be reached by successive line search along the axes.

$$\arg\min_{(x_1,\dots,x_N)} f(x_1,\dots,x_N) = \left(\arg\min_{(x_1)} f(x_1,\dots),\dots,\arg\min_{(x_N)} f(\dots,x_N)\right)$$
(3-11)

Therefore, if a certain function is *not separable*, there must be interactions between at least two variables in the decision vector. Motivated by this, we provide definition of *interaction*: two decision variables i and j are *interacting* if there is a decision vector \vec{x} whose i^{th} and j^{th} variables can be substituted with

values x_i' and x_j' so that Equation 3-12 holds,

$$\exists \vec{x}, x_i', x_j' : (f(x_1, \dots, x_i, \dots, x_j, \dots, x_N) < f(x_1, \dots, x_i', \dots, x_j, \dots, x_N)) \land (f(x_1, \dots, x_i, \dots, x_j', \dots, x_N) > f(x_1, \dots, x_i', \dots, x_j', \dots, x_N))$$
(3-12)

Recall that the interaction we discuss here is about the contribution of one variable to the fitness affected by value setting of another variable. So when considering the interaction between a pair of variables, say the i^{th} and the j^{th} variables, all the rest variables should *keep fixed*. As we can see from the Equation 3-12, our definition strictly follow the rule, making the learning mechanism based on the new definition never commit a single type-II error. Empirical verification of this claim will be shown latter in the experimental study 5.3.

4 Impact of Problem Decomposition Strategy on Cooperative Coevolution

In this chapter, we are going to empirically analyze the impact of problem decomposition strategy on the success of cooperative coevolutionary algorithms (CCEAs). Many studies [3, 17, 24, 31, 32, 35, 36, 37, 38, 39, 40] have stated the ideal problem decomposition strategy should categorize the mutually tightly interacting variables into the same subproblem, while keeping the interdependencies weak among distinct subcomponents. Thus they paid great efforts on getting to this "optimal" decomposition strategy as close as possible. However, none of them has ever investigated on what exactly will a optimal decomposition maintain the global search ability of CCEAs, how much truly will an inferior decomposition approach will trap CCEAs into local optima.

By incorporating prior knowledge which varies from none to full, influence of decomposition strategy from inferior to superior can be simulated. Thereby, we expect to gain helpful insight from the experimental results in this chapter.

4.1 Interpretation of Problem Decomposition Strategy

Based on the definition of variable interaction given in Equation 3-12, we are going to interpret problem decomposition strategy as a binary array \vec{A}_{intr} with size of $\frac{N \times (N-1)}{2}$, where N is number of decision variables. Each element in \vec{A}_{intr} represents the relationship between a pair of decision variables: 1 for interacting and 0 for non-interacting. P_{prior} is the percentage of \vec{A}_{intr} 's randomly selected elements filled with the correct values (either 1 or 0). And all the rest elements are left with 0, the rationality for consistently choosing 0 will be discussed later in detailed in Section 4.2. We can translate the \vec{A}_{intr} array into a number of non-overlapping groups by merging interacting variables into same groups. Every group contains at least one decision variables and stand for one subcomponent of problem. This is how we bridge the variable interactions into subcomponents in CCEAs. Then we can conduct experiment study.

	Function Name	Sep	MultiModal
f_{14}	20-group Shifted 50-rotated Elliptic Function	No	No
f_{15}	20-group Shifted 50-rotated Rastrigin's Function	No	Yes
f_{16}	20-group Shifted 50-rotated Ackley Function	No	Yes
f_{17}	20-group Shifted 50-rotated Schwefel's Function	No	No

Table 4.1: Four $\frac{D}{m}$ -group m-nonseparable functions

4.2 How to Handle Unknown Elements in \vec{A}_{intr}

As stated in Section 2, randomly known elements in \vec{A}_{intr} are filled with correct values, either 1 or 0. Now the question is, how to tackle with all rest elements? Note that the P_{prior} is a controlling parameter for tuning how corresponding decomposition strategy close to the optimal one. In other words, what truly matters is the decomposition strategy that the prior interaction knowledge leads to. Since it is difficult to quantify the decomposition strategy itself, we apply the dynamics of quantity to the percentage of prior interaction knowledge instead. Conventional CCEAs usually treat each decision variable as a single subpopulation [3, 12, 16, 24, 38, 41]. Filling unknown decision variables with 0 is the best way to simulate the situation when conventional decomposition strategies approach the optimal one gradually. Additionally, our interaction learning approach, which is going to be discussed in Subsection 5.2.1, is a bottom-up approach and also considers variables that not having been found interaction as independent ones.

4.3 Experimental Studies

4.3.1 Experimental Setup

• Benchmark set: Four $\frac{D}{m}$ -group m-nonseparable functions (see Table 4.1) proposed in CEC'2010 Special Session [8]. (D=1000, m=50). They are listed in Table 4.2.

- Optimizer: JADE [42] with the modifications introduced in [35], for plugging the raw algorithm into CC.
- Maximal fitness evaluation for each run: 3.0e + 06
- Results were collected based on 25 independent runs on each function.

4.3.2 Experimental Results with P_{prior} varying from 0% to 100%

Intuitively, we set the range of P_{prior} from 0% to 100%, since the dynamics of CCEA's performance when prior information ranges from none to full are supposed to be explored. Limited by computational resources, we divide the interval [0%,100%] by 10. Comprehensive experimental results are presented in Table 4.2.

Experimental results implies: incorporating prior grouping knowledge do enhance the performance of CCEA remarkably. However, the increase of P_{prior} appears to be useful only at the interval [0%, 10%]. The underlying reason for this a bit counterintuitive result will be uncovered in Subsection 4.3.4. The results clearly shows that the sampling points of P_{intr} 's value should be focus on the interval of [0%, 10%], in order to provide more fine-grain insight into the dynamics of CCEA's performance.

4.3.3 Experimental Results with P_{prior} varying from 1% to 9%

In the subsection, the step size of sampling point is decrease to be 1%, and the range of interest is restricted in [1%, 9%]. Experimental results are offered in Table 4.3.

Experimental results in Table 4.3 verifies our hypothesis perfectly. It can be observed that, making use of only around 10% of variable interaction knowledge is able to make the CCEA at one magnitude closer in fitness to the optimal when termination is issued.

-		Percentage of Prior Interaction Knowledge										
	Prob.	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
	min	4.71e + 08	5.99e+06	4.78e + 06	4.73e + 06	5.58e + 06	4.31e+06	5.31e+06	5.14e+06	4.46e + 06	4.85e + 06	4.67e + 06
	7_{th}	5.64e + 08	6.99e + 06	$5.42\mathrm{e}{+06}$	5.33e + 06	$6.25\mathrm{e}{+06}$	5.17e + 06	$5.93\mathrm{e}{+06}$	$5.65\mathrm{e}{+06}$	$5.27\mathrm{e}{+06}$	$5.62\mathrm{e}{+06}$	$5.24\mathrm{e}{+06}$
	med.	5.97e + 08	7.63e + 06	$6.18\mathrm{e}{+06}$	$5.80\mathrm{e}{+06}$	$6.67\mathrm{e}{+06}$	$5.66\mathrm{e}{+06}$	$6.49\mathrm{e}{+06}$	$5.97\mathrm{e}{+06}$	$5.99\mathrm{e}{+06}$	$6.32\mathrm{e}{+06}$	$5.64\mathrm{e}{+06}$
f_{14}	19_{th}	6.29e+08	8.14e + 06	$6.42\mathrm{e}{+06}$	6.23e + 06	7.10e+06	$5.98\mathrm{e}{+06}$	$6.97\mathrm{e}{+06}$	$6.37\mathrm{e}{+06}$	$6.45\mathrm{e}{+06}$	$6.75\mathrm{e}{+06}$	6.23e + 06
	max	7.38e + 08	$1.09\mathrm{e}{+07}$	7.73e + 06	7.35e + 06	7.94e + 06	6.37e + 06	$7.40\mathrm{e}{+06}$	7.63e + 06	7.95e + 06	$9.10\mathrm{e}{+06}$	7.27e + 06
	mean	5.97e + 08	7.73e+06	6.03e + 06	5.81e + 06	6.65e + 06	5.54e + 06	6.47e + 06	5.98e + 06	5.99e + 06	6.31e + 06	5.78e + 06
	std	5.98e + 07	1.17e + 06	6.53e + 05	7.30e + 05	6.88e + 05	5.79e + 05	6.43e + 05	5.79e + 05	9.15e + 05	1.07e + 06	7.16e + 05
	min	1.33e+04	1.34e+03	1.43e+03	1.47e + 03	1.43e+03	1.39e+03	1.46e + 03	1.44e+03	1.44e+03	1.44e + 03	1.36e + 03
	7^{th}	1.39e + 04	1.58e + 03	1.53e+03	1.52e + 03	1.57e + 03	1.54e + 03	1.52e + 03	1.54e + 03	1.51e + 03	$1.48\mathrm{e}{+03}$	1.55e + 03
	med.	1.42e+04	1.63e + 03	1.56e + 03	1.56e + 03	1.63e+03	1.58e + 03	1.56e + 03	1.57e + 03	1.55e + 03	1.55e + 03	1.60e + 03
f_{15}	19^{th}	1.45e + 04	1.65e + 03	1.60e + 03	1.62e+03	1.70e + 03	1.71e + 03	1.62e+03	1.63e + 03	$1.60\mathrm{e}{+03}$	1.61e + 03	$1.65\mathrm{e}{+03}$
	max	1.49e+04	1.81e + 03	$1.69\mathrm{e}{+03}$	1.74e + 03	$1.85\mathrm{e}{+03}$	$1.76\mathrm{e}{+03}$	$1.78\mathrm{e}{+03}$	$1.78\mathrm{e}{+03}$	$1.85\mathrm{e}{+03}$	$1.74\mathrm{e}{+03}$	1.81e + 03
	mean	1.42e+04	1.61e+03	1.56e+03	1.58e + 03	1.63e+03	1.60e+03	1.58e + 03	1.58e + 03	1.56e + 03	1.56e + 03	1.60e+03
	std	4.40e+02	$9.29e{+01}$	6.81e + 01	7.94e+01	9.37e + 01	1.08e + 02	8.25e + 01	7.78e + 01	8.84e + 01	8.48e + 01	9.98e + 01
	min	3.90e+02	9.95e-01	1.90e-13	2.37e-01	2.26e-13	1.83e-01	2.36e-13	1.73e-01	1.83e-13	2.01e-13	2.36e-01
	7^{th}	3.95e+02	1.76e + 00	2.38e-01	2.77e-01	2.41e-01	2.64e-01	2.34e-01	2.44e-01	2.68e-01	2.34e-01	2.63 e-01
	med.	3.96e+02	2.32e+00	3.95 e- 01	4.02e-01	2.78e-01	3.26e-01	3.82e-01	2.93e-01	3.34e-01	3.50 e-01	3.82 e- 01
f_{16}	19^{th}	3.96e + 02	3.62e+00	4.39e-01	4.16e-01	4.20e-01	4.46e-01	4.33e-01	4.24 e - 01	4.12e-01	4.05 e-01	4.37e-01
	max	3.97e + 02	7.07e+00	4.16e+00	4.92e-01	2.41e+00	5.15 e-01	2.18e+00	4.80 e-01	1.03e+00	4.67e-01	1.56e+00
	mean	3.95e+02	2.83e + 00	4.84 e-01	3.64 e-01	3.92 e- 01	3.54 e-01	3.91 e- 01	3.24 e-01	3.54 e- 01	3.12e-01	4.06 e - 01
	std	1.36e+00	1.54e+00	7.76e-01	8.36e-02	4.35e-01	9.69e-02	3.91e-01	9.79e-02	1.87e-01	1.16e-01	2.54e-01
	min	8.70e+05	3.75e+02	2.04e-04	6.81e-05	1.70e-04	2.96e-04	4.16e-04	6.49e-05	8.74e-05	2.67e-04	1.22e-04
	7^{th}	9.54e + 05	$1.43e{+03}$	3.29 e - 03	3.23 e-03	5.77e-03	9.27e-03	$4.45\mathrm{e}\text{-}02$	3.24 e - 03	2.22e-03	1.90e-03	2.46 e - 03
		1.01e+06	2.02e+03	$4.06\mathrm{e}\text{-}02$	$6.46\mathrm{e}{+00}$	$1.55\mathrm{e}{+01}$	$5.31\mathrm{e}{+00}$	$2.49\mathrm{e}{+01}$	$1.56\mathrm{e}\text{-}01$	$3.18\mathrm{e}{+00}$	$6.03\mathrm{e}\text{-}01$	2.10e+01
f_{17}	19^{th}	1.10e + 06	3.87e + 03	6.21e+00	$6.85\mathrm{e}{+01}$	$9.20\mathrm{e}{+01}$	$6.30e{+01}$	7.73e + 01	$8.31e{+00}$	7.18e + 01	$8.52e{+00}$	8.61e + 01
	max	1.26e + 06	5.59e + 03	1.44e + 02	2.23e+02	3.87e + 02	1.53e + 02	2.30e+02	9.39e + 01	2.17e+02	5.06e + 02	2.53e+02
	mean	1.03e + 06	2.50e + 03	1.60e + 01	4.86e + 01	$6.\overline{47e+01}$	$3.\overline{34e+01}$	5.32e + 01	$9.\overline{51e+00}$	$4.\overline{17e+01}$	4.29e+01	6.15e + 01
	std	1.01e+05	1.55e+03	3.82e+01	7.49e+01	1.01e+02	4.77e + 01	7.05e+01	2.04e+01	6.54e + 01	1.16e+02	7.63e+01

Table 4.2: Result of experimental studies with P_{prior} varying from 0% to 100%. Numbers in the first line stand for the percentage of given interaction information. The results for the *best*, 7^{th} , median, 14^{th} , worst runs as well as the mean and standard deviation over 25 independent runs are presented.

		Percentage of Prior Interaction Knowledge										
	Prob.	1%	2%	3%	4%	5%	6%	7%	8%	9%		
	min	3.87e + 08	2.87e + 08	1.83e + 08	8.71e+07	3.41e+07	2.53e+07	1.41e+07	6.40e + 06	7.49e + 06		
	7^{th}	4.26e + 08	3.19e + 08	2.05e + 08	1.15e + 08	4.41e+07	3.03e+07	$1.61\mathrm{e}{+07}$	7.64e + 06	9.16e + 06		
	med.	4.32e + 08	3.34e + 08	2.20e+08	1.24e + 08	4.99e + 07	3.34e + 07	1.73e + 07	8.08e + 06	1.12e+07		
f_{14}	19^{th}	4.63e + 08	$3.51\mathrm{e}{+08}$	2.25e + 08	1.34e + 08	5.52e + 07	3.62e + 07	$1.86\mathrm{e}{+07}$	8.75e + 06	1.35e + 07		
	max	4.89e + 08	3.98e + 08	2.71e + 08	1.72e + 08	6.56e + 07	4.07e + 07	2.83e + 07	1.01e + 07	1.70e + 07		
	mean	4.41e + 08	$3.38e{+08}$	2.19e + 08	1.24e + 08	4.97e + 07	3.34e + 07	1.75e + 07	8.16e + 06	1.14e + 07		
	std	2.68e + 07	2.80e + 07	2.18e + 07	1.81e + 07	7.66e + 06	3.92e + 06	2.85e + 06	8.45e + 05	2.77e + 06		
	min	1.10e+04	8.09e+03	6.52e + 03	4.23e+03	2.62e+03	2.01e+03	1.58e + 03	1.64e + 03	1.47e + 03		
	7^{th}	1.14e+04	8.77e + 03	6.90e + 03	4.47e + 03	2.91e+03	2.12e+03	1.77e + 03	1.74e + 03	1.60e + 03		
	med.	1.17e + 04	9.11e+03	7.07e + 03	4.65e + 03	3.02e+03	2.20e+03	1.82e + 03	1.82e + 03	1.66e + 03		
f_{15}	19^{th}	1.20e+04	9.36e + 03	7.31e+03	$4.83e{+03}$	3.13e+03	2.27e + 03	$1.90e{+03}$	1.88e + 03	1.72e + 03		
	max	1.32e + 04	9.95e + 03	7.91e + 03	4.90e+03	3.40e+03	2.47e + 03	2.11e+03	2.20e+03	1.81e+03		
	mean	1.18e + 04	9.09e + 03	7.14e + 03	$4.64e{+03}$	3.03e + 03	$2.21e{+03}$	$1.84e{+03}$	1.82e + 03	1.66e + 03		
	std	4.42e + 02	4.78e + 02	3.52e + 02	2.09e+02	2.03e+02	1.27e + 02	1.22e+02	1.25e + 02	7.66e + 01		
	min	3.94e + 02	3.47e + 02	1.41e+02	4.43e+01	3.27e+01	1.82e + 01	1.25e+01	2.87e + 00	3.21e+00		
	7^{th}	4.00e+02	3.64e + 02	1.82e + 02	$4.86e{+01}$	$3.58e{+01}$	2.70e+01	1.47e + 01	6.32e+00	4.35e+00		
	med.	4.02e+02	3.73e + 02	1.98e + 02	5.19e+01	$3.80e{+01}$	$2.88e{+01}$	1.75e + 01	7.72e + 00	4.84e + 00		
f_{16}	19^{th}	4.03e + 02	3.81e + 02	2.14e+02	5.76e + 01	$4.10e{+01}$	$3.06e{+01}$	$1.90e{+01}$	9.76e + 00	6.16e + 00		
	max	4.05e + 02	3.97e + 02	2.27e + 02	7.04e+01	5.36e + 01	3.36e+01	2.31e+01	1.28e + 01	1.33e+01		
	mean	4.01e+02	3.73e + 02	1.94e + 02	5.33e+01	$3.90e{+01}$	2.85e + 01	1.72e+01	7.85e + 00	5.36e + 00		
	std	2.61e+00	1.30e+01	2.27e+01	6.66e + 00	4.64e+00	3.22e+00	2.84e+00	2.51e+00	1.99e+00		
	min	6.89e + 05	4.37e + 05	3.38e + 05	1.83e + 05	8.03e+04	3.50e + 04	1.71e+04	1.57e + 04	2.34e+03		
	7^{th}	7.77e + 05	5.31e + 05	4.06e + 05	2.10e+05	1.05e + 05	3.98e + 04	2.45e+04	2.37e + 04	5.81e + 03		
	med.	8.25e + 05	5.82e + 05	$4.33e{+05}$	2.37e + 05	1.17e + 05	4.24e+04	2.81e + 04	2.81e + 04	8.67e + 03		
f_{17}	19^{th}	8.63e + 05	6.62e + 05	4.69e + 05	2.49e + 05	1.35e + 05	5.18e + 04	3.69e + 04	$3.30e{+04}$	1.05e + 04		
	max	1.02e + 06	7.38e + 05	5.69e + 05	2.97e + 05	1.57e + 05	6.47e + 04	4.83e + 04	4.28e + 04	1.35e + 04		
	mean	8.26e + 05	5.97e + 05	4.42e + 05	2.31e + 05	1.17e + 05	4.64e + 04	2.98e + 04	2.88e + 04	8.13e + 03		
	std	8.07e + 04	8.02e+04	5.53e + 04	2.93e+04	1.97e + 04	8.70e + 03	7.70e + 03	7.47e + 03	2.82e + 03		

Table 4.3: Result of experimental studies with P_{prior} varying from 1% to 9%. Numbers in the first line stand for the percentage of given interaction information. The results for the *best*, 7^{th} , median, 14^{th} , worst runs as well as the mean and standard deviation over 25 independent runs are presented.

		Percentage of Prior Interaction Knowledge									
	Prob.	0%	1%	3%	5%	7%	9%	100%			
	mean	5.97e + 08	4.41e+08	2.19e+08	4.97e + 07	1.75e + 07	1.14e+07	5.78e + 06			
f_{14}	std	5.98e + 07	2.68e + 07	2.18e + 07	7.66e + 06	2.85e + 06	2.77e + 06	7.16e + 05			
	group	1000	745	345	120	60	30	20			
	mean	1.42e+04	1.18e + 04	7.14e + 03	3.03e+03	1.84e + 03	1.66e + 03	1.60e + 03			
f_{15}	std	4.40e+02	4.42e+02	3.52e + 02	2.03e+02	1.22e+02	7.66e + 01	9.98e + 01			
	group	1000	764	346	122	46	30	20			
	mean	3.95e+02	4.01e+02	1.94e + 02	3.90e+01	1.72e+01	5.36e+00	4.06e-01			
f_{16}	std	1.36e+00	2.61e+00	2.27e + 01	4.64e + 00	2.84e + 00	1.99e+00	2.54 e-01			
	group	1000	773	316	122	53	28	20			
	mean	1.03e+06	8.26e + 05	4.42e + 05	1.17e + 05	2.98e + 04	8.13e+03	6.15e+01			
f_{17}	std	1.01e+05	8.07e + 04	5.53e + 04	1.97e + 04	7.70e + 03	2.82e + 03	7.63e + 01			
	group	1000	756	305	108	51	26	20			

Table 4.4: Impact of Problem Decomposition Strategy on the Success of CCEA. Group stands for the number of subpopulations. The mean and standard deviation over 25 independent runs are presented.

4.3.4 Insights into the Underlying Reasons

This subsection is going to provide some insights into the dynamics of the performance of CCEA by associating P_{intr} , result of solutions and problem decomposition strategies all together in Table 4.4. Experimental results show that:

- 1. Given $P_{intr} = 9\%$, the corresponding decomposition strategies, which yield around 25 subpopulations, is already quite close to the optimal one that is supposed to have 20 subpopulations. It is the reason why further increase of P_{intr} make little progress in enhancing the performance of CCEA, as shown previously in Table 4.2.
- 2. The relationship between P_{intr} and the number of subpopulations is nonlinear. More precisely, the speed of merging subpopulations decease greatly as the P_{intr} . Figure 4.1 to Figure 4.4 illustrate the relationship. The empirical

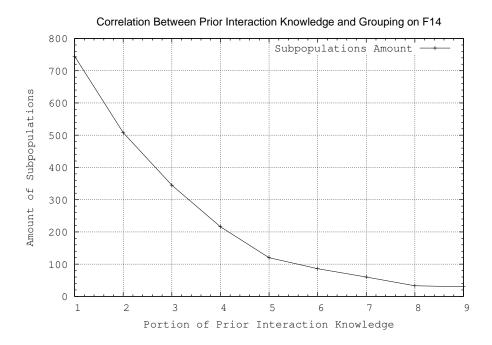


Figure 4.1: Decrease of the Number of Groups as the P_{intr} increases for f_{14}

data perfectly matches our expectation: merging subpopulations gradually reduces the number of undiscovered interactions, which in return, makes it harder to further merge more subpopulations.

- 3. The experimental data clearly show subpopulations gravitate towards broad suboptimal peaks surrounding Nash Equilibria [43] in the joint space.
- 4. Being closer towards the optimal decomposition strategy enhance the performance of CCEA gradually. Convergence curves for all four functions provided from Figure 4.5 to Figure 4.8.

As the comprehensive empirical studies have been conducted to verify the impact of a good problem strategy on the success of CCEA. A novel variable interaction learning mechanism is proposed in the next chapter: Chapter 5, incorporating the mechanism into CCEA yields excellent performance.

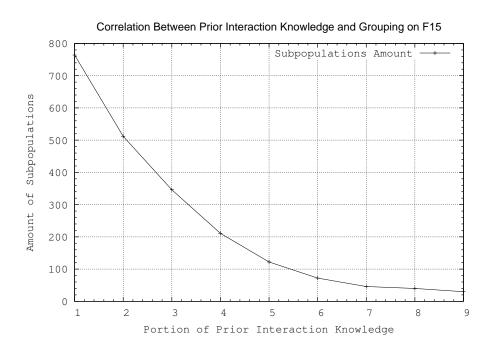


Figure 4.2: Decrease of the Number of Groups as the P_{intr} increases for f_{15}

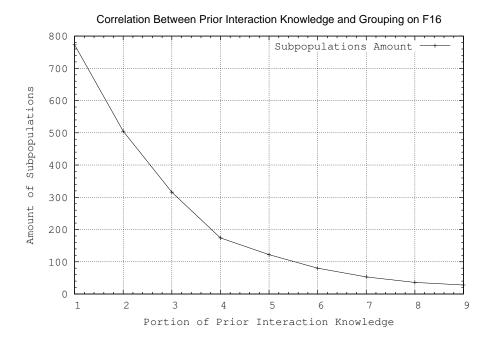


Figure 4.3: Decrease of the Number of Groups as the P_{intr} increases for f_{16}

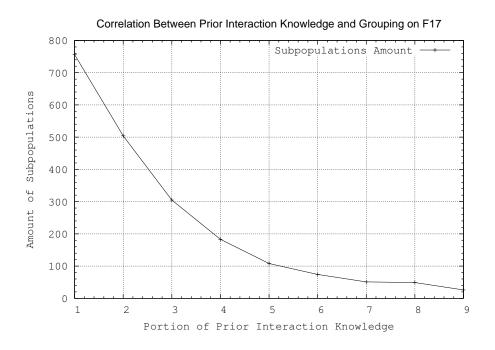


Figure 4.4: Decrease of the Number of Groups as the P_{intr} increases for f_{17}

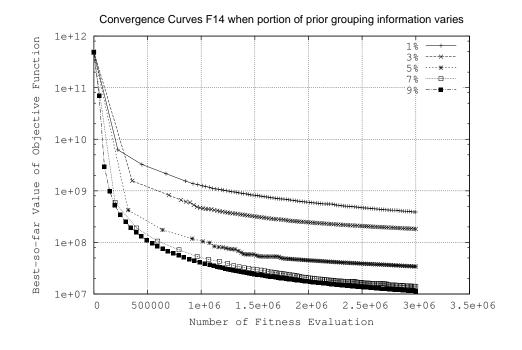


Figure 4.5: Convergences curves of f_{14}

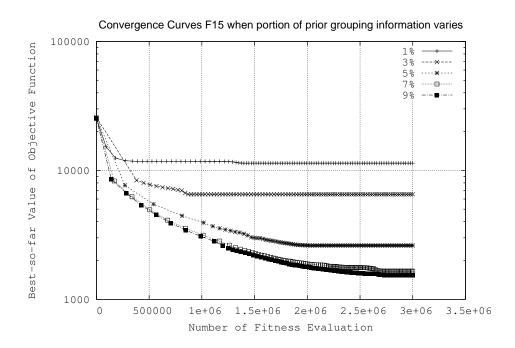


Figure 4.6: Convergences curves of f_{15}

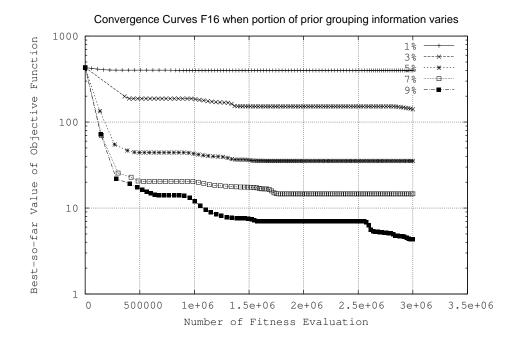


Figure 4.7: Convergences curves of f_{16}

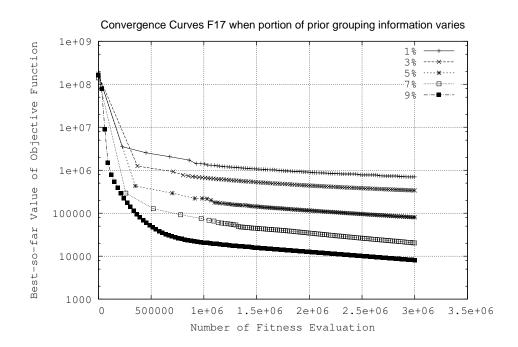


Figure 4.8: Convergences curves of f_{17}

5 Enable Variable Interaction Learning in Cooperative Coevolutionary Algorithms

By introducing learning stage into Cooperative Coevolution (CC), we develop an highly effective CC-based algorithms. The learning stage is basis of the definition of Variable Interaction given in Equation 3-12.

5.1 Background

Evolutionary Algorithms (EAs) have been widely applied for solving both numerically and combinational optimization tasks [44]. Finding solutions for a problem usually becomes harder when the number of decision variables increases because of the curse of dimensionality. As a consequence, [16] reports that there is a rapid decline in performance of conventional EAs when dealing with large-scale problems. In order to improve the ability to solve high-dimensional optimization tasks, [45] proposes a co-evolution approach for combinational optimization problems and [24] further generalizes the approach to a universal framework: Cooperative Coevolution (CC). CC algorithms tackle the curse of dimensionality with a divide-and-conquer method which separates the search space into subspaces of lower dimensionality. They therefore decompose the decision vector into groups of variables which can be optimized cooperatively in cycles. After each cycle, the information gained in the separate optimization steps is joined for the next iteration. This approach yields good performance both in benchmark problems and real-world applications [24, 25]. Despite this success, for the most important part of CC, the problem decomposition strategy, no satisfying solution has yet been developed. In this thesis, we introduce a general, scalable, and highly efficient method for this purpose, called Cooperative Coevolution with Variable Interaction Learning (CCVIL).

The rest of this chapter is organized as follows. In the next section, we give a short outline of the cooperative cooperationary idea in general and list related work in the area of problem decomposition in CC. CCVIL is then discussed in detail in Section 5.2 and experimentally studied by using a set of twenty large-scale

benchmark problems in Section 5.3. It achieves excellent results in these experiments and frequently outperforms two related, state-of-the-art CC techniques as well as the optimizer which it uses internally. We finally conclude our thesis in Section 5.4 where we also give pointers to future work.

5.1.1 Discovering Decision Variable Interactions

The decomposition strategy that identifies interacting decision variables and divides the search space into subspaces of lower dimensionality is the most important component of CC algorithms. A function f is separable according to [8] if Equation 5-1 holds, i.e., if its global optimum can be reached by successive line search along the axes. Therefore, if a certain function is not separable, there must be interactions between at least two variables in the decision vector. Motivated by this, we provide definition of *interaction*: two decision variables i and j are interacting if there is a decision vector \vec{x} whose i^{th} and j^{th} variable can be substituted with values x'_i and x'_j so that Equation 5-2 holds. A simple application of Equation 5-2 to two-dimensional Schwfel's problem 1.2 is illustrated in Figure 5.1

$$\arg\min_{(x_1,\dots,x_N)} f(x_1,\dots,x_N) = \left(\arg\min_{(x_1)} f(x_1,\dots), \dots, \arg\min_{(x_N)} f(\dots,x_N)\right)$$
(5-1)

$$\exists \vec{x}, x_i', x_j' : (f(x_1, \dots, x_i, \dots, x_j, \dots, x_N) < f(x_1, \dots, x_i', \dots, x_j, \dots, x_N)) \land (5-2)$$

$$(f(x_1, \dots, x_i, \dots, x_i', \dots, x_N) > f(x_1, \dots, x_i', \dots, x_i', \dots, x_N))$$

The idea behind the decomposition of the decision variables in CC into groups G_1, G_2, \ldots, G_m is that the fitness function f can be approximated as a linear combination of component functions f_1, f_2, \ldots, f_m . The domains of the functions f_i have the lower dimensionality $|G_i| < N$ since their results only depend on the variables in the corresponding group G_i . Although it is usually only possible to compute f but not its components f_i , the knowledge that the groups G_i can be optimized separately can speed up the optimization process significantly.

For discovering interactions between variables in the *decomposition* step, a simple method is suggested in [30]: Assume that \vec{best} would be the vector of the best values for each decision variable discovered so far. After coevolutionary

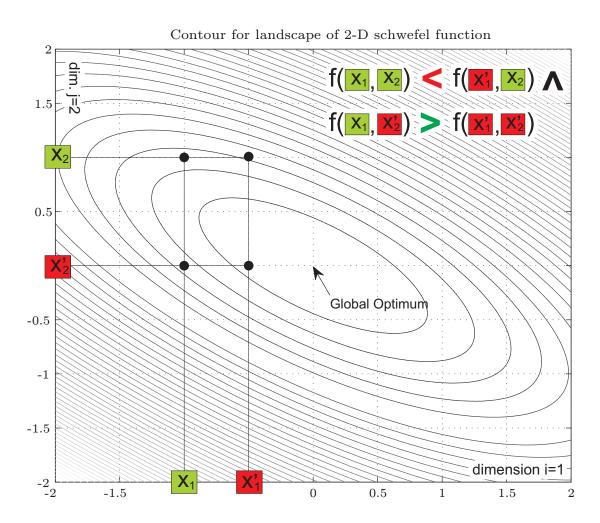


Figure 5.1: Application of Equation 5-2 to two-dimensional Schwfel's problem 1.2

$$x_{j} = \begin{cases} new_{i} & \text{if } j = i \\ best_{j} & \text{otherwise} \end{cases}$$
 (5-3)
$$x'_{j} = \begin{cases} new_{i} & \text{if } j = i \\ rand_{k} & \text{if } j = k \\ best_{j} & \text{otherwise} \end{cases}$$
 (5-4)

optimizing dimension i, the best individual $n\vec{e}w$ in the population popcc of the optimizer only focusing on dimension i is extracted as well as a random candidate $r\vec{a}nd$ from the global population pop (with $n\vec{e}w \neq r\vec{a}nd \neq b\vec{e}st$). Based on these three vectors, two new candidate vectors \vec{x} and \vec{x}' are created according to Equations 5-3 and 5-4 for testing whether dimensions i and k interact. The value at index k of \vec{x} is better than the k^{th} value of \vec{x}' , since it comes from the vector of best known values $b\vec{e}st$ whereas x'_k stems from the random population member $r\vec{a}nd$. Optimizing dimension i should not influence this relation and $f(\vec{x}) \leq f(\vec{x}')$ would hold if the variables were separable. Therefore, if $f(\vec{x}') < f(\vec{x})$, i.e., \vec{x}' is better than \vec{x} , there likely is an interaction between dimension i and k [30].

5.1.2 Related Works

In the early stage of the development of CC, researchers mainly adopted two simple decomposition methods: one-dimension based and splitting-in-half methods [24, 38]. These two methods do not take the interaction between components into consideration. Thus, they cannot solve problems consisting of non-trivial variable interactions.

In order to mitigate this problem, a multi-level pyramidal genetic algorithm is utilized in [46] to better deal with multiple-choice scheduling. In the area of numerical optimization, Yang et al. proposed two effective CC-based algorithms, Differential Evolution using Cooperative Coevolution with adaptive grouping strategy (DECC-G) [17] and Multilevel Cooperative Coevolution (MLCC) [40]. DECC-G uses a constant group size, for instance 100, and randomly decomposes the high-dimensional variable vector into several such groups. These are then optimized with a certain EA. DECC-G with a small group size works properly for separable problems while highly nonseparable problems can better be solved with large group size. The problem of DECC-G is that the best group size is not

known in advance. In order to overcome it, MLCC adopts a multilevel strategy for decomposition. It maintains a decomposer pool from which decomposers with different group sizes are selected depending on the problem under investigation and the stage of the evolution. For nonseparable problems, MLCC tends to select the decomposers with large group sizes. In the opposite case, MLCC prefers to choose the decomposers with smaller group sizes. However, determining a good pool of decomposers is hard in practice since the interaction between variables is usually not known beforehand. This, in turn, would lead to a waste of function evaluations.

By using the technique of learning variable interactions used in [30] and outlined here in Section 5.1.1 CC can become more adaptable. Nevertheless, the way it is used in [30] suffers severe shortcomings. For example, the maximum group size is limited to two, which rarely is the case real-world problems. Moreover, the *choice* of the dimensions i and k for interaction investigation (see Equations 5-3 and 5-4), frequently leads to the detection of non-existing interactions [30]. In [30], it is possible that dimension i and k are not optimized in successive phases. Thus, changes in other dimensions of \overrightarrow{best} may take place in the mean time which violates the condition of Equation 5-2.

CCVIL overcomes the problem of the DECC-G and the MLCC algorithm, i.e., the group sizing, by using the interaction learning method of [30]. The choice of the dimensions i and k for interaction detection, however, is conducted in a way which prevents the discovery of non-existing interactions.

5.2 Cooperative Coevolution with Variable Interaction Learning

We propose a novel CC-framework called *Cooperative Coevolution with Variable Interaction Learning* (CCVIL) with incremental group sizes for solving large-scale optimization problems of separable, partially-nonseparable, and nonseparable character. Instead of setting the group sizes as a constant or defining a set of values from which to choose them, we allow the optimization process to adapt them by learning the interaction between the decision variables. The whole pro-

cedure of CCVIL is divided into two stages, the *learning stage* and *optimization* stage executed once in exactly this sequence. Both stages are divided into cycles and phases, similar to the conventional CC framework (see Section 2.2).

5.2.1 Learning Stage

The learning stage of CCVIL optimizes one dimension after the other, similar to the simple CC approach given in Algorithm 2. While doing this, it only tests the currently and the previously optimized dimension for interaction. Since only these two dimensions changed between the application of the interaction detection mechanism of [30], Equation 5-2 can never be violated and only becomes true for real interactions. Therefore, the flaw of detecting non-existent interactions is avoided. Before each learning cycle, the order of visiting the dimensions is randomly permutated so that each two dimension have the same chance to be processed in a row. The details of the learning stage are presented in Algorithm 3.

The efficiency of CCVIL becomes clear from a stochastic point of view. The probability of placing any two (possibly interacting) dimensions i and j in an N-dimensional problem adjacently in one random permutation Π is 2/N. The probability $p_{capt}(K)$ that this happens in at least once in K learning cycles then is given in Equation 5-5.

$$p_{capt}(K) = 1 - (1 - 2/N)^K (5-5)$$

In a 1000-dimensional problem, the probability of putting any two (possibly interacting) variables adjacently in a random permutation and examining interaction between them during K = 500 cycles is already $p_{capt}(K) = 0.6325$ and raises to 0.7984 for 800 cycles. Given a limited number of fitness function evaluations, as many learning cycles as possible should thus be performed. Therefore, the population size and generation limit of the internal optimizer should be as small as possible during the learning stage (3 and 1, respectively, in this work). CCVIL issues an independent restart for each cycle to prevent possible loss of population diversity during the learning stage.

We furthermore set a lower and an upper threshold for the number of learning cycles: \check{K} and \hat{K} . If CCVIL cannot detect interactions between any two

Algorithm 3: $groupInfo \leftarrow$ Learning Stage of CCVIL

```
1 K \leftarrow 0;
 2 groupInfo \leftarrow \{\{1\}, \{2\}, \dots, \{N\}\}\}// initially assume full separability
                  // start a new learning cycle
        \Pi \leftarrow random permutation of dimension indices \{1, 2, \dots, N\};
        K \longleftarrow K + 1;
 \mathbf{5}
        lastIndex \longleftarrow 0;
 6
        (subpop, pop, \vec{best}) \leftarrow \mathbf{initializeCC}(3, 3, \dots, 3) / / \text{ use } NP_i = 3 \ \forall i \in 1..N
 7
        for i = 1 to N do // start a new learning phase, i.e., tackle next
 8
        dimension
            if lastIndex \neq 0 then
 9
                G_1 \longleftarrow \mathbf{find}(groupInfo, \Pi_i)
                                                               // find the group containing \Pi_i
10
                G_2 \leftarrow \mathbf{find}(\mathit{groupInfo}, \mathit{lastIndex}) \ / / \ \mathbf{find} \ \mathbf{group} \ \mathsf{of} \ \mathsf{last} \ \mathsf{optimized}
11
                     variable
            if i = 1 \lor (G_1 \neq G_2) then
12
                for j = 1 to NP do
13
                  | popcc_j \longleftarrow (best_1, \dots, best_{\Pi_i-1}, subpop_{\Pi_i, j}, best_{\Pi_i+1}, \dots) 
14
                (popcc, n\vec{e}w) \leftarrow \mathbf{optimizer}(popcc, \Pi_i) // any optimizer, we used JADE
15
                subpop_{\Pi_i} \longleftarrow popcc_{\Pi_i};
16
                best_{\Pi_i} \leftarrow new_{\Pi_i};
17
                if lastIndex \neq 0 then
18
                    Compose \vec{x} and \vec{x'} according to Equations 5-3 and 5-4;
19
                    if f(\vec{x}) < f(\vec{x'}) then // interaction between dim. i and lastIndex?
20
                       groupInfo \longleftarrow ((groupInfo \setminus \{G_1\}) \setminus \{G_2\}) \cup (\{G_1 \cup G_2\});
21
                lastIndex \longleftarrow \Pi_i // only test successively optimized dimensions
22
23 until (|groupInfo| = 1) \lor [(K > \check{K}) \land (|groupInfo| = N)] \lor (K > \hat{K});
24 return groupInfo // return the set of mutually separable groups of
        interacting variables
```

variables in the first \check{K} cycles of learning stage, it assumes that the problem is fully separable or that the interactions are rather weak and immediately switches to the optimization stage. Additionally, a transition to the optimization stage is enforced after \hat{K} cycles even if not all interactions have been discovered in order to limit the runtime used for learning. As default settings, we recommend 10 for \check{K} and to set \hat{K} to the number of cycles needed to achieve 80% for $p_{capt}(\hat{K})$ (see Equation 5-5). This number can be computed by $\hat{K} \geq \log(1-0.8)/\log(1-2/N)$. However, \hat{K} should not lead to a consumption of more than 60% of the function evaluations in the learning stage.

5.2.2 Optimization Stage

The user of our CC framework is completely free in the choice of the optimizer to be used for the variables grouped together. During both, the interaction learning and the optimization stage of CCVIL, we apply JADE for this purpose. JADE is an enhanced variant of Differential Evolution (DE) [47, 48] with improved speed and reliability compared with plain DE [42]. In each phase of the optimization stage of CCVIL, the optimizer is applied to the complete subspace defined by one group $G_i \in groupInfo$ (whereas the remaining variables of the candidate vectors are constant and correspond to the representatives from the other groups).

During the learning stage, CCVIL may divide the variables into groups of different sizes. In the optimization step, the population size NP_i and number of generations Gen_i granted to the optimizer for processing the group G_i should depend on its size. As a rule of thumb, we set $Gen_i = \min\{|G_i|+5,500\}$. Whereas the population size NP_i of JADE is set as small as possible during the interaction learning, in the optimization stage, we apply an adaptive strategy. The initial value here is $NP_i = |G_i|+10$, which is sufficient for unimodal problems. After the population loses its diversity and ceases to improve, an independent restart with thrice the population size is performed as suggested in [49]. This is done when the relative fitness improvement of the current cycle compared to the previous on is below 10^{-2} , i.e., $(f_{K-1} - f_K)/f_K < 10^{-2}$.

Furthermore, the difficulties in solving the component functions may vary a lot. Therefore, it is reasonable to stop optimizing converged groups, when no improvements can be achieved for a certain number of cycles (in the context of this work, we set this threshold to five).

5.3 Experimental Studies

5.3.1 Experimental Setup

For benchmarking CCVIL, we choose the set of twenty 1000-dimensional functions provided by in [8]. These functions represent high-dimensional problems with different degrees of variable interactions. This makes them especially suitable to test the ability of our algorithm which was designed for tackling this kind of tasks. We compared CCVIL to DECC-G [17], MLCC [40], and JADE [42]. In the experiments, we grant three million fitness function evaluations to each algorithm run. We fixed the population size in JADE to 1000 and used default parameter settings for DECC-G and MLCC are obtained from the related publications [17, 40]. For each algorithm and benchmark function, 25 independent runs were performed.

5.3.2 Benchmark Functions and Learned Groups

In Table 5.1, we list the characteristics of the benchmark functions applied in our experiments. The column Sep denotes functions which are separable according to Equation 5-1. Most of the non-separable functions consist of mutually separable groups of interacting variables. f_{10} is the sum of ten rotated instances of Rastrigin's function applied to groups of 50 decision variables each and one non-rotated instance of the same function applied to the remaining 500 decision variables. Since the plain Rastrigin's function is separable (and the rotated version is not), an ideal interaction learning stage would thus discover 500 separable groups of size 1 and plus 10 groups of size 50, as listed in column Groups (real). f_{18} is composed of 20 Rosenbrock's functions, each applied to 50 decision variables, leading to 20 "real" groups.

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

Table 5.1: The main characteristics of the 20 benchmark functions [8]

In the last column of Table 5.1, we noted the median of the number of groups discovered by CCVIL during the 25 runs. From these results, we can clearly see that CCVIL most often is able to represent the interactions between the variables correctly. Due to the limitation \hat{K} imposed on the runtime of the learning stage, it may not discover all interactions, i.e., may not merge all interacting groups, and thus, yield a slightly higher number of groups.

Furthermore, we notice that, for most of the benchmark functions related to Ackley's function (f_3, f_6, f_{11}) , CCVIL combines more variables than expected. The reason for this is that Ackley's function (See Figure 5.2) is separable according to Equation 5-1, but not additively separable [50], i.e., it cannot be divided into an exact arithmetic sum of component functions, as pointed out in [51]. Therefore, our algorithm correctly picks up interactions since it aims at representing the fitness function as linear combination of component functions.

The decision variables of Rosenbrock's function, although entirely nonseparable, exhibit only a very weak interaction. Our algorithm discovers that functions f_8 , f_{13} , f_{18} and f_{20} can best be treated as separable problems, i.e., problems with 1000 independent decision variables.

5.3.3 Comparison with other CC-based algorithms and JADE

In Table 5.2, we provide the results of the comparison of our algorithm with DECC-G, MLCC, and plain JADE. We list the mean results of the 25 runs for each benchmark as well as the standard deviations. The columns named R denote the outcomes of a two-tailed Mann-Whitney U test [52] with 5% significance level. A W in column R_1 stands for statistically significant win of CCVIL against both, DECC-G and MLCC, a L a loss, and "–" means that no significant difference was found. Column R_2 represents the same comparison between CCVIL and the native JADE (used as optimizer in CCVIL).

Table 5.2 shows that CCVIL is clearly superior to DECC-G and MLCC. It outperforms them in 15 benchmarks and only loses on f_3 , the separable but not *additively* separable Ackley function. CCVIL also wins against its internal

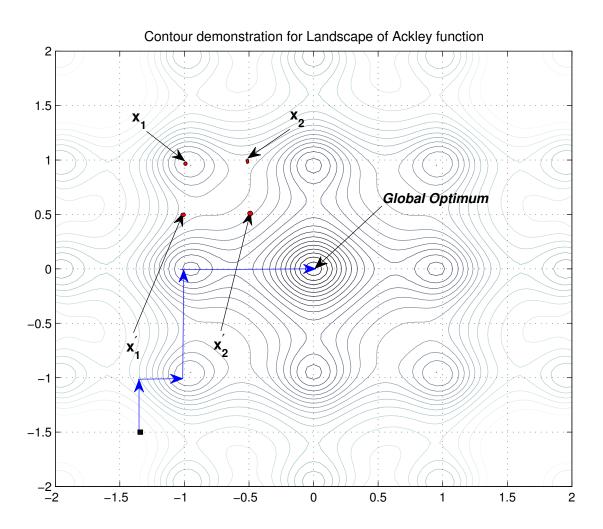


Figure 5.2: Ackley's function is separable yet non additively-separable

	CC	CVIL	DE	CC-G	MI	LCC	R_1	Naive JADE R_2
	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev		Mean Std Dev
f_1	1.55e-17	7.75e-17	2.93e-07	8.62e-08	1.53e-27	7.66e-27	_	1.57e+04 1.38e+04 W
f_2	6.71e-09	2.31e-08	1.31e + 03	3.24e+01	5.55e-01	2.20e+00	W	7.66e+03 9.67e+01 W
f_3	7.52e-11	6.58e-11	1.39e+00	9.59e-02	9.86e-13	3.69e-12	L	4.52e+00 2.41e-01 W
f_4	$5.00e{+12}$	3.38e + 12	9.62e + 12	3.43e + 12	1.70e + 13	5.38e + 12	W	6.14e+09 3.81e+09 L
f_5	1.76e + 08	6.47e + 07	2.63e + 08	8.44e + 07	3.84e + 08	6.93e + 07	W	1.35e+08 1.21e+07 L
f_6	2.94e + 05	6.09e+05	4.96e + 06	8.02e+05	1.62e + 07	4.97e + 06	W	1.94e+01 $1.79e-02$ -
f_7	8.00e + 08	2.48e + 09	1.63e + 08	1.38e + 08	6.89e + 05	7.36e + 05	_	2.99e+01 3.30e+01 -
f_8	6.50e + 07	3.07e+07	6.44e + 07	2.89e + 07	4.38e + 07	3.45e + 07	_	1.19e+04 4.92e+03 L
f_9	6.66e + 07	1.60e + 07	3.21e + 08	3.39e+07	1.23e + 08	1.33e+07	W	2.70e+07 2.08e+06 L
f_{10}	1.28e + 03	7.95e + 01	1.06e + 04	2.93e+02	3.43e + 03	8.72e + 02	W	8.50e+03 2.30e+02 W
f_{11}	3.48e + 00	1.91e+00	2.34e + 01	1.79e+00	1.98e + 02	6.45 e-01	W	9.29e+01 9.66e+00 W
f_{12}	8.95e + 03	5.39e+03	8.93e + 04	6.90e + 03	3.48e + 04	4.91e + 03	W	$6.21e+03\ 1.34e+03\ -$
f_{13}	5.72e + 02	2.55e + 02	5.12e + 03	3.95e + 03	2.08e + 03	7.26e + 02	W	1.87e+03 1.11e+03 W
f_{14}	1.74e + 08	2.68e + 07	8.08e + 08	6.06e + 07	3.16e + 08	2.78e + 07	W	1.00e+08 8.84e+06 L
f_{15}	2.65e + 03	9.34e+01	1.22e+04	9.10e+02	7.10e + 03	1.34e + 03	W	3.65e+03 $1.09e+03$ W
f_{16}	7.18e + 00	2.23e+00	7.66e + 01	8.14e+00	3.77e + 02	4.71e + 01	W	2.09e+02 2.01e+01 W
f_{17}	2.13e+04	9.16e + 03	2.87e + 05	1.97e + 04	1.59e + 05	1.43e + 04	W	7.78e+04 5.87e+03 W
f_{18}	1.33e+04	1.00e+04	2.46e + 04	1.05e+04	7.09e + 03	4.77e + 03	_	3.71e+03 9.58e+02 L
f_{19}	$3.\overline{52e+05}$	2.04e+04	1.11e+06	5.00e + 04	$1.\overline{36e+06}$	$7.\overline{31e+04}$	W	3.48e+05 1.67e+04 -
f_{20}	1.11e+03	3.04e+02	4.06e + 03	3.66e + 02	2.05e+03	1.79e + 02	W	2.06e+03 2.01e+02 W

Table 5.2: Comparison with other CC-based algorithms and plain JADE.

optimizer JADE alone in ten out of 20 benchmark functions and loses in six. Especially for separable and highly non-separable functions, CCVIL proofs to be advantageous. The performance of CCVIL is worse than its JADE in most single-group non-separable functions. The reason is the structure of the benchmark set [8] where a large factor (1 million) is put in front of the nonseparable component function. If even a single interaction is not discovered by CCVIL, it will perform worse than JADE which treats the fitness function as completely nonseparable. This fact leads us to the conclusion that CCVIL can achieve much better results if the learning phase can proceed sufficiently long to discover all interactions. To find a good strategy to distribute runtime between the learning and the optimization stage of CCVIL is thus an interesting point for future research.

5.4 Summary

In this chapter, we introduced a novel CC framework called Cooperative Coevolution with Variable Interaction Learning, or CCVIL for short. In the related work study, we showed that currently no efficient method for finding groups of interacting variables in CC exists. CCVIL fills this gap with a two-stage approach: In a learning step, the interactions between the decision variables of a (potentially very high-dimensional) search space are detected. In the second stage, these groups are optimized according to the traditional CC model. We showed in an experimental study based on twenty 1000-dimensional benchmark functions with different degrees of variable interaction and group sizes that CCVIL can outperform two very efficient, state-of-the-art CC approaches as well as its internal optimizer JADE.

6 Enhance the Effectiveness of Variable Interaction Learning Mechanism

As indicated in Chapter 5, the learning stage in CCVIL [35] consumes as much as 60% of the maximal fitness evaluation, yet it is still incapable to detect all existing interactions. This major shortcoming renders CCVIL impractical in real-world applications. Thereby in this chapter, three techniques are proposed to enhance the effectiveness of variable interaction learning process. According to the experimental studies, these three strategies play a highly positive role, improving the effectiveness of variable interaction learning mechanism statistical significantly.

6.1 Simplify Learning Process by Employing Occam's Razor

The original learning mechanism is basically adopting JADE [42] as the dimension-by-dimension search algorithm, for discovering specific points in the search space. As reported by paper [42], the highlights of JADE is its reliability and potential of handling high-dimensional problems. Yet in the learning process, the target is to optimize in the fastest fashion. What learning stage do is just to identify the variable interactions, instead of obtaining a reliable solution. Clearly, performing optimization in a rapid pace is not JADE's strong suit. Therefore, other greedy yet less reliable search algorithms are likely to speed up the learning process.

Next, Occam's razor [53] is employed in selecting the search algorithm for learning process. The principle of Occam's razor basically recommends that, if goal can be achieved in a simple fashion, do not make it complex. Hence, as the simplest algorithm one can ever think of, random sampling is employed as in the learning process. On the other hand, the random permutation, which is for picking up variables for considering interactions, can be replaced with randomly selection. The change makes the learning process easier to implement with complexity of O(N), while random permutation requires sophisticated techniques like

Fisher-Yates shuffle to achieve the same order of complexity.

6.1.1 Random Sampling Learning Process (RS-LP)

The two techniques introduced above result in a new learning process, which is called Random Sampling Learning Process (RS-LP). The general paradigm of RS-LP is sketched out in Algorithm 4.

```
Algorithm 4: groupInfo \leftarrow Random Sampling Learning Process (RS-LP)
 1 groupInfo \leftarrow \{\{1\}, \{2\}, \dots, \{N\}\}\}; // initially assume full separability
 2 repeat // start a new learning cycle
      repeat // randomly select two variables from distinct groups
          \{indexI, indexJ\} \leftarrow \mathbf{randIndex}(); // \mathbf{randomly} \mathbf{select} \mathbf{two}
              distinct indexes from [1, N]
          G_I \leftarrow \mathbf{find}(groupInfo, indexI); // \mathbf{find} \text{ returns the index of}
 5
          group that the variable belongs to with respect to groupInfo
          G_J \longleftarrow \mathbf{find}(groupInfo, indexJ);
      until G_I \neq G_J;
 6
      \{in\vec{d}iv_1, in\vec{d}iv_2, in\vec{d}iv_1', in\vec{d}iv_2'\} \leftarrow initialIndiv(); // initialize four
 7
           individuals to the same genotype with given bound
      randGene_I \leftarrow initialGene(); // initialize gene with given bound
 8
      randGene_J \leftarrow initialGene();
 9
      // generate four distinct individuals that follow Equation 3-12
      indiv_2[indexJ] \leftarrow randGene_J;
      indiv'_1[indexI] \leftarrow randGene_I;
11
      indiv'_{2}[indexI] \leftarrow randGene_{I}; indiv'_{2}[indexJ] \leftarrow randGene_{J};
12
      if (f(in\vec{d}iv_1) - f(in\vec{d}iv_2)) \times (f(in\vec{d}iv_1') - f(in\vec{d}iv_2')) < 0 then
          groupInfo \leftarrow ((groupInfo \setminus \{G_1\}) \setminus \{G_2\}) \cup (\{G_1 \cup G_2\});
14
          // the indexI^{th} and indexJ^{th} variables interact, merge the
          corresponding groups
15 until termination condition is met (See 6.1.1);
```

6.1.2 Discussion on Parameter Termination Condition

Similar to the learning stage of CCVIL (See Subsection 5.2.1), we need to set up a lower threshold to terminate the learning process if the problem is considered as separable or only very weakly non-separable, and a upper threshold to stop learning process for nonseparable problems.

Let us firstly consider the setting for the *upper threshold*. As we have $\binom{N}{2} = \frac{N \times (N-1)}{2}$ pairs of variables, it is highly necessary to check variable pairs for interaction at least $\frac{N \times (N-1)}{2}$ times, so that each pair has one chance to be checked in average. Moreover, each checking consumes 3 fitness evaluation. Therefore, in case of N=1000, we need to have 1.5×10^6 fitness for non-separable functions.

Next, the setting for the lower threshold and its rationality will be discussed. We would like to guarantee that for a N-dimensional problem, when there are M pairs of interacting variables, the probability of checking one of the interacting variable pairs for at least one time is greater than 99%. Let's see, how many number of fitness evaluation it will cost to achieve this. Notate P_{check} as the probability that we just mentioned and K is the number of times of checking variable pairs, we have $P_{check} = 1 - \left(1 - \frac{M}{\binom{N}{2}}\right)^K = 1 - \left(1 - \frac{M}{\frac{N \times (N-1)}{2}}\right)^K > 0.99$.

As for those problems where only extremely tiny portion of variable pairs are interacting, it is actually unworthy spending enormous amount of computational resources to capture the tiny interactions. What amount of portion should be considered as "tiny" is of course in the context of the scale of problem itself. As a rule of thumb, we treat $M < \frac{1}{N} \times \binom{N}{2}$ as only "tiny" portion of variables are interacting, which we won't try to detect. In other words, we simply take the interaction in problems where $M \geq \frac{1}{N} \times \binom{N}{2}$ into consideration.

Therefore, once $P_{check}(M = \frac{1}{N} \times {N \choose 2}) = 1 - (1 - \frac{1}{N})^K > 0.99$, we will be able to ensure $P_{check} \geq P_{check}(M = \frac{1}{N} \times {N \choose 2}) > 0.99$. For 1000-dimensional problem (i.e., N = 1000), Equation 6-1 holds:

$$K(N = 1000) > \frac{\log(1 - 0.99)}{\log(1 - \frac{1}{N})} = \frac{\log(1 - 0.99)}{\log(1 - \frac{1}{1000})} = \frac{\log(0.01)}{\log(0.999)} = 4602.867216939$$
(6-1)

since each checking consumes 3 fitness evaluation, we need to have 1.4×10^4 fitness

evaluation to examine whether it is a separable function.

6.1.3 Experimental Study

In this section, experimental results are presented to verify the efficacy of the newly proposed learning process RS-LP and the enhancement of CCVIL by replacing the original learning stage. The empirical studies is based on the benchmark set proposed by CEC'2010 Special Session on Large-Scale Global Optimization (LSGO). Experimental setup is consistent with the studies in Subsection 5.3.1.

The improvement on the efficacy of learning process is demonstrated in the first place. Table 6.1 show the considerable advancement in the learning variable interactions:

- In many functions, RS-LP is capable to discover all existing interactions, while Qri-LS rarely achieve this.
- The interactions in Rosenbrock's function is difficult for both Qri-LS and RS-LP to discover. Taking a closer look at the Rosenbrock's function, we observe that there are only N-1 pairs of interacting variables. While for other strongly non-separable function like Schwefel's Problem 1.2, the amount of interacting variable pairs jump up to $\frac{N\times(N-1)}{2}$. This could provide some clues for why interactions in Rosenbrock's problem are so hard to be discovered.

In Table 6.2, we observe that with the new Ori-LS, CCVIL finds significantly better solution on many problems than the Ori-LS. RS-LP even achieves better performance than MA-SW-Chains [54], which is the highly-recognized champion of the competition on large-scale global optimization at CEC'2010. This demonstrates the competence of our new approach goes beyond just CCEAs, its global search ability is leading among a variety of bio-inspired methods.

Benchmark	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}
Ori-LS found	1000	1000	969	963	952	921	952	1000	627	516
RS-LP found	1000	1000	397	951	951	394	951	1000	510	510
Real group num.	1000	1000	951	951	951	951	951	951	510	510
Benchmark	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}	f_{16}	f_{17}	f_{18}	f_{19}	f_{20}
Ori-LS found	501	522	1000	232	37	39	42	1000	1	1000
0	001	022	1000	232	31	39	42	1000	1	1000
RS-LP found	286	510	1000	$\frac{232}{21}$	20	$\frac{39}{20}$	20	1000	1	1000

Table 6.1: Comparison of the learning efficacy between the Original Learning Stage (Ori-LS) in CCVIL and the newly proposed Random Sampling Learning Process (RS-LP). The median values are listed.

6.2 Inspirations From Linkage Learning Techniques

6.2.1 Background

Linkage is referred to the interdependencies among decision variables. In the past few decades, there are increasingly studies focus on linkage learning, for the purpose of handling complicated, large-scale problems [9, 55]. In other words, linkage learning and variables interactions learning has the shared destination. Even though their encoding scheme differ, linkage learning techniques deal with bit strings while variable interactions learning approaches work on real values, the ideas from both techniques are supposed to share and researchers on the two topics can learn from each other therefore. Here this chapter, we would like to initiate the tunnel from linkage learning to variable interaction learning, since linkage learning has long being studies [27, 28, 29, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66] and we, as the vanguard of exploring variable interaction learning, started our work in 2010 [35].

	Ori-LS-	·CCVIL	R_1	RS-LP-CCVIL		R_2	MA-SW	-Chains
f_1	1.55e-17	7.75e-17	ı	0.00e+00	0.00e+00	W	2.10e-14	1.99e-14
f_2	6.71e-09	2.31e-08	L	2.97e-08	7.29e-08	W	8.10e+02	5.88e + 01
f_3	7.52e-11	6.58e-11	L	7.07e-01	2.69e-01	L	7.28e-13	3.40e-13
f_4	5.00e+12	3.38e + 12	W	3.66e+12	7.55e + 12	-	3.53e + 11	3.12e+10
f_5	1.76e + 08	6.47e + 07	-	2.10e+08	1.39e + 08	-	1.68e + 08	1.04e + 08
f_6	2.94e+05	6.09e + 05	L	6.81e+05	5.87e + 05	L	8.14e+04	2.84e + 05
f_7	8.00e + 08	2.48e + 09	L	9.22e+08	4.61e + 09	L	1.03e+02	8.70e+01
f_8	6.50e + 07	3.07e + 07	W	3.99e+07	3.48e + 07	L	1.41e+07	3.68e + 07
f_9	6.66e + 07	1.60e + 07	W	5.88e + 06	9.87e + 05	W	1.41e + 07	1.15e + 06
f_{10}	1.28e+03	7.95e + 01	W	9.67e + 02	4.15e + 01	W	2.06e + 03	1.40e + 02
f_{11}	3.48e+00	1.91e + 00	-	2.44e+00	1.66e + 00	W	3.77e + 01	6.85e + 00
f_{12}	8.95e+03	5.39e + 03	W	6.62e+01	1.73e + 02	L	3.62e-06	5.92e-07
f_{13}	5.72e + 02	2.55e + 02	-	5.57e + 02	8.05e+01	W	1.25e + 03	5.72e+02
f_{14}	1.74e + 08	2.68e + 07	W	1.80e+07	2.21e + 06	W	3.10e+07	2.19e+06
f_{15}	2.65e+03	9.34e+01	W	2.20e+03	7.22e+01	W	2.72e + 03	1.22e+02
f_{16}	7.18e+00	2.23e+00	W	3.28e+00	1.89e + 00	W	1.01e+02	1.45e + 01
f_{17}	2.13e+04	9.16e + 03	W	1.49e+02	1.24e + 02	L	1.24e+00	1.25e-01
f_{18}	1.33e+04	1.00e + 04	W	2.43e+03	3.54e + 03	L	1.30e + 03	4.36e+02
f_{19}	3.52e+05	2.04e+04	-	3.47e + 05	1.77e + 04	L	2.85e + 05	1.78e + 04
f_{20}	1.11e+03	3.04e+02	-	1.18e+03	2.59e+02	-	1.07e + 03	7.29e+01

Table 6.2: Comparison with the original CCVIL, as well as MA-SW-Chains, which is the champion of the CEC'2010 Competition on Large-Scale Global Optimization [8]. The columns named R_1 and R_2 are the results of Wilcox Rank Sum Test TODO: Ref at 5% significance level. 'W' indicates RS-LP-CCVIL is statistically significantly better than the algorithm in comparison, which is Ori-LS-CCVIL with column R_1 and MA-SW-Chains with columns R_2 . 'L' in columns R_1 and R_2 stands for a statistical inferiority of RS-LP-CCVIL against the compared algorithm. '-' means the difference is insignificant.

6.2.2 Generalizing the Definition of Variable Interaction

Recall that *interdependences* among decision variable is about "two variables in a problem are interdependent if the fitness contribution or optimal setting for one variable depends on the setting of the other variable". In the definition of variable interaction previously introduced by us (See Equation 3-12), "the order relation of two individuals, which differ in only one allele, is inverted by the setting of another allele" is described as variable interaction. More precisely, we consider the inversion of order relation equivalent to the interdependency. It is incomplete yet. The interdependencies among variables are supposed to include more scenarios that are not part of our previous definition.

Munetomo shows a more general definition of linkage, which of course deals with bit strings [67] (See Definition 6.2.1).

Definition 6.2.1. Two position i and j are linked w.r.t. a function f if there is some string s such that $\Delta f_i(s[j \to 0]) \neq \Delta f_i(s[j \to 1])$. $\Delta f_i(s[j \to 0])$ stands for the change in fitness when i^{th} allele flip over if the j^{th} allele assigned to 0.

Motivated by Streeter's definition of linkage, we propose a more general definition of variable interaction, which is a extension of our previous one (See Equation 3-12). Two key changes are made to accommodate Definition 6.2.1 from the field of bit-string to the domain of real value: 1) The flipping operation on the i^{th} bit is changed to the assignment of a randomly selected value over search space, to the i^{th} variable. 2) Two randomly selected values are set to j^{th} variable respectively for considering the impact of the setting of the j^{th} variable. We abstract the generalized definition of variable interaction in Definition 6.2.2.

Definition 6.2.2. Two variable are interacting with regard to a function f, if there exist four points in the search space, such that

$$f(x_1, \dots, x_i, \dots, x_j, \dots, x_N) - f(x_1, \dots, x_i', \dots, x_j, \dots, x_N) \neq$$

$$f(x_1, \dots, x_i, \dots, x_i', \dots, x_N) - f(x_1, \dots, x_i', \dots, x_i', \dots, x_N)$$

$$(6-2)$$

6.2.3 Will Local Search be Helpful for Learning Variable Interaction?

In [50], an algorithm using efficient linkage discovery in conjunction with local search is presented. The algorithm basically employs a local search to make the string under consideration for linkage optimal with respect to one-bit perturbations, before perform linkage discovery operations. Yet the underlying reason for employing local search is still unknown. Is it beneficial for finding the globally optimal solution or good for discovering linkage? Streeter does not uncover it in [50].

As our target in this chapter is to develop an efficient variable interaction learning process, it would be intriguing to investigate whether performing local search with respect to the fitness will enhance the learning process? Even though the learning process has a different objective, i.e., the features of interacting variables, objective as such to search for global optimum with respect to fitness and to discover variable interactions might be aligned to some degree.

Again based on the principles of Occam's Razor, the simplest local search methods "Random Walk" will be employed in learning variable interactions. Empirical studies will later be presented in Subsection 6.2.5.

6.2.4 Reducing the Runtime Complexity of Learning Process By Binary Search

The most efficient linkage learning techniques in terms of runtime complexity so far was proposed by Streeter [50]. He formally proofs that, his approach is capable to discover linkage of an additively separable functions with runtime of $O(N \times log(N))$, which approaches before [50] consumes a runtime of order $O(N^2)$. The substantial advancement is basically achieved by incorporating binary search.

For a specific linkage between i^{th} and j^{th} positions, the discover process can be divided into two stages: 1) to locate one of the position (either i^{th} or j^{th} in this case) involved in the specific linkage, which requires at most O(N) trials. 2)

to locate the other one position with binary search, and time with an order of O(log(N)) must be taken.

Streeter's method is shown to discover linkage on the basis of bit-strings. Whether the method will work with real-value representation, which is actually a more natural and popular way of presenting variables, is still left to disclose. Inspired by [50], we attempt to accelerate the learning process (RS-LP) in Subsection 6.2.5 one step further by introducing binary search. We list the general structure of the Learning Process with Binary Search (LP-BS) in Algorithm 5, Algorithm 6 and Algorithm 7.

6.2.5 Experimental Studies

The flow of our comprehensive empirical studies is listed in Figure 6.1, the abbreviations listed in the brackets will be used for reference from now on. All six learning strategy with different combination of the three techniques are benchmarked with the same experimental setting in 5.3.1, except for the optimization stage is removed, for we are currently interested in the efficiency of various learning mechanism. Non-separable functions from f_{14} to f_{20} are involved in the empirical study.

The results are presented in Table 6.3. A number of interesting points can be inferred from the results:

- 1. Nearly all modification indicated in the arc of Figure 6.1 play remarkably positive role in improving the efficiency of learning process. For instance, see the first column, i.e., the one on f_{14} with 'RS-LP', 'RS-GenDef-LP' and 'RS-GenDef-BinSearch', changes made to the learning strategy both shorten the convergence time to nearly 10% of the one without changes. RS-LP can not even discover all interactions, while RS-GenDef-BinSearch detects all interaction with only 1% of the fitness evaluation.
- 2. In the case of f_{19} , introducing binary search for interacting position actually degrade the efficiency of learning process. This is because the A_{intr} (see Section for reference) of f_{19} is all filled with 1. Basically, every check with

Algorithm 5: $groupInfo \leftarrow$ Learning Process with Binary Search (LP-BS)

```
1 groupInfo \leftarrow \{\{1\}, \{2\}, \dots, \{N\}\}\}; // initially assume full separability
 2 repeat // start a new learning cycle
      indexI \leftarrow \mathbf{randIndex}(); // \mathbf{randomly}  select one distinct indexes
          from [1,N]
      \{in\vec{d}iv_1, in\vec{d}iv_2\} \leftarrow initialIndiv(); // initialize two individuals to
 4
           the same genotype with given bound
      Let \{j_1, \dots, j_m\} be the indexes of variables belonging to groupInfo[indexI];
 \mathbf{5}
      indiv_2[j_1,\cdots,j_m] \longleftarrow indiv_1[j_1,\cdots,j_m]; // in\vec{d}iv_1 \text{ and } in\vec{d}iv_2 \text{ differ}
 6
           only in alleles not belonging to qroupInfo[indexI]
      if testInteraction(indiv_1,indiv_2,indexI) = true then
 7
          // using binary search
          indexJ \leftarrow \mathbf{findInteractIndex}(in\vec{d}iv_1, in\vec{d}iv_2, indexI);
 8
      else /* testInteraction returns false, no interaction is found */
 9
          indexJ \leftarrow -1;
10
      if indexJ \neq -1 then
11
          G_I \leftarrow \mathbf{find}(groupInfo, indexI); // \mathbf{find} \text{ returns the index of}
12
          group that the variable belongs to with respect to groupInfo
          G_J \leftarrow \text{find}(qroupInfo, indexJ);
          groupInfo \leftarrow ((groupInfo \setminus \{G_1\}) \setminus \{G_2\}) \cup (\{G_1 \cup G_2\});
13
          // the indexI^{th} and indexJ^{th} variables interact, merge the
          corresponding groups
```

14 until termination condition is met;

Algorithm 6: $testResult \leftarrow \mathbf{testInteraction}(in\vec{d}iv_1, in\vec{d}iv_2, indexI)$

```
// Generating four individuals in Definition 6.2.2 by perturbation randGene_I \leftarrow \textbf{initialGene}(indiv_1, indiv_2, indexI); // initialize gene with given bound randiv_1 \leftarrow indiv_1; indiv_1'[indexI] \leftarrow randGene_I; randGene_I; randGene_I; randGene_I; if (f(indiv_1) - f(indiv_2)) \neq (f(indiv_1') - f(indiv_2')) then randGene_I \leftarrow true; celse /* no interaction is detected */ randGene_I \leftarrow true; false;
```

Algorithm 7: $indexJ \leftarrow findInteractionIndex(in\vec{d}iv_1,in\vec{d}iv_2,indexI)$

```
1 Let \{j_1, j_2, \dots, j_{\sigma}\}, be the \sigma distinct positions for which the values of in\vec{d}iv_1 and in\vec{d}iv_2 differ;

2 if \sigma = 1 then

3 | indexJ \leftarrow j_1;

4 else

5 | indiv_3 \leftarrow indiv_2; indiv_3[j_1, \dots, j_{\lfloor \sigma/2 \rfloor}] \leftarrow indiv_2[j_1, \dots, j_{\lfloor \sigma/2 \rfloor}];

6 | if testInteraction(in\vec{d}iv_1, in\vec{d}iv_3, indexI) = true then

7 | indexJ \leftarrow findInteractIndex(in\vec{d}iv_1, in\vec{d}iv_3, indexI);

8 | else

9 | indexJ \leftarrow findInteractIndex(in\vec{d}iv_2, in\vec{d}iv_3, indexI);
```

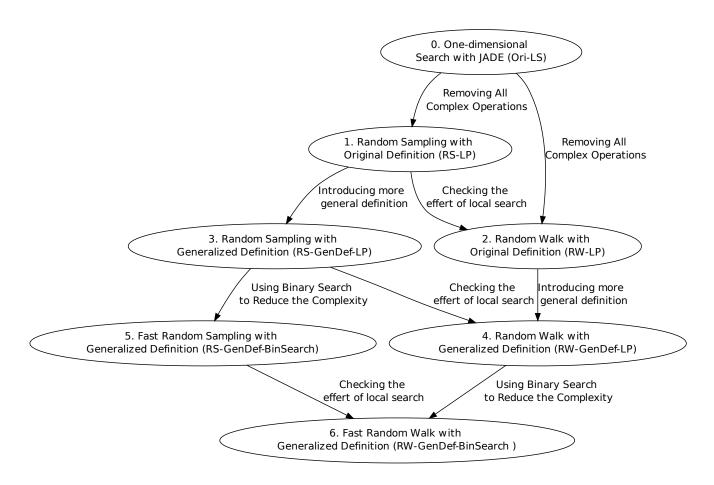


Figure 6.1: The mind map of designing experimental studies in Subsection 6.2.5

	RS-L	P	RS-GenD	ef-LP	RS-GenDef-BinSearch		
f_{14}	2.98e + 06	(60)	2.99e+05	(20)	4.41e+04	(20)	
f_{15}	5.16e+05	(20)	2.99e+05	(20)	4.41e+04	(20)	
f_{16}	4.50e+05	(20)	2.99e+05	(20)	4.41e+04	(20)	
f_{17}	2.98e+06	(73)	2.99e+05	(20)	4.41e+04	(20)	
f_{18}	2.94e+06	(976)	2.99e+06	(241)	5.55e + 04	(20)	
f_{19}	2.84e + 05	(1)	4.00e+03	(1)	4.06e+04	(1)	
f_{20}	2.94e+06	(974)	2.99e+06	(217)	5.67e + 04	(1)	
	RW-L	ıP	RW-GenI	ef-LP	RW-GenDet	-BinSearch	
f_{14}	RW-L 1.55e+06	(20)	RW-GenE 2.67e+05	Def-LP (20)	RW-GenDet 4.41e+04	F-BinSearch (20)	
f_{14} f_{15}							
	1.55e+06	(20)	2.67e + 05	(20)	4.41e+04	(20)	
f_{15}	1.55e+06 7.94e+05	(20) (20)	2.67e+05 2.67e+05	(20) (20)	4.41e+04 4.41e+04	(20) (20)	
f_{15} f_{16}	1.55e+06 7.94e+05 4.69e+05	(20) (20) (20)	2.67e+05 2.67e+05 2.67e+05	(20) (20) (20)	4.41e+04 4.41e+04 4.41e+04	(20) (20) (20)	
f_{15} f_{16} f_{17}	1.55e+06 7.94e+05 4.69e+05 7.75e+05	(20) (20) (20) (20)	2.67e+05 2.67e+05 2.67e+05 2.67e+05	(20) (20) (20) (20)	4.41e+04 4.41e+04 4.41e+04 4.41e+04	(20) (20) (20) (20)	

Table 6.3: Six strategies (the interpretation for the exotic abbreviations can be found in Figure 6.1) for variable interaction learning are categorized into two classes: 1) using Random Sampling as the search algorithm, the abbreviation starts with 'RS'; 2) using Random Walk as the search algorithm, the abbreviation starts with 'RW'. The consumption of fitness evaluation when the learning process converges, i.e., no more groups is merged after that point, are listed left side of column. And the number of groups left when learning process converges or reach the maximal fitness evaluation is presented inside the right brackets. All data listed are the medians over 25 independent runs.

generalized definition is capable to detect the interaction (We can deduce this from the fact that, 4.00e+03 fitness evaluation are spend in merging 999 interacting variables, and we know beforehand that each check requires 4 fitness evaluations). While using binary search, which requires at least log_2N (= $log_21000 = 10$) to locate the interacting position, it thus consumes 10 times more fitness evaluation than the one without binary search.

- 3. The random walk appears to be always superior than random sampling, which inspires us to try more greedy search algorithm for further accelerating the learning process.
- 4. A weird result is found when RW-GenDef-LP is employed on f_{18} . It seems that RW-GenDef-LP merges more groups than expected, the underlying reasons will be uncovered in our future works.
- 5. We are pretty confident that, by applying the three techniques in this subsection will definitely enhance the performance of CCVIL further, and resulting CCVIL will outperform the MA-SW-Chains in a more remarkable way. We will verify this very soon in the future by empirical studies.

7 Conclusions

7.1 Summary

In this thesis, we intended to solve large-scale numerical optimization effectively, since it is a hot topic and has a variety of applications over many fields. Current approaches often fail to find satisfactory solutions to large-scale problems, since they suffer from the "Curse of Dimensionality". We handled large-scale problems in a divide and conquer manner: firstly decompose problem into several non-overlapping subcomponents and solve them in a round-robin fashion. Many paper reports that inappropriate decomposition strategy can lead to the existence of interdependencies among subcomponents, which in return, significantly degrade the global search ability of Cooperative Coevolutionary Algorithms (CCEAs). Consequently, recent works are devoted to increase the chance of grouping interacting variables into same subcomponents. Yet none of them is capable to decompose problems ideally, i.e., to group strongly interacting variables into the same subcomponents, leaving no/weak interdependencies among distinct subcomponents.

On the other hands, linkage learning, which has been extensively studies for many years, is the similar topic in the field of genetic algorithms. A couple of competent linkage learning techniques have been developed in the past few decades. Nevertheless, previous studies on linkage are only limited with the representation of bit strings, there are few studies addressing the possibility as well as the importance of interdependency identification with the presentation of real values.

Motivated by the discussion above, we initiates the research on incorporating interdependency identification and developed several highly competent CCEAs. We firstly addressed serious weaknesses of current formulations for interdependencies among decision variables, and proposes a new definition for independences, which refer as variable interaction from then on. By conducting comprehensive experimental studies, we firmly demonstrate the importance of problem decomposition strategy, making use of variable interaction knowledge dramatically im-

prove the performance of CCEAs. As a result, we proposed a novel two-stage CCEA, Cooperative Coevolution with Variable Interaction Learning (CCVIL). The first stage of CCVIL is learning stage, which is a systematic learning process designed by us. Experimental studies showed that CCVIL dominates state-of-the-art CCEAs in terms of the quality of found solutions, it also enhance the large-scale search ability of its internal optimizer JADE.

Even though CCVIL displays great performance, there are still a lot of space where CCVIL can be improved. Probably the most outstanding drawback of CCVIL is its high consumption of computational resources in learning stage. Thereby, we targeted on accelerating variable learning process. Firstly, we removes all complex operations in learning stage according to the principle of Occam's Razor. We replaced JADE with random sampling as dimension-by-dimension search algorithm, the new learning process is notated as RS-LP. Simulation results show that RS-LP discovers more variable interactions with less fitness evaluation, and of course finds much better solution the original CCVIL (Ori-LS-CCVIL). In addition, RS-LP-CCVIL take the superior position in comparison with MA-SW-Chains, which is the champion of CEC'2010 competition on large-scale global optimization.

Finally, we introduce three techniques inspired by linkage learning approaches. They are:

- 1. generalization of the definition of variable interaction,
- 2. incorporating local search into the dimension-by-dimension search algorithm,
- 3. and discovering the position of interaction using binary search.

7.2 Future Works

As stated before, the purpose of our thesis is to initiate the studies on incorporating variable interaction techniques in cooperative coevolutionay algorithms and solve real-world large-scale problems effectively. Due to the originality of our work, there are a number of aspects where further attention could be paid on. We list some of them in the following subsections.

7.2.1 Verification of the Efficacy of CCVIL in various Real-World Applications

All previous work of ours' are based on benchmark function set. Even though the task force who introduces the benchmark are sort of authoritative, nobody can tell how well the benchmark functions can estimate the complex scenarios emerging in real-world. Consequently, the great performance of CCVIL on benchmark set does not necessarily mean our approaches work well in practice, too. Actually, some researchers has employed CCEAs in several practical domains, such as Multi-Agent Systems [25] and Wireless Sensor Deployment [68]. It would be interesting to figure out whether our variable interaction learning mechanism will work or not in such applications.

7.2.2 The Scalability of CCVIL

Obviously in industrial and engineering problems, the number of variables is not necessarily limited to 1000 or some other constants. We of course noticed this, thus set the running parameters of CCVIL with respect to dimensionality and discuss the underlying rationality as well. Yet how these setting would actually work out are left to uncover.

7.2.3 Multiple Optimization Strategies in the Optimization Stage

As the size of subpopulations may vary from one to thousands, it is intuitive to employ different search algorithms with regard to the number of variables in the corresponding subpopulations. For small-scale subproblems, some greedy local search techniques should find sufficiently good solution. Yet for large-scale subproblems, less greedy and more reliable algorithms are mostly preferred over others.

7.2.4 More Sophisticated Approaches for Separable yet Not Additively Separable Functions

As shown before in the experimental studies, CCVIL always perform poor on all functions related with Ackley's function, which is a typical separable yet not additively separable function. CCVIL is incapable to distinguish such kind of problem from the non-separable ones, it tends to consider functions like Ackley's as non-separable ones and solve it in an inefficient way.

7.2.5 Evolving Variable Partitioning

All variable interactions techniques introduced in this thesis are based on the analysis of fitness landscape. We expect such analysis is worthy and should truly reflect the characteristics of the problems, and understanding the problems' features will be beneficial for optimization. However, such analysis sometime could be misleading and harmful for search, as in the case of Ackley's function. Therefore, it would be intriguing to self-adapt the variable partitioning with respect to fitness, since this feature is supposedly a better guide for optimization.

Bibliography

- [1] Nicholas I. M. Gould, Dominique Orban, and Philippe L. Toint. Numerical methods for large-scale nonlinear optimization. *Acta Numerica*, 14:299–361, 2005.
- [2] Brett M. Averick and Jorge J. Moré. Model problems for large-scale optimization. via website: http://www.mcs.anl.gov/more/tprobs/. URL http://www.mcs.anl.gov/~more/tprobs/.
- [3] Mitchell A. Potter. The Design and Analysis of a Computational Model of Cooperative Coevolution. PhD thesis, George Mason University, 1997.
- [4] Weicai Zhong, Jing Liu, Mingzhi Xue, and Licheng Jiao. A multiagent genetic algorithm for global numerical optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 34(2):1128 1141, april 2004. ISSN 1083-4419. doi: 10.1109/TSMCB.2003.821456.
- [5] Kenneth Alan De Jong. Evolutionary Computation: A Unified Approach. MIT Press, 2006.
- [6] Stephanie Forrest. Genetic Algorithms: Principles of Natural Selection Applied to Computation. Science, 261:872–878, August 1993. doi: 10.1126/science.8346439.
- [7] Zbigniew Michalewicz. Genetic Algorithms + Data Structures = Evolution Programs (3rd). Springer-Verlag New York, Inc., New York, NY, USA, 1996. ISBN 3-540-58090-5.
- [8] Ke Tang, Xiaodong Li, Ponnuthurai Nagaratnam Suganthan, Zhenyu Yang, and Thomas Weise. Benchmark functions for the CEC'2010 special session and competition on large scale global optimization. Technical report, NICAL, USTC, Hefei, Anhui, China, 2009. http://nical.ustc.edu.cn/cec10ss.php.

- [9] David E. Goldberg. Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989. ISBN 0201157675.
- [10] Darrell Whitley. The genitor algorithm and selection pressure: why rank-based allocation of reproductive trials is best. In *Proceedings of the third international conference on Genetic algorithms*, pages 116–121, San Francisco, CA, USA, 1989. Morgan Kaufmann Publishers Inc. ISBN 1-55860-006-3. URL http://portal.acm.org/citation.cfm?id=93126.93169.
- [11] Xin Yao and Yong Liu. Fast evolutionary programming. In *Proceedings of the Fifth Annual Conference on Evolutionary Programming*, pages 451–460, 1996.
- [12] Yong Liu, Xin Yao, Qiangu Zhao, and Tetsuya Higuchi. Scaling Up Fast Evolutionary Programming with Cooperative Coevolution. In *IEEE Congress on Evolutionary Computation*, volume 2, pages 1101–1108. IEEE Press, 2001.
- [13] Hans-Georg Beyer and Hans-Paul Schwefel. Evolution strategies a comprehensive introduction. Natural Computing, 1:3–52, 2002. ISSN 1567-7818. URL http://dx.doi.org/10.1023/A:1015059928466. 10.1023/A:1015059928466.
- [14] 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.
- [15] Scott Kirkpatrick. Optimization by simulated annealing: Quantitative studies. Journal of Statistical Physics, 34:975–986, 1984. ISSN 0022-4715. URL http://dx.doi.org/10.1007/BF01009452. 10.1007/BF01009452.
- [16] Frans van den Bergh and Andries P. Engelbrech. A cooperative approach to particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, 8(3):225–239, 2004.

- [17] Zhenyu Yang, Ke Tang, and Xin Yao. Large Scale Evolutionary Optimization Using Cooperative Coevolution. *Information Sciences*, 178(15):2985–2999, 2008.
- [18] Richard E. Bellman. Dynamic Programming. Princeton University Press, Princeton, NJ, USA, 1957.
- [19] David E. Goldberg and Kalyanmoy Deb. A comparative analysis of selection schemes used in genetic algorithms. In *Foundations of Genetic Algorithms*, pages 69–93. Morgan Kaufmann, 1991.
- [20] A. Brindle. Genetic Algorithms for Function Optimization. Phd thesis, University of Alberta, 1981.
- [21] H.P.P. Schwefel. Evolution and Optimum Seeking: The Sixth Generation. John Wiley & Sons, Inc. New York, NY, USA, 1993.
- [22] Ingo Rechenberg. Evolutionsstrategie 94, volume 1 of Werkstatt Bionik und Evolutionstechnik. Frommann-Holzboog, Stuttgart, 1994.
- [23] Chang-Yong Lee and Xin Yao. Evolutionary programming using mutations based on the lévy probability distribution. *IEEE Transaction on Evolutionary Computation*, 8(1):1–13, 2004.
- [24] Mitchell A. Potter and Kenneth Alan De Jong. Cooperative coevolution: architecture for evolving coadapted subcomponents. *Evolutionary Computation*, 8(1):1–29, 2000.
- [25] Chern Han Yong and Risto Miikkulainen. Cooperative Coevolution of Multi-Agent Systems. *University of Texas at Austin, Austin, TX*, 2001.
- [26] Edward D. Weinberger. NP Completeness of Kauffman's N-k Model, a Tuneably Rugged Fitness Landscape.
- [27] Georges Raif Harik. Learning gene linkage to efficiently solve problems of bounded difficulty using genetic algorithms. PhD thesis, Ann Arbor, MI, USA, 1997.

- [28] Ying-Ping Chen. Extending the Scalability of Linkage Learning Genetic Algorithms: Theory and Practice. PhD thesis, University of Illinois at Urbana-Champaign, 2004.
- [29] Sung-Soon Choi, Kyomin Jung, and Byung Ro Moon. Lower and upper bounds for linkage discovery. *IEEE Trans. Evolutionary Computation*, 13 (2):201-216, 2009. URL http://dblp.uni-trier.de/db/journals/tec/tec13.html#ChoiJMO9.
- [30] Karsten Weicker and Nicole Weicker. On the improvement of coevolutionary optimizers by learning variable interdependencies. In *Proc.* 1999 Congress on Evolutionary Computation (CEC'99). IEEE Press, pages 1627–1632, 1999.
- [31] Myung Kim and Joung Ryu. Species merging and splitting for efficient search in coevolutionary algorithm. In Chengqi Zhang, Hans W. Guesgen, and Wai-Kiang Yeap, editors, *PRICAI 2004: Trends in Artificial Intelligence*, volume 3157 of *Lecture Notes in Computer Science*, pages 332–341. Springer Berlin / Heidelberg, 2004. URL http://dx.doi.org/10.1007/978-3-540-28633-2_36.
- [32] Myung Kim, Joung Ryu, and Eun Kim. A coevolutionary algorithm with spieces as varying contexts. In Rajiv Khosla, Robert Howlett, and Lakhmi Jain, editors, *Knowledge-Based Intelligent Information and Engineering Systems*, volume 3684 of *Lecture Notes in Computer Science*, pages 906–906. Springer Berlin / Heidelberg, 2005. URL http://dx.doi.org/10.1007/11554028_26.
- [33] Myung Won Kim and Joung Woo Ryu. An efficient coevolutionary algorithm using dynamic species control. *International Conference on Natural Computation*, 3:431–435, 2007. doi: http://doi.ieeecomputersociety.org/10.1109/ICNC.2007.191.
- [34] Edwin D. De Jong, Richard A. Watson, and Dirk Thierens. On the complexity of hierarchical problem solving. In Katja Verbeeck, Karl Tuyls, Ann Nowé, Bernard Manderick, and Bart Kuijpers, editors, *BNAIC*, pages

- 339-340. Koninklijke Vlaamse Academie van Belie voor Wetenschappen en Kunsten, 2005. URL http://dblp.uni-trier.de/db/conf/bnaic/bnaic2005.html#JongWT05.
- [35] Wenxiang Chen, Thomas Weise, Zhenyu Yang, and Ke Tang. Large-Scale Global Optimization Using Cooperative Coevolution with Variable Interaction Learning. In Robert Schaefer, Carlos Cotta, Joanna Kolodziej, and Günter Rudolph, editors, Parallel Problem Solving from Nature PPSN XI, volume 6239 of Lecture Notes in Computer Science, pages 300–309. Springer Berlin / Heidelberg, 2010. URL http://dx.doi.org/10.1007/978-3-642-15871-1_31.
- [36] Hemant Singh Kumar and Tapabrata Ray. Divide and conquer in coevolution: A difficult balancing act. In Lim Meng Hiot, Yew Soon Ong, Ruhul Amin Sarker, and Tapabrata Ray, editors, Agent-Based Evolutionary Search, volume 5 of Evolutionary Learning and Optimization, pages 117–138. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-13425-8. URL http://dx.doi.org/10.1007/978-3-642-13425-8_6.
- [37] Mohammad N. Omidvar and Xiaodong Li. Cooperative coevolutionary algorithms for large scale optimisation. Technical report, The Royal Melbourne Institute of Technology (RMIT).
- [38] Mitchell A. Potter and Kenneth Alan De Jong. A cooperative coevolutionary approach to function optimization. In 3rd Conf. on Parallel Problem Solving from Nature, volume 2, pages 249–257, 1994.
- [39] Tapabrata Ray and Xin Yao. A Cooperative Coevolutionary Algorithm with Correlation Based Adaptive Variable Partitioning. pages 983 –989, May. 2009. doi: 10.1109/CEC.2009.4983052.
- [40] Zhenyu Yang, Ke Tang, and Xin Yao. Multilevel cooperative coevolution for large scale optimization. In *IEEE Congress on Evolutionary Computation*, pages 1663–1670. IEEE Press, 2008.

- [41] Chi-Keong Goh and Kay Chen Tan. A competitive-cooperative coevolutionary paradigm for dynamic multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 13(1):103 –127, feb. 2009. ISSN 1089-778X. doi: 10.1109/TEVC.2008.920671.
- [42] Jingqiao Zhang and Arthur C. Sanderson. JADE: adaptive differential evolution with optional external archive. *IEEE Transactions on Evolutionary Computation*, 13(5):945–958, 2009.
- [43] John Forbes Nash. Non-cooperative Games. PhD thesis, Princeton University, Princeton, NJ, May 1950. URL http://www.princeton.edu/mudd/news/faq/topics/nash.shtml.
- [44] R. Sarker, M. Mohammadian, and X. Yao. *Evolutionary Optimization*. Kluwer Academic Publishers Norwell, MA, USA, 2002.
- [45] P. Husbands and F. Mill. Simulated co-evolution as the mechanism for emergent planning and scheduling. In 4th Intl. Conf. on Genetic Algorithms, pages 264–270. Morgan Kaufmann, 1991.
- [46] U. Aickelin. A Pyramidal Evolutionary Algorithm with Different Inter-Agent Partnering Strategies for Scheduling Problems. In *GECCO Late-Breaking* Papers, pages 1–8.
- [47] Kenneth Price, Rainer Storn, and Jouni A. Lampinen. Differential Evolution: A Practical Approach to Global Optimization. Springer-Verlag, 2005. ISBN 3-540-20950-6.
- [48] Uday K. Chakraborty, editor. Advances in Differential Evolution. Springer, Berlin, 2008. URL http://portal.acm.org/citation.cfm?id=SERIES11878.1479611.
- [49] Ann Auger and Nikolaus Hansen. A restart CMA evolution strategy with increasing population size. In *IEEE Congress on Evolutionary Computation*, volume 2. IEEE Press, 2005.

- [50] Matthew J. Streeter. Upper bounds on the time and space complexity of optimizing additively separable functions. In *Genetic and Evolutionary Computation–GECCO 2004*, pages 186–197. Springer, 2004.
- [51] Nikolaus Hansen and Stefan Kern. Evaluating the CMA evolution strategy on multimodal test functions. In Parallel Problem Solving from Nature – PPSN VIII, pages 282–291. Springer, 2004.
- [52] Henry B. Mann and Donald R. Whitney. On a test of whether one of two random variables is stochastically larger than the other. *The Annals of Mathematical Statistics*, 18(1):50–60, 1947. doi: 10.1214/aoms/1177730491.
- [53] C. E. Rasmussen and Z. Ghahramani. Occam's razor. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, Advances in Neural Information Processing Systems 13, pages 294–300. MIT Press, Cambridge, MA, 2001.
- [54] Daniel Molina, Manuel Lozano, and Francisco Herrera. Ma-sw-chains: Memetic algorithm based on local search chains for large scale continuous global optimization. In *IEEE Congress on Evolutionary Computation*, pages 1–8. IEEE, 2010. URL http://dblp.uni-trier.de/db/conf/cec/cec2010.html#MolinaLH10.
- [55] YP Chen, T.L. Yu, K. Sastry, and D.E. Goldberg. A survey of linkage learning techniques in genetic and evolutionary algorithms. Technical report, 2007.
- [56] T.S.P.C. Duque and D.E. Goldberg. A new method for linkage learning in the ECGA. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 1819–1820. ACM, 2009.
- [57] Georges Harik. Linkage learning via probabilistic modeling in the ECGA. Technical report, Illinois Genetic Algorithms Laboratory, 1999.
- [58] Robert B. Heckendorn and Alden H. Wright. Efficient linkage discovery by limited probing, 2003.

- [59] Yuan-Wei Huang and Ying-ping Chen. On the detection of general problem structures by using inductive linkage identification. In GECCO '09: Proceedings of the 11th Annual conference on Genetic and evolutionary computation, pages 1853–1854, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-325-9. doi: http://doi.acm.org/10.1145/1569901.1570200.
- [60] Andrew W. Moore. K-means and hierarchical clustering. On-line from website: http://www.cs.cmu.edu/awm/tutorials, November 2001. This is a comprehensable tutorial for clustering, it introduces k-means and Hierarchical clustering—Single Linkage Hierarchical Clustering.
- [61] Masaharu Munetomo, Masaharu Munetomo, David E. Goldberg, and David E. Goldberg. Linkage identification by non-monotonicity detection for overlapping functions. *Evolutionary Computation*, 7:377–398, 1999.
- [62] Martin Pelikan, David E. Goldberg, and Erick Cantú-Paz. Linkage problem, distribution estimation, and Bayesian networks. *Evolutionary Computation*, 8(3):340, 2000.
- [63] Dirk Thierens. The linkage tree genetic algorithm. In Robert Schaefer, Carlos Cotta, Joanna Kolodziej, and Günter Rudolph, editors, Parallel Problem Solving from Nature – PPSN XI, volume 6238 of Lecture Notes in Computer Science, pages 264–273. Springer Berlin / Heidelberg, 2011. URL http://dx.doi.org/10.1007/978-3-642-15844-5_27. 10.1007/978-3-642-15844-5_27.
- [64] Dirk Thierens. Linkage tree genetic algorithm: first results. In GECCO '10: Proceedings of the 12th annual conference comp on Genetic and evolutionary computation, pages 1953–1958, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0073-5. doi: http://doi.acm.org/10.1145/1830761.1830832.
- [65] Miwako Tsuji and Masaharu Munetomo. Linkage analysis in genetic algorithms. In Lakhmi C. Jain, Mika Sato-Ilic, Maria Virvou, George A. Tsihrintzis, Valentina Emilia Balas, and Canicious Abeynayake, editors,

- Computational Intelligence Paradigms, volume 137 of Studies in Computational Intelligence, pages 251–279. Springer, 2008. ISBN 978-3-540-79473-8. URL http://www.springerlink.com/content/n714123130160303/.
- [66] Miwako Tsuji, Masaharu Munetomo, and Kiyoshi Akama. Linkage identification by fitness difference clustering. *Evolutionary Computation*, 14(4): 383–409, 2006. URL http://dblp.uni-trier.de/db/journals/ec/ec14. html#TsujiMA06.
- [67] Masaharu Munetomo and David E. Goldberg. Linkage identification by non-monotonicity detection for overlapping functions. *Evolutionary Computation*, 7(4):377–398, 1999.
- [68] Xingyan Jiang, Yuanzhu Chen, and Tina Yu. Dynamic cooperative coevolutionary sensor deployment via localized fitness evaluation. In Günter Rudolph, Thomas Jansen, Simon Lucas, Carlo Poloni, and Nicola Beume, editors, Parallel Problem Solving from Nature PPSN X, volume 5199 of Lecture Notes in Computer Science, pages 225–235. Springer Berlin / Heidelberg, 2008. URL http://dx.doi.org/10.1007/978-3-540-87700-4_23. 10.1007/978-3-540-87700-4_23.