國立交通大學

資訊科學與工程研究所

碩 士 論 文

論分佈估計演算法中之可視鏈結、
有效分佈、與模型刪改

Sensible Linkage, Effective Distributions, and Model
Pruning in Estimation of Distribution Algorithms

研 究 生：莊仲堯

指導教授：陳穎平　教授

中 華 民 國 九 十 七 年 七 月

論分佈估計演算法中之可視鏈結、有效分佈、與模型刪改
Sensible Linkage, Effective Distributions, and Model Pruning
in Estimation of Distribution Algorithms

研 究 生：莊仲堯　　　　　Student：Chung-Yao Chuang

指導教授：陳穎平　　　　　Advisor：Ying-ping Chen

國 立 交 通 大 學
資 訊 科 學 與 工 程 研 究 所
碩 士 論 文

A Thesis

Submitted to Institute of Computer Science and Engineering

College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science

July 2008

Hsinchu, Taiwan, Republic of China

中 華 民 國 九 十 七 年 七 月

論分佈估計演算法中之可視鏈結、有效分佈、與模型刪改

學生：莊仲堯　　　　　　　　　　　　指導教授：陳穎平

國立交通大學資訊科學與工程研究所

摘　　要

　　分佈估測演算法(Estimation of Distribution Algorithms, EDAs)是一種演化式計算的方法，此種演算法利用建立機率模型的方式去自動偵測決策變數之間的關聯，這種自動判斷決策變數之間關聯的能力，能使最佳化程序以較好的方式搜尋解空間。在這篇論文中，我們探討分佈估測演算法的一個弱點：在最佳化問題中，部份子問題有重要程度不同時，演算法會浪費部份的搜尋在不確定的解空間；由此，我們提出一個改善的方法，我們捨棄一般的完整機率建模，改為只使用完整機率模型中利於有效搜尋的部份，而得到一個節省的搜尋策略。發展出的方法利用統計上的性質去辨認出機率模型中可能對於有效搜尋無幫助的部份，進而刪除之，只使用剩餘的部份模型做解空間的搜尋。實驗結果顯示，所提的方式在部份子問題有重要程度不同時，能夠有效的降低搜尋最佳解所需的代價。

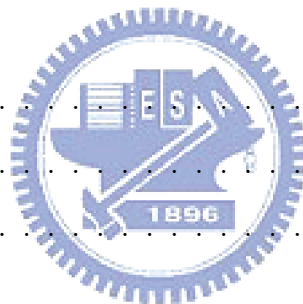關鍵字：分佈估測演算法、基因演算法、演化計算、機率模型、模型刪改、
　　　　可視鏈結、黑箱最佳化問題

Abstract

Estimation of distribution algorithms (EDAs) are a class of evolutionary algorithms that replace the traditional variation operators, such as mutation and crossover, by building a probabilistic model on promising solutions and sampling the built model to generate new candidate solutions. Using probabilistic models for exploration enables these methods to use advanced techniques of statistics and machine learning for automatic discovery of problem structures. However, in some situations, complete and accurate identification of all problem structures by probabilistic modeling is not possible because of certain inherent properties of the given problem. In this work, we illustrate one possible cause of such situations with problems composed of structures of unequal fitness contributions. Based on the illustrative example, a notion is introduced that the estimated probabilistic models should be inspected to reveal the effective search directions, and we propose a general approach which utilizes a reserved set of solutions to examine the built model for likely inaccurate fragments. Furthermore, the proposed approach is implemented in the extended compact genetic algorithm (ECGA) and experimented on several sets of problems with different scaling difficulties. The results indicate that the proposed method can significantly assist ECGA to handle problems comprising structures of disparate fitness contributions and therefore may potentially help EDAs in general to overcome those situations in which the entire structure of the problem cannot be recognized properly due to the temporal delay of emergence of some promising partial solutions.

keywords:

Sensible linkage, effective distribution, linkage sensibility, probabilistic model, model pruning, estimation of distribution algorithm, extended compact genetic algorithm, genetic algorithms, evolutionary algorithm, evolutionary computation, black-box optimization.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Over the last decade, a lot of research efforts in the field of evolutionary computation have been devoted to optimization methods that build probabilistic models on promising solutions and sample the built model to generate new candidate solutions. These methods are called *estimation of distribution algorithms* (EDAs) [1, 2, 3]. By using probabilistic models for exploration, EDAs are able to utilize advanced methods of machine learning and statistics to automatically capture the likely structure of promising solutions and exploit the identified problem regularities to facilitate further search. Thus, EDAs are often deemed as efficient and reliable techniques to tackle many optimization problems, especially when the problem specific knowledge is not available.

1.1 EDAs in A Nutshell

Similar to other evolutionary algorithms, the procedure of EDAs, as listed in Algorithm 1, starts from initializing a population of solutions which can be randomly generated or produced with some heuristics. Then, the algorithm iterates through the selection process, which picks a set of promising solutions from the current population, and the creation of new candidate solutions, which recombines the set of selected solutions to form new solutions. The prominent feature of EDAs is that the recombination of solutions is done by building and sampling probabilistic models. From an abstract perspective, the selected set of solutions can be viewed as samples drawn from an unknown probability distribution. Knowing that distribution would allow the optimization method to generate new solutions that are somehow similar to the ones contained in the original selected set of promising

---
Algorithm 1 General Outline of Estimation of Distribution Algorithms
---
   Initialize a population P with n solutions.
   while the stopping criteria are not met **do**
       Evaluate the solutions in P.
       $P' \leftarrow$ apply selection on P.
       $M \leftarrow$ build a probabilistic model on P'.
       $O \leftarrow$ generate new candidate solutions by sampling M.
       Incorporate O into P.
   end while
---

solutions.

Although the actual probability distribution is unknown, there are techniques capable of estimating that distribution by using the set of selected solutions. Using these estimation techniques, we can obtain probabilistic models that reflect the possible formation of good solutions. In other words, an accurate distribution estimate is able to capture the structure, or the "building blocks" [4, 5], of the problem[1]. By sampling this distribution, we can ensure an effective mixing and reproduction of building blocks.

1.2 Research Objectives

In most studies, it is presumed that EDAs can detect the structure of the problem by recognizing the regularities existed within the promising solutions. However, as illustrated in the later chapters, for certain problems, EDAs are actually unable to identify the entire structure of the problem because the set of selected solutions on which the probabilistic model is built contains insufficient information regarding some parts of the problem and renders EDAs incapable of processing these parts accurately.

In this thesis, we start at observing the evolutionary process of an EDA in dealing with an exponentially scaled problem and recognizing that the population on which the probabilistic model is built does not necessarily contain sufficient information for all problem structures to be detected completely and accurately. Based on the observation, a general concept is proposed that the estimated probabilistic models should be inspected to reveal the effective search directions, and we provide a practical approach which utilizes a reserved set of solutions to examine the built model for the fragments that may be

---

[1]As argued later in this thesis, this statement is not entirely true. However, this implicit assumption was pervasively accepted in the research community of EDAs.

inconsistent with the actual problem structure. Furthermore, the proposed approach is implemented in the extended compact genetic algorithm [6] and experimented on several sets of problems with different scaling difficulties [5] to demonstrate the applicability.

1.3 Road Map

In the next chapter, we will briefly review the research topics concerning this study. After that, chapter 3 demonstrates the interaction between the scaling difficulty and probabilistic model building performed by EDAs. More specifically, we will investigate how the scaling difficulty shadows the ability of EDAs to recognize problem structures and causes inaccurate processing on some parts of the solutions. Accordingly, a general approach will be proposed to resolve such a problem and enforce accurate processing during the optimization process. In chapter 4, an implementation of the proposed approach in the extended compact genetic algorithm will be detailed. Chapter 5 presents the empirical results, followed by the discussion and observations on the results. Finally, chapter 6 concludes this thesis.

Chapter 2

Background

An optimization problem can be defined by specifying (1) the set of all potential solutions to that problem and (2) a measure to evaluate the quality of each possible solution that reflects the objectives. The goal is to search for solutions that maximize the specified quality measure. An interesting and difficult class of optimization problems is called *black-box optimization problems*, in which there is no problem-specific domain knowledge available besides the quality measure (the objective function). The only way of learning something about the relation between semantics of solutions and the quality measure is to sample new candidate solutions and evaluate them using the objective function.

A popular way to deal with black-box optimization problems is by applying genetic algorithms (GAs) [7, 4]. There are several features of GA that make it particularly suitable for these problems: only objective function required, population-based search, and exploration by combining pieces of promising solutions. Because of the simplicity of the idea and its wide applicability, GAs are becoming an increasingly important area of computational optimization.

This chapter starts with a short review of GAs. Following that, Section 2.2 describes a heavily studied problem that occurs when applying genetic algorithms called *building block disruption problem*. Also, a briefly review of research efforts to that problem is given. Especially, we emphasize on the application of estimation of distribution algorithms (EDAs) to overcome the building block disruption problem. Then, in Section 2.3, we move to another topic concerning this study — the so-called *scaling difficulty*. Finally, Section 2.4 points out the research direction of this study.

2.1 Genetic Algorithms

Genetic algorithms are search techniques loosely based on the paradigm of natural evolution, in which species of creatures tend to adapt to their living environments by mutation and inheritance of useful traits. Genetic algorithms mimic this mechanism by introducing artificial selections and genetic operators to discover and recombine partial solutions. By properly growing and mixing promising partial solutions, which are often referred to as building blocks (BBs) [5], GAs are capable of efficiently solving a host of problems. The ability of implicitly processing a large number of partial solutions has been recognized as an important source of the computational power of GAs. According to the Schema theorem [7], short, low-order, and highly fit sub-solutions increase their share to be combined, and also as stated in the building block hypothesis [4], GAs implicitly decompose a problem into sub-problems by processing building blocks. This decompositional bias is a good strategy for tackling many real-world problems, because real-world problems can oftentimes be reliably solved by combining the pieces of promising solutions in the form of problem decomposition.

2.2 Building Block Disruption Problem

Unfortunately, proper growth and mixing of building blocks are not always achieved. GA in its simplest form employing fixed representations and problem-independent recombination operators often breaks the promising partial solutions while performing crossovers. This can cause the vanishing of crucial building blocks and thus lead to the convergence to local optima.

In order to overcome the building block disruption problem, a variety of techniques have been proposed and developed, which can be roughly classified into three categories:

1. Evolving representations or operators;

2. Perturbation methods.

3. Probabilistic modeling for promising solutions;

The objective of the first class of techniques is to manipulate the representation of solutions during the search process such that members of the promising sub-solutions are less likely to be separated by crossover operators. Various reordering and mapping operators were proposed. In this line of research, the messy GA (mGA) [8] and its more efficient descendant—the fast messy GA (fmGA) [9]—identify linkage by exploiting building blocks. The problem of techniques in this category is that reordering operators are often too slow and lose the race against selection, resulting in the premature convergence to local optima. Another technique in this category, the linkage learning GA (LLGA) [10], employs a two-point crossover over circular representation of strings to maintain tight linkage. While LLGA works well on exponentially scaled problems, it is inefficient in handling uniformly scaled problems [10] [11].

The methods in the second category examine the fitness differences by conducting perturbations on the variables to detect dependencies among them. For example, the gene expression messy GA (GEMGA) [12] employs a perturbation method to detect the sets of tightly linked variables represented by weight values assigned to each solution. GEMGA records fitness changes caused by perturbation of every variable for strings in the population and detects relations among variables according to the possibilities that the variables may construct the local optima. Linkage identification by nonlinearity check (LINC) [13] detects nonlinearity by using pairwise perturbations in order to identify the linkage information. It assumes that nonlinearity exists within variables to form a building block. If the fitness difference by simultaneous perturbations at a pair of variables is equal to the sum of fitness differences by perturbation at each variable in the pair, then these variables can be viewed as to reside within different and independent subproblems, and therefore, these variables can be optimized separately. Linkage information identified by LINC is represented as sets of variables. Each set contains tightly linked variables forming a building block and such a set is called a linkage set. The descendant of LINC, linkage identification by non-monotonicity detection (LIMD) [14], adopts non-monotonicity instead of nonlinearity and detects linkage by checking violations of the monotonicity conditions. Although perturbation methods require extra fitness function evaluations in addition to the running of GA, they have the advantage of being able to identify low

salience building blocks. Heckendorn and Wright [15] generalized this category through a Walsh analysis.

In this study, we focus on the approaches in the third category which are often called the estimation of distribution algorithms (EDAs) [1, 2, 3]. These methods construct probabilistic models of promising solutions and utilize the built models to generate new solutions. Early EDAs, such as the population-based incremental learning (PBIL) [16] and the compact genetic algorithm (cGA) [17], assume no interaction between decision variables, i.e., decision variables are assumed independent of each other. Subsequent studies start from capturing pairwise interactions, such as mutual-information-maximizing input clustering (MIMIC) [18], Baluja's dependency tree approach [19], and the bivariate marginal distribution algorithm (BMDA) [20], to modeling multivariate interactions, such as the extended compact genetic algorithm (ECGA) [6], the Bayesian optimization algorithm (BOA) [21], the estimation of Bayesian network algorithm (EBNA) [22], the factorized distribution algorithm (FDA) [23], and the learning version of FDA (LFDA) [24]. With the reasoning of dependencies among variables by building probabilistic models, these approaches can capture the structure of the problem and thus avoid the disruption of identified partial solutions.

For example, consider a 4-bit trap function that takes a binary string of length 4,

$$
f_{trap_4}(x_1x_2x_3x_4) = \begin{cases} 4, & \text{if u} = 4; \\ 3 - u, & \text{otherwise.} \end{cases} ,
$$

where u is the number of ones in the string $x_1x_2x_3x_4$. Suppose that we are dealing with a 16-bit maximization problem formed by concatenating four 4-bit trap functions,

$$
f(s_1s_2\cdots s_{16}) = \sum_{i=0}^{3} f_{trap_4}(s_{4i+1}s_{4i+2}s_{4i+3}s_{4i+4}) ,
$$

where $s_1s_2\cdots s_{16}$ is a solution string and the variables to the same sub-function are considered as strongly related. Assume that we use ECGA [6], which uses a class of multivariate probabilistic models called marginal product models (MPMs)[1], to tackle this problem. By observing subsequent generations of the optimization process, a series of models built by

---

[1]See Chapter 4 for more detailed information about ECGA and marginal product models.

Generation	Marginal Product Model
1	[1 2 3 4] [5 6 7 8] [9 10 11 12] [13 14 15 16]
2	[1 2 3 4] [5 6 7 8] [9 10 11 12] [13 14 15 16]
3	[1 2 3 4] [5 6 7 8] [9 10 11 12] [13 14 15 16]
⋮	... ...

Table 2.1: Marginal product models built by ECGA when solving a uniformly scaled problem. The variables are denoted by their indexes. Each group of variables represents a marginal model in which a marginal distribution resides.

ECGA can be obtained like those listed in Table 2.1. From the table, we can see that the built model capture the structure of the problem properly from the first generation (i.e., the model building algorithm reasoned a probabilistic model that has the structure consistent with the decomposition of the problem.) and it continued to identify the building blocks correctly. By sampling the probabilistic model that is consistent with the problem structure, new candidate solutions can be generated in a BB-wise way. As a result, the building block disruption can be avoid.

2.3 Scaling Difficulty

Another topic concerning this study is the impact of disparate scale among different building blocks on the behavior and performance of the evolutionary algorithms. It is commonly observed that building blocks with higher marginal fitness contributions – salient building blocks – converge before those with lower marginal fitness contributions. This sequential convergence behavior is often referred to as domino convergence [25].

For the real-world applications, it is often the case that some parts of the problem are more prominent and contribute more to the fitness evaluation than other parts. Such a situation can pose two types of difficulties. Firstly, because the processing in the population is statistical in nature, the disparate scaling can cause inaccurate processing of less salient building blocks [26, 27]. The second difficulty arises because the lower salience of a building block generally causes it to be processed at a later time compared to those of higher salience. This delay on timeline can cause the building block to converge under random pressures, instead of properly selective ones.

Other previous studies on this topic include the explicit role of scalings in a systematic experimental setting [28], a theoretical model on convergence behavior of exponentially scaled problems [25], an extension of that model to building blocks of length more than one variable [29] and a convergence model of linkage learning genetic algorithm (LLGA) [10] on problems of different scaling difficulties [30].

2.4 Direction

Although the above mentioned scaling difficulty exists in a number of problems and degrades the performance of many evolutionary algorithms, there are scant investigations concerning the behavior of EDAs with the presence of scaling difficulties. In this study, we make an attempt to explore how the scaling difficulty affects EDAs and propose a practical countermeasure to assist EDAs on problems with different scalings. Specifically, we propose a notion that the estimated probabilistic models should be examined to enforce accurate processing of building blocks and prevent random drifting from taking place.

In the remainder of this thesis, our approach will be demonstrated and evaluated on the test problems constructed by concatenating several trap functions. Specifically, a k-bit trap function is a function of unitation[2] which can be expressed as

$$f_{trap_k}(x_1 x_2 \cdots x_k) = \begin{cases} k, & \text{if u = k ,} \\ k - 1 - u, & \text{otherwise.} \end{cases},$$

where u is the number of ones in the binary string $x_1 x_2 \cdots x_k$. The trap functions were used pervasively in the studies concerning EDAs and other evolutionary algorithms because they provide well-defined structures among variables, and the ability to recognize inter-variable relationships is essential to solve the problems consisting of traps [31, 32].

---

[2]A function of unitation is a function whose value depends only on the number of ones in the binary input string.

Chapter 3

From Sensible Linkage
to Effective Distributions

The ability of EDAs to handle the building block disruption problem comes primarily from the explicitly modeling of selected, promising solutions by using probabilistic models. The model construction algorithms, though differ in their representative power, capture the likely structures of good solutions by processing the population-wise statistics collected from the selected solutions. By reasoning the dependencies among different parts of the problem and the possible formations of good solutions, reliable mixing and growing of building blocks can be achieved. As noted in [6], learning a good probability distribution is equivalent to learning linkage, where linkage refers to the dependencies among variables or equivalently the decomposition of the problem.

3.1   An Illustrative Example

In most studies on EDAs, it is presumed that EDAs can detect linkage by recognizing building blocks according to the information contained in the set of selected solutions. However, in this study, we argue that in some situations, accurate and complete linkage information can not be acquired by distribution estimation because the selected set of solutions on which the model is built contains insufficient information on the less salient parts of the problem. For example, consider a 16-bit maximization problem,

$$f(s_1 s_2 \cdots s_{16}) = \sum_{i=0}^{3} \left(5^{3-i} f_{trap_4}(s_{4i+1} s_{4i+2} s_{4i+3} s_{4i+4}) \right) \,,$$

where $s_1 s_2 \cdots s_{16}$ is a solution string. Note that different from the example presented in Section 2.2 and many other studies in EDAs in which the test problems are uniformly

scaled, i.e., subproblems are of equal salience, in this problem, each elementary trap function is scaled exponentially. This scaling is an abstraction for problems of distinguishable prominence or solving priority among the constituting subproblems. Again, suppose that we choose ECGA [6], which uses a class of multivariate probabilistic models called marginal product models (MPMs), to deal with this problem. By observing subsequent generations of the optimization process, a series of models built by ECGA can be obtained like those listed in Table 3.1. In this table, variables are denoted by their index numbers. Each group of variables represents a marginal model in which a marginal distribution resides and the converged variables are crossed out.

It can be observed that the models shown in Table 3.1 are only partially correct. More specifically, in each generation, only the most salient building block on which the population have not converged is correctly modeled[1]. This is caused by the fact that some part of the problem contributes much more than all others in combine. If one part of the problem is worth essentially more than others, then this part of the solution solely determines the chance regarding whether or not the solution will be selected. As a consequence, only the most salient building block can provide sufficient information to be modeled correctly, since the model searching is performed based on the selected solutions. The rest parts of the model are primarily the result of low salience partial solutions "hitchhiking" on the more salient building blocks.

3.2 Sensible Linkage

From the example demonstrated in the previous section, we can see that not all building blocks can be detected from a given set of selected solutions by probabilistic model building. Model building algorithms can not "see" the entire structure of the problem from the selected set of solutions because disparate scalings among different building blocks prevents complete linkage information from being included in the selected population. In this work, we will refer to this concept as *linkage sensibility* and those problem structures that can be identified properly using the given set of solutions are called *sensible linkage*.

---

[1]Note that because ECGA uses minimum description length (MDL) principle to guide the model searching, the converged variables are not modeled jointly. This is completely alright considered they are converged. See Chapter 4 for more detailed information.

Generation	Marginal Product Model	Effective Partial Model
1	[1 2 3 4] [5 10 16] [6 7] [8 9 12] [11 14 15] [13]	[1 2 3 4]
2	~~[1 2 3 4]~~ [5 6 7 8] [9 13 16] [10 14 15] [11 12]	[5 6 7 8]
3	~~[1 2 3 4]~~ ~~[5 6 7 8]~~ [9] [10 11 12 13 16] [14 15]	[9 10 11 12]
4	~~[1 2 3 4]~~ ~~[5 6 7 8]~~ ~~[9]~~ ~~[10 11 12]~~ [13 14 15 16]	[13 14 15 16]

Table 3.1: Marginal product models built by ECGA when solving an exponentially scaled problem. The variables are denoted by their indexes. Each group of variables represents a marginal model in which a marginal distribution resides. The converged variables are crossed out. The third column shows the effective marginal models which are consistent with the problem structure.

From this notion, we can re-examine EDAs on the building block disruption problem. It is clear that the disruption problem still exists in the insensible portion of the problem because that part of the problem can not be modeled properly. Though the above example is an extreme case of scalings that each subproblem is exponentially scaled, in real-world problems, it is still oftentimes the case that the constituting subproblems are weighted differently which implies the linkage might be just partially sensible. In addition to the building block disruption problem, the random drifting of the less salient parts of the problem mentioned in chapter 2 even worsen the situation. Those problems are usually handled by increasing population sizes when EDAs are applied. However, we can deal with this situation in another way if it is possible to distinguish sensible linkage from insensible ones.

3.3 Effective Distributions

The idea of sensible linkage can be closely mapped into another notion called *effective distributions*. By effective distributions, we mean that by sampling these distributions, the solution quality can be reliably advanced. Thus, the essential conditions for effective distributions are

- the consistency with building blocks, and

- the provision of good directions for further search.

If it is possible to extract effective distributions from the built probabilistic model, we can perform partial sampling using only the effective distributions and leave the rest parts of the solutions unchanged. Thus, the diversity is maintained and we are free from the building block disruption and random drifting problems. For instance, returning to the earlier 16-bit optimization problem, if it is possible to identify those partial models which are built on the sensible linkage like [1 2 3 4] in the first generation and [5 6 7 8] in the second generation (see the third column of Table 3.1), we can sample only the corresponding marginal distributions which are, in this case, effective. That is, in the first generation, for each solution string, we re-sample only $s_1 s_2 s_3 s_4$ according to the

marginal distribution and keep $s_5 s_6 \cdots s_{16}$ unchanged. In the second generation, we re-sample only $s_5 s_6 s_7 s_8$ according to the marginal distribution and keep $s_9 s_{10} \cdots s_{16}$ with the same values ($s_1 s_2 s_3 s_4$ are converged). In this way, we do not have to resort to increasing population sizes to deal with the problem caused by the disparate building block scalings.

The above thoughts leave us one complication: the identification of effective distributions. However, direct identification of effective distributions may not be an easy task if not impossible. Thus, it may be wise to adopt a complementary approach – to identify those distributions that are *not* likely to be effective. If there is a way to identify the ineffective distributions, we can bypass them and sample only the rest distributions and thus, approximate the result of knowing effective distributions. Our idea is that if we split the entire population into two sub-populations and use only one sub-population for building probabilistic model, we can utilize the second sub-population to collect statistics for possible indications of ineffectiveness of certain partial distributions in the probabilistic model built on the first sub-population. That is, with certain appropriate heuristics or criterion, we can prune the likely ineffective portions of the model.

In the next chapter, our implementation of the proposed concept in ECGA will be detailed. More specifically, a judging criterion will be proposed to detect the likely ineffective marginal distributions of a given marginal product model.

Chapter 4

ECGA with Model Pruning

This chapter starts at reviewing the extended compact genetic algorithm. Based on the idea of detecting the inconsistency of statistics gathered from the two sub-populations, a mechanism is devised to identify the possibly ineffective parts of the built probabilistic model. Finally, an optimization algorithm incorporating the proposed technique is described in detail.

4.1   Extended Compact Genetic Algorithm

The extended compact genetic algorithm (ECGA) [6] uses a product of marginal distributions on a partition of the variables. This kind of probability distribution belongs to a class of probabilistic models known as marginal product models (MPMs). In this kind of model, subsets of variables can be modeled jointly, and each subset is considered independent of other subsets. In this work, the conventional notation is adopted that variable subsets are enclosed in brackets. Table 4.1 presents an example of MPM defined over four variables: s_1, s_2, s_3 and s_4. In this example, s_2 and s_4 are modeled jointly and each of the three variable subsets ($[s_1]$, $[s_2 \ s_4]$ and $[s_3]$) is considered independent of other subsets. For instance, the probability that this MPM generates a sample $s_1s_2s_3s_4 = 0101$ is calculated as follows,

$$P(s_1s_2s_3s_4 = 0101) = P(s_1 = 0) \times P(s_2 = 1, s_4 = 1) \times P(s_3 = 0)$$

$$= 0.4 \times 0.4 \times 0.5.$$

In fact, as its name suggested, a marginal product model represents a distribution that is a "product" over the marginal distributions defined over variable subsets.

$[s_1]$	$[s_2\ s_4]$	$[s_3]$
$P(s_1 = 0) = 0.4$	$P(s_2 = 0, s_4 = 0) = 0.4$	$P(s_3 = 0) = 0.5$
$P(s_1 = 1) = 0.6$	$P(s_2 = 0, s_4 = 1) = 0.1$	$P(s_3 = 1) = 0.5$
	$P(s_2 = 1, s_4 = 0) = 0.1$	
	$P(s_2 = 1, s_4 = 1) = 0.4$	

Table 4.1: An example of marginal product model that defines a joint distribution over four variables. The variables enclosed in the same brackets are considered dependent and modeled jointly. Each variable subset is considered independent of other variable subsets.

In ECGA, both the structure and the parameters of the model are searched and optimized with a greedy approach to fit the statistics of the selected set of promising solutions. The measure of a good MPM is quantified based on the minimum description length (MDL) principle [33], which assumes that given all things are equal, simpler distributions are better than complex ones. The MDL principle thus penalizes both inaccurate and complex models, thereby, leading to a near-optimal distribution. Specifically, the search measure is the MPM complexity which is quantified as the sum of model complexity, C_m, and compressed population complexity, C_p. The greedy MPM search first considers all variables as independent and each of them forms a separate variable subset. In each iteration, the greedy search merges two variable subsets that yields the most $C_m + C_p$ reduction. The process continues until there is no further merge that can decrease the combined complexity.

The model complexity, C_m, quantifies the model representation in terms of the number of bits required to store all the marginal distributions. Suppose that the given problem is of length ℓ with binary encoding, and the variables are partitioned into m subsets with each of size k_i, $i = 1 \ldots m$, such that $\ell = \sum_{i=1}^{m} k_i$. Then the marginal distribution corresponding to the ith variable subset requires $2^{k_i} - 1$ frequency counts to be completely specified. Taking into account that each frequency count is of length $\log_2(n+1)$ bits, where n is the population size, the model complexity, C_m, can be defined as

$$C_m = \log_2(n + 1) \sum_{i=1}^{m} \left(2^{k_i} - 1 \right) .$$

The compressed population complexity, C_p, quantifies the suitability of the model in terms of the number of bits required to store the entire selected population (the set of

promising solutions picked by selection operator) with an ideal compression scheme applied. The compression scheme is based on the partition of the variables. Each subset of the variables specifies an independent "compression block" on which the corresponding partial solutions are optimally compressed. Theoretically, the optimal compression method encodes a message of probability p_i using $-\log_2 p_i$ bits. Thus, taking into account all possible messages, the expected length of a compressed message is $\sum_i -p_i \log_2 p_i$ bits, which is optimal. In the information theory [34], the quantity $-\log_2 p_i$ is called the *information* of that message and $\sum_i -p_i \log_2 p_i$ is called the *entropy* of the corresponding distribution. Based on the information theory, the compressed population complexity, C_p, can be derived as

$$C_p = n \sum_{i=1}^{m} \sum_{j=1}^{2^{k_i}} -p_{ij} \log_2 p_{ij} \ ,$$

where p_{ij} is the frequency of the jth possible partial solution to the ith variable subset observed in selected population.

Note that in the calculation of C_p, it is assumed that the jth possible partial solution to the ith variable subset is encoded using $-\log_2 p_{ij}$ bits. This assumption is fundamental to our technique to identify the likely ineffective marginal distributions. More precisely, the information of the partial solutions, $-\log_2 p_{ij}$, is a good indicator of inconsistency of statistics gathered from two separate sub-populations.

4.2 Model Pruning

Our technique to identify the possibly ineffective fragments of a marginal product model is based on the notion that ECGA uses the compression performance to quantify the suitability of a probabilistic model for the given set of solutions. The degree of compression is a quite representative metric to the fitness of modeling, because all good compression methods are based on capturing and utilizing the relationships among data. Thus, if the compression scheme of the MPM built on one set of solutions is incapable of compressing another set of solutions produced under the same condition, then it is very likely that the obtained MPM is, at least, partially incorrect. Using this property, we can perform a systematical checking on the given MPM for the likely ineffective portions.

Suppose that the population of solutions, P, is split into two sub-populations S and T. The model searching is performed on S', the set of promising solutions selected from S. Then we can use the statistics collected from T', the set of solutions selected from T, to examine the built probabilistic model, M. Since each marginal model functions independently, they can be inspected separately. Recalling the former description that a variable subset, which specifies a marginal model, is viewed as a "compression block" that encodes each possible partial solution according to the marginal distribution. That is, the jth possible partial solution to the ith variable subset is encoded using $-\log_2 p_{ij}$ bits, where p_{ij} is the frequency of the jth possible partial solution to the ith variable subset observed in S'. Assume that the given problem is of length ℓ with binary encoding, and there are m variable subsets with each of size k_i, $i = 1 \dots m$, in the built model M. For the ith marginal model, $i = 1 \dots m$, we can check whether or not

$$\sum_{j=1}^{2^{k_i}} q_{ij}(-\log_2 p_{ij}) > k_i \; ,$$

where q_{ij} is the frequency of the jth possible partial solution to the ith variable subset collected from T'. If the inequality holds, then the compression scheme employed in the ith marginal model is not a good one for compressing the corresponding partial solutions in T', because it encodes a k_i-bit partial solution to a bit string of expected length more than k_i bits. Using the earlier reasoning, such a condition indicates that the marginal model is likely ineffective because T' does not agree on this part of the modeling. Otherwise, it should be able to compress the partial solutions in T'.

Explained from a machine learning perspective [35], a good model should generalize well to the unseen instances. Otherwise, it captures coincidental regularities among training data. If the model building is performed on the portion where linkage is not sensible from the given set of solutions, it will "overfit" to those partial solutions (hitchhikers) that were not subjected to proper selection pressures. Consequently, the regularities captured by this part of modeling tend to be inconsistent with the true problem structure. Furthermore, the partial solutions that were not subjected to proper selection pressures appear to be random, and it brings about the phenomenon of random drifting mentioned in chapter 2. By its nature, the drifting is random, and two different sub-populations

tend to drift in two different directions. Thus, we can use the statistical inconsistency between S' and T' to locate possible drifting portions of the solutions and identify the likely ineffective parts of the model. By removing the likely ineffective parts, we can forge a partial but more effective model.

An issue in practice concerning the calculation of the inequality is that sometimes one or several possible partial solutions are absent in the set of selected solutions, and leave $-\log_2 p_{ij}$ undefined because $p_{ij} = 0$. Currently, we handle this practical problem by assigning a very small value, smaller than $1/n$, to the p_{ij}'s that are zero and normalizing them such that p_{ij}'s are sum to 1 (that is, $\sum_j p_{ij} = 1$).

4.3 Integration

In this section, the optimization process incorporating ECGA and the proposed technique is described. This combination helps ECGA to achieve better performance where disparate scalings exist among different parts of the problem.

The procedure is presented in Algorithm 2. This process starts at initializing a population of solutions. After initialization, the solutions are evaluated, and then the entire population is randomly split into two sub-populations. Selection operations are performed on the two sub-populations separately with the same selection pressure. Model building is performed on one of the sub-populations. The other sub-population is used to prune the built model using the technique previously proposed. Finally, all solutions in the population are altered by sampling the remaining marginal distributions, which are considered effective, in the pruned model. These steps are repeated until the stopping criteria are met.

A prominent difference between the above process and the regular EDAs is that the sampling may not include all decision variables. As introduced in section 3.3, the existing solutions are altered by sampling only the marginal distributions surviving the model pruning process. Thus, a solution string might not be entirely modified in an iteration. This technique hence avoids random drifting and inaccurate processing of low-salience building blocks by postponing the processing until the sufficient linkage information is

---
Algorithm 2 ECGA with Model Pruning

Initialize a population P with n solutions of length ℓ.

while the stopping criteria are not met **do**

    Evaluate the solutions in P.

    Divide P into S and T at random.

    $S' \leftarrow$ apply t-wise tournament selection on S.

    $T' \leftarrow$ apply t-wise tournament selection on T.

    $M \leftarrow$ build the MPM on S' with greedy search.

    $M' \leftarrow$ prune M based on the inconsistency with T'.

    for each remaining marginal distribution D in M' **do**

        for each solution $\mathbf{s} = s_1 s_2 \cdots s_\ell$ in P **do**

            Change the values in $\mathbf{s}$ partially by sampling D.

        end for

    end for

end while

---

available. In this way, better performance in terms of function evaluations can be achieved if disparate scalings exist among different parts of the problem.

To see that the proposed method meets its design purpose, Table 4.2 lists the models before and after pruning when the earlier exponentially scaled problem is solved by Algorithm 2. It can be seen that the proposed approach appropriately removes the ineffective parts during each stage of the optimization process[1]. To further illustrate the behavior of the proposed approach, the algorithm is applied to another problem with a different scaling called *overloaded scaling*[2],

$$
f(s_1 s_2 \cdots s_{16}) = \sum_{i=0}^{1} f_{trap_4}(s_{4i+1} s_{4i+2} s_{4i+3} s_{4i+4}) +
$$
$$
\sum_{i=2}^{3} \frac{1}{5} f_{trap_4}(s_{4i+1} s_{4i+2} s_{4i+3} s_{4i+4}) ,
$$

where $s_1 s_2 \cdots s_{16}$ is a solution string. The overloaded cases are those with two kinds of scalings, where some subproblems are at the high level and the rest are at the low one. The models before and after pruning when such a problem is solved are shown in Table 4.3. It can be observed that the proposed method works as expected in splitting the solving process according to the scaling structure. The two subproblems of higher salience are handled first, and the two subproblems of lower salience are solved later.

---

[1]Note that the converged variables are not modeled jointly because ECGA uses minimum description length principle to guide the model searching. This is completely alright considered they are converged.

[2]As mentioned in [5], the word "overloaded" is a reference to the application of this idea in the early messy GA work [28], where such distributions were used to try to overload or overwhelm the ability of the messy GA to keep all building blocks present through all phases of the process.

Generation	Before Pruning	After Pruning
1	[1 2 3 4] [5 13 16] [6 7 12] [8 11] [9 10] [14 15]	[1 2 3 4]
2	[1 2 3 4] [5 6 7 8] [9 14] [10 15] [11 13 16] [12]	[1] [2] [3] [4] [5 6 7 8]
3	[1] [2] [3] [4] [5] [6] [7] [8] [9 10 11 12] [13 14] [15 16]	[1] [2] [3] [4] [5] [6] [7] [8] [9 10 11 12]
4	[1] [2] [3] [4] [5] [6] [7] [8] [9] [10] [11] [12] [13 14 15 16]	[1] [2] [3] [4] [5] [6] [7] [8] [9] [10] [11] [12] [13 14 15 16]

Table 4.2: Marginal Product Models before and after pruning when the 16-bit exponentially scaled problem is solved by the proposed approach.
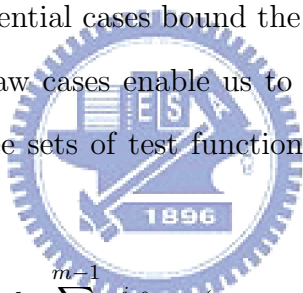
Generation	Before Pruning	After Pruning
1	[1 2 3 4] [5 6 7 8] [9 16] [10 14 15] [11 13] [12]	[1 2 3 4] [5 6 7 8]
2	[1 2 3 4] [5 6 7 8] [9 13 14] [10 12] [11 15] [16]	[1 2 3 4] [5 6 7 8]
3	[1] [2] [3] [4] [5] [6] [7] [8] [9 10 11 12] [13 14 15 16]	[1] [2] [3] [4] [5] [6] [7] [8] [9 10 11 12] [13 14 15 16]
4	[1] [2] [3] [4] [5] [6] [7] [8] [9 10 11 12] [13 14 15 16]	[1] [2] [3] [4] [5] [6] [7] [8] [9 10 11 12] [13 14 15 16]

Table 4.3: Models before and after pruning when the 16-bit sample problem of the overloaded scaling is solved by the proposed approach.

Chapter 5

Experiments

The experiments are designed for observing the behavior of the proposed approach on sets of problems with different scaling difficulties. Furthermore, various selection pressures are also taken into considerations to make a more thorough observation. In this study, three bounding models of scalings [5] are considered: exponential, power-law, and uniform. While the uniform and exponential cases bound the scaling performance of an algorithm at two extremes, the power-law cases enable us to see the behavior in between. Based on the different scalings, three sets of test functions are constructed using f_{trap_4} as the elemental function:

$$\text{Exponential:} \quad \sum_{i=0}^{m-1} 5^i f_{trap_4}(s_{4i+1}s_{4i+2}\cdots s_{4i+4}) \tag{5.1}$$

$$\text{Power-law:} \quad \sum_{i=0}^{m-1} (i+1)^3 f_{trap_4}(s_{4i+1}s_{4i+2}\cdots s_{4i+4}) \tag{5.2}$$

$$\text{Uniform:} \quad \sum_{i=0}^{m-1} f_{trap_4}(s_{4i+1}s_{4i+2}\cdots s_{4i+4}) \tag{5.3}$$

For simplicity, the stopping criterion is set such that a run is terminated when all solutions in the population converge to the same fitness value.

5.1 Effect of Selection Pressure

It is easily conceivable that the precondition for the proposed approach to work as expected is that the MPM search correctly recognizes at least *some* problem structures. Based on the precondition, the model pruning mechanism can accordingly perform its duty in eliminating the ineffective parts and preserve the verified marginal models. Thus, it is

quite important to set an appropriate selection pressure so that at least the more salient building blocks can emerge properly and as a consequence, be identified by the MPM search.

In order to make the following experiments meaningful, we firstly perform the tests to find proper selection pressures for both the proposed approach and the original ECGA. Because tournament selection is adopted, the selection pressure is altered by changing the tournament size. We consider tournament sizes ranging from 8 to 24, and the problem instances used to make the observations are of length 40 bits ($m = 10$) and 80 bits ($m = 20$). For simplicity, the splitting of population is performed in the way that the two resulting sub-populations are disjoint and of equal size. For each tournament size, the minimum population size required such that on average, $m - 1$ building blocks converge to the correct values in 50 runs for each of the two problem instances is determined by a bisection method.

The results on exponentially, power-law, and uniformly scaled problems are presented in Figures 5.1, 5.2, and 5.3, respectively. It can be observed that for all three scalings, the original ECGA performs best under tournament sizes 12 or 16. On the other hand, although tournament sizes 12 and 16 also work well for the proposed method, the best tournament size varies from one scaling to another.

5.2   Impact on Population Requirements

This section describes the experimental settings and results of the proposed method compared to that of the original ECGA on three sets of problems with different scaling difficulties. The problem size ranges from 40 to 80 bits ($m = 10 \ldots 20$). For each problem instance, the minimum population size required such that on average, $m - 1$ building blocks converge to the correct values in 50 runs is determined by a bisection method. As in the previous experiment, the splitting of population is also performed in the way that the two resulting sub-populations are disjoint and of equal size. Two selection pressures are adopted by setting tournament size t to 12 and 16. The reason for using these two tournament sizes is because our approach is compared with the original ECGA which
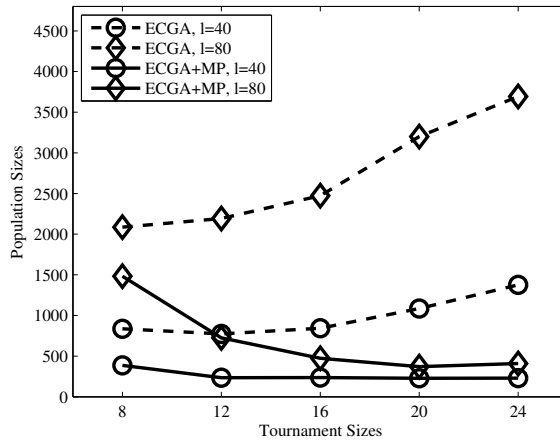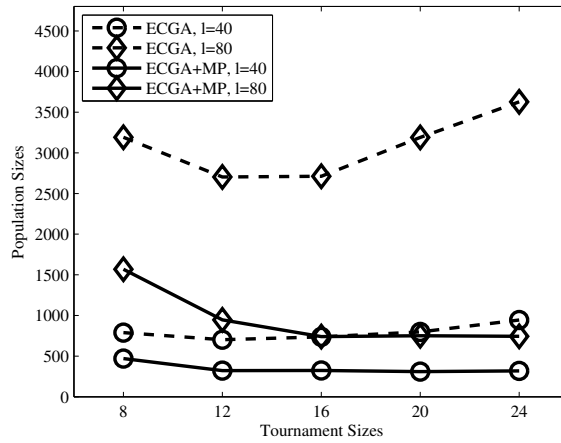
(a) Population Sizes
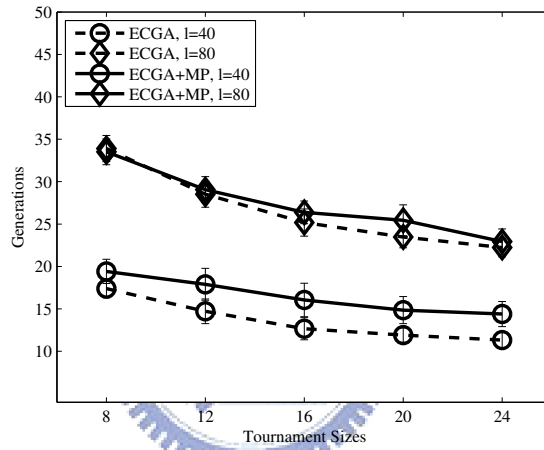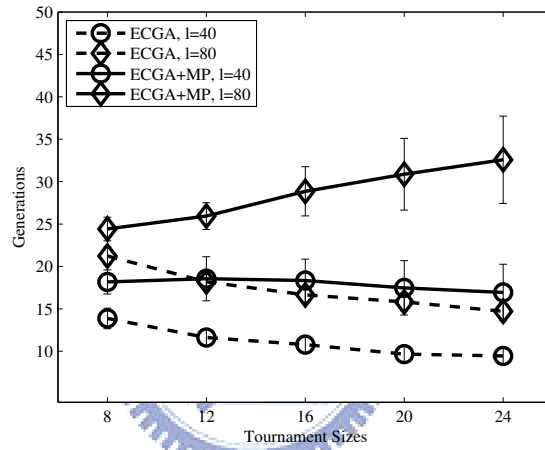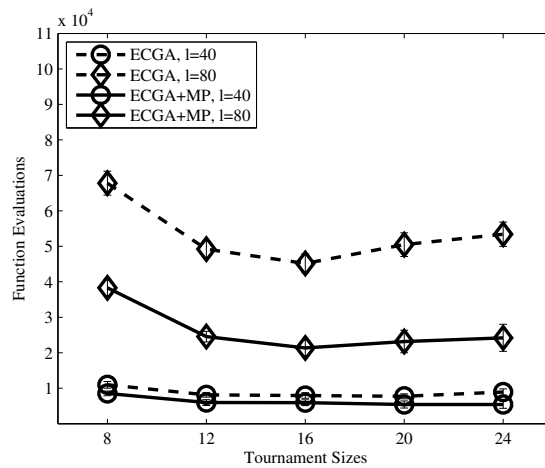

(b) Generations
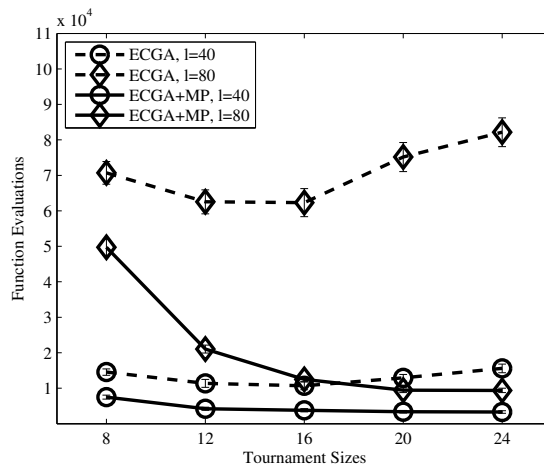
(c) Function Evaluations

Figure 5.1: Empirical results of the proposed method and the original ECGA on 40-bit and 80-bit *exponentially scaled problems*. Five tournament sizes ranging from 8 to 24 are experimented to observe the behaviors of the two algorithms under different selection pressures.
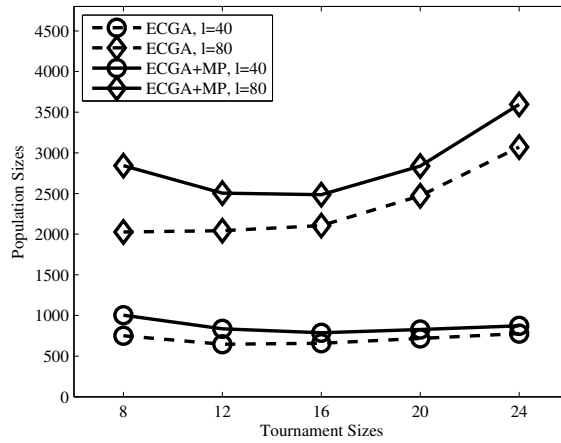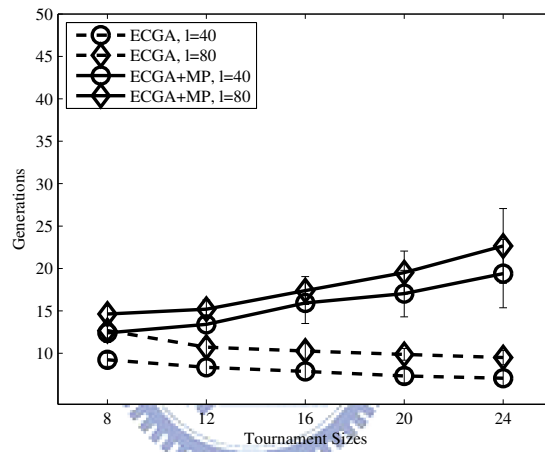
(a) Population Sizes

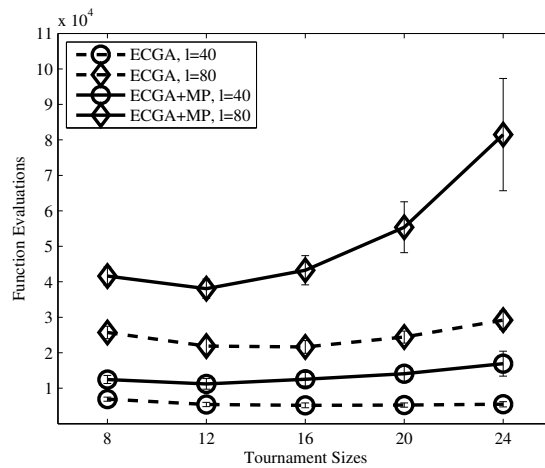(b) Generations

(c) Function Evaluations

Figure 5.2: Empirical results of the proposed method and the original ECGA on 40-bit and 80-bit *power-law scaled problems*. Five tournament sizes ranging from 8 to 24 are experimented to observe the behaviors of the two algorithms under different selection pressures.

(a) Population Sizes

(b) Generations

(c) Function Evaluations

Figure 5.3: Empirical results of the proposed method and the original ECGA on 40-bit and 80-bit *uniformly scaled problems*. Five tournament sizes ranging from 8 to 24 are experimented to observe the behaviors of the two algorithms under different selection pressures.

seems to perform better using $t = 12$ or $t = 16$ according to the previous set of experiments. Otherwise, a question might arise that whether or not the inferior performance of the original ECGA under some scaling difficulties comes from the inappropriate setting of selection pressure.

The empirical results on exponentially scaled problems are shown in Figure 5.4. The minimum population sizes required by the proposed method are much lower than that needed by the original ECGA and grow in a very slow rate. The same situation is also observed in the function evaluations and shows that the proposed method works remarkably well.

Figure 5.5 shows the results on power-law scaled problems. The results on the minimum population sizes are similar to the previous set of experiments. The proposed method still uses fewer function evaluations, but the differences are reduced.

The empirical results on uniformly scaled problems are presented in Figure 5.6. As expected, the proposed method requires larger population sizes than that needed by the original ECGA. Because for uniformly scaled problems, the model building process can correctly identify all the building blocks, the verification on the built model may just be useless. The results confirmed that the function evaluations used by the proposed method are about twice as many as that needed by the original ECGA under the same selection pressure.

5.3   Building vs. Verifying

This section describes the sets of experiments on the proposed method to reveal the change in performance when the different splitting ratios of the two sub-populations are adopted. It presents the experimental results to illustrate the behaviors under different scalings. The splitting ratio $(|T|/|S+T|)$ ranges from 0.1 to 0.8. The 60-bit problems $(m = 15)$ are adopted as test functions. For each splitting ratio, the minimum population size required such that on average, $m - 1$ building blocks converge to the correct values in 50 runs is determined by a bisection method. As in the previous experiments, tournament sizes 12 and 16 are used.
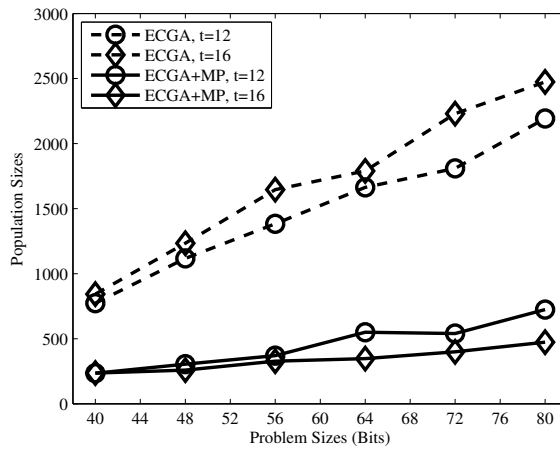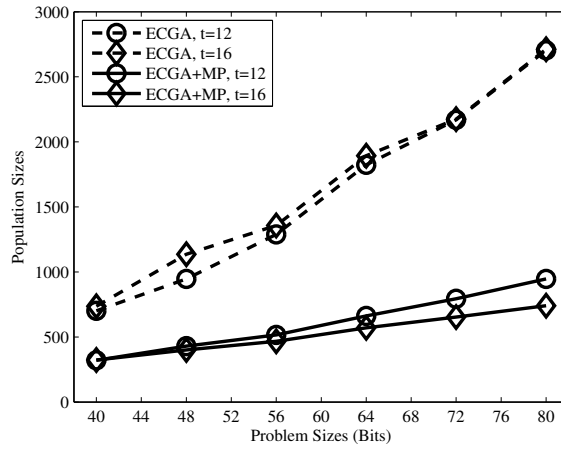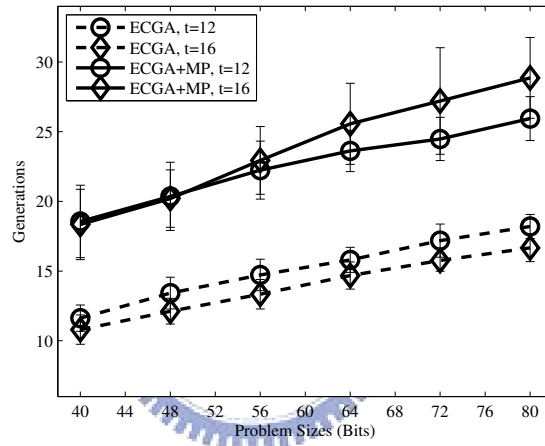
(a) Population Sizes
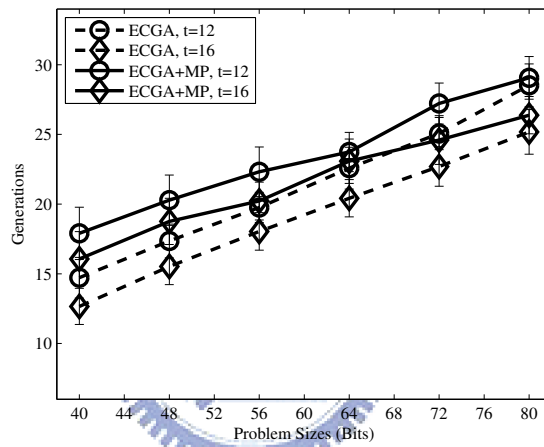
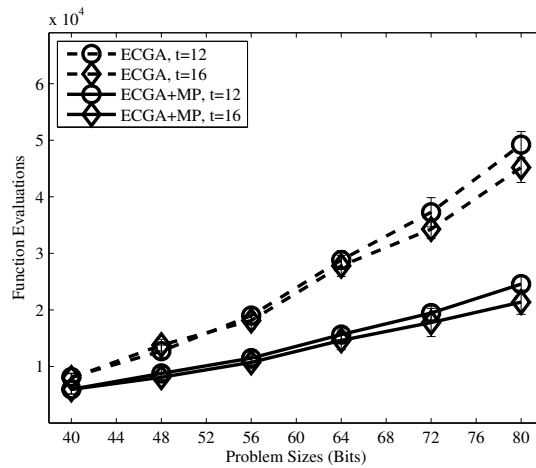(b) Generations
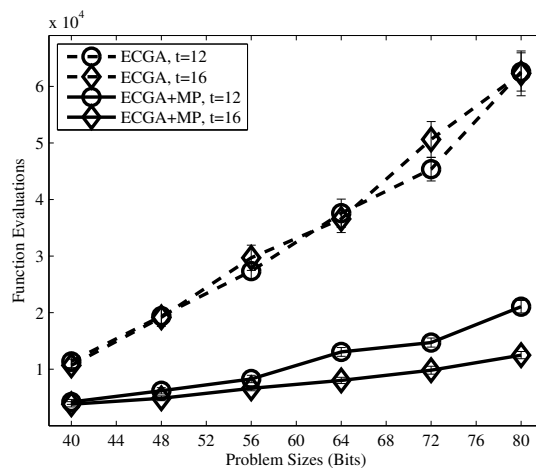
(c) Function Evaluations

Figure 5.4: Empirical results of the proposed method compared to the original ECGA on *exponentially scaled problems* using tournament sizes $t = 12$ and $t = 16$. The problem sizes ranging from 40 to 80 bits are experimented to observe the performance of the algorithms.
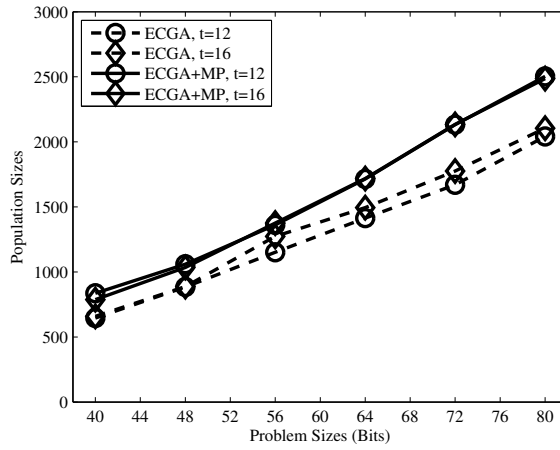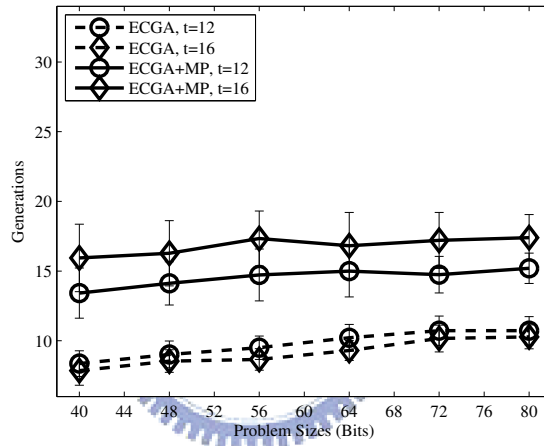
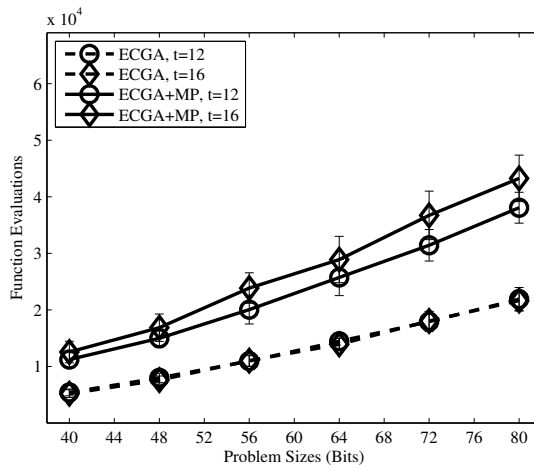(a) Population Sizes

(b) Generations

(c) Function Evaluations

Figure 5.5: Empirical results of the proposed method compared to the original ECGA on *power-law scaled problems* using tournament sizes $t = 12$ and $t = 16$. The problem sizes ranging from 40 to 80 bits are experimented to observe the performance of the algorithms.

(a) Population Sizes

(b) Generations

(c) Function Evaluations

Figure 5.6: Empirical results of the proposed method compared to the original ECGA on *uniformly scaled problems* using tournament sizes $t = 12$ and $t = 16$. The problem sizes ranging from 40 to 80 bits are experimented to observe the performance of the algorithms.
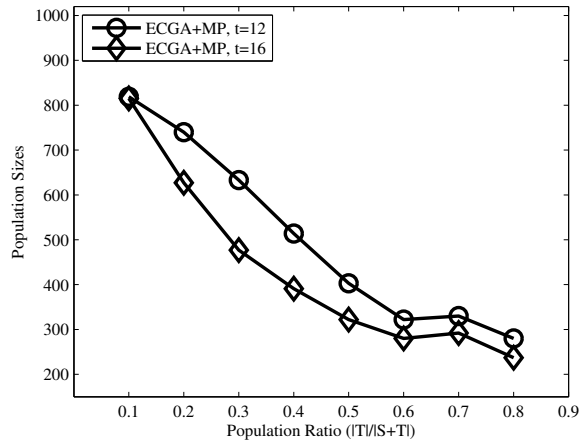
The empirical results on exponentially scaled problems are shown in Figure 5.7. For both tournament sizes, the required population size decreases as the splitting ratio increases. However, the generation increases with the splitting ratio. The combined effect is that the minimum required function evaluation appears when the splitting ratio is 0.6, and the required function evaluations grow when the splitting ratio either increases or decreases.

Figure 5.8 presents the results on power-law scaled problems. Different from the exponentially scaled case, the required population size does not strictly decreases with the increment of the splitting ratio. The population size firstly decreases as the splitting ratio grows and then hits the turning points at 0.5 ($t = 16$) or 0.6 ($t = 12$). Similar to the exponentially scaled case, the generation increases with the splitting ratio. For both tournament sizes, the minimum function evaluation appears when the splitting ratio is 0.3.

Figure 5.9 shows the results on uniformly scaled problems. As expected, Figures 5.9(a), 5.9(b), and 5.9(c) all shares a common pattern that the population size, the generation, and the function evaluation increase with the splitting ratio. It is because in the uniformly scaled case, the linkage is always completely sensible, and there is no need to verify and prune the built probabilistic model.

5.4   Discussion

As aforementioned, we utilize the existence of disparate scalings in problems to create a controlled experimental environment in order to study the situation in which the complete, accurate linkage information may or may not be available to estimation of distribution algorithms. According to the obtained results, the proposed approach does improve the original ECGA on problems where disparate scalings exist among building blocks. Illustrated by Figures 5.4(c) and 5.5(c), significant reductions in function evaluations are achieved. Moreover, for the uniformly scaled problem in which the linkage are always completely sensible, it can be clearly observed that the proposed method uses just nearly twice as many function evaluations as the original ECGA. Such an observation indicates
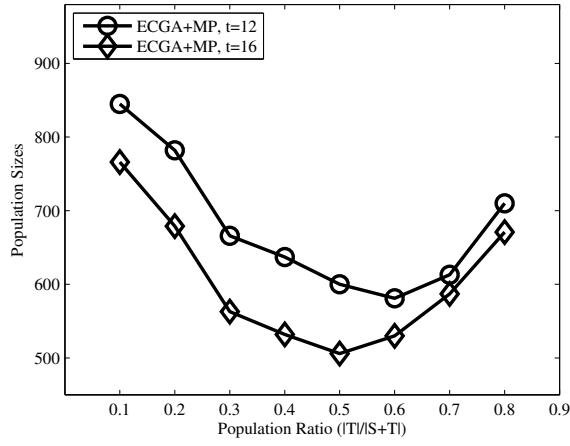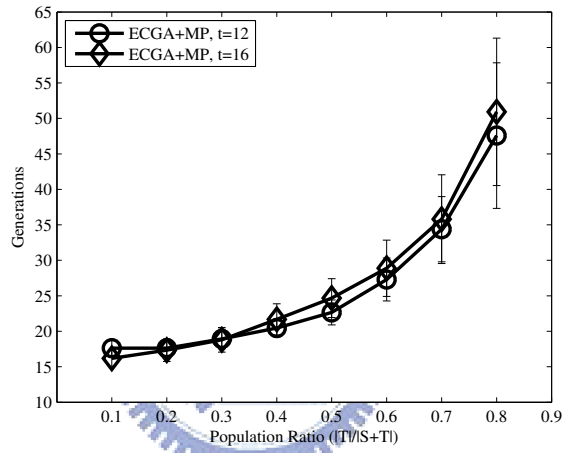
(a) Population Sizes
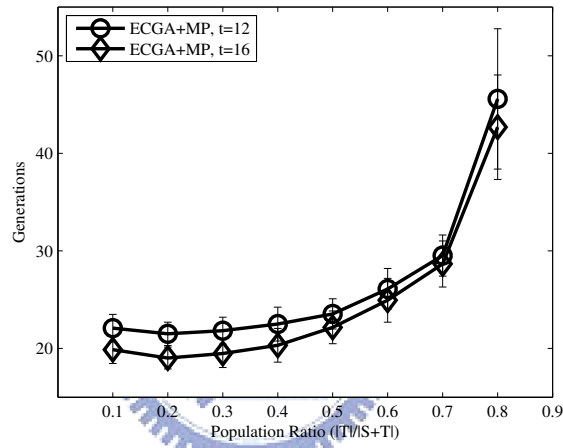
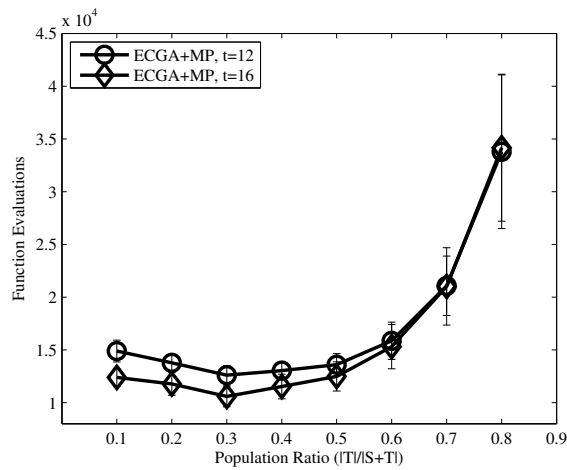(b) Generations
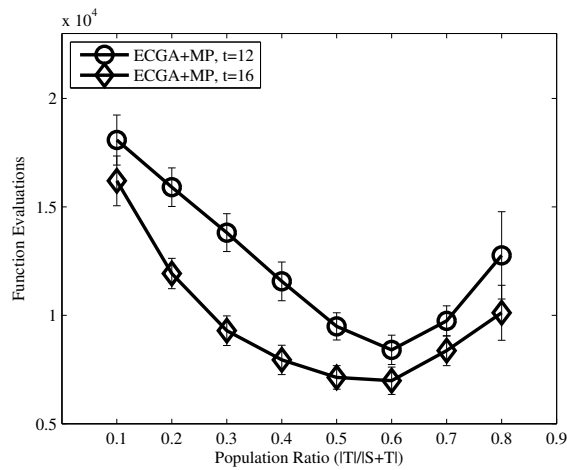
(c) Function Evaluations

Figure 5.7: Empirical results of the proposed method on 60-bit exponentially scaled problems with different splitting ratios between the two sub-populations. The splitting ratio $(|T|/|S + T|)$ ranging from 0.1 to 0.8 are experimented to observe the change in performance of the proposed approach.
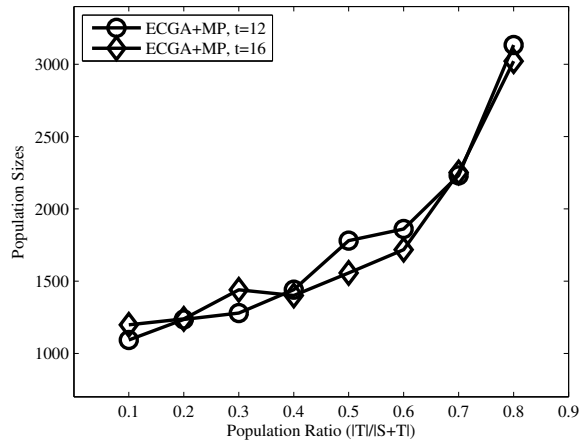
(a) Population Sizes

(b) Generations

(c) Function Evaluations

Figure 5.8: Empirical results of the proposed method on 60-bit power-law scaled problems with different splitting ratios between the two sub-populations. The splitting ratio $(|T|/|S + T|)$ ranging from 0.1 to 0.8 are experimented to observe the change in performance of the proposed approach.
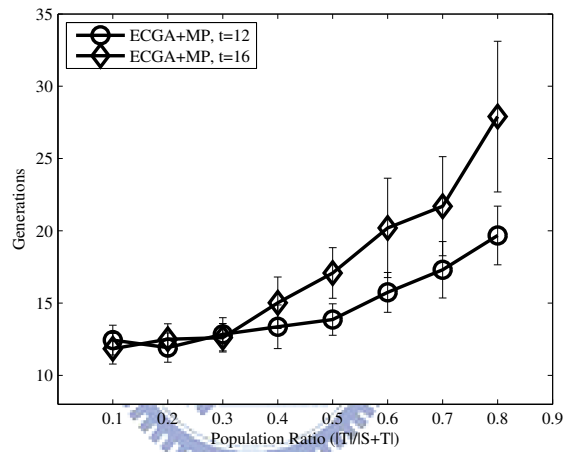
(a) Population Sizes

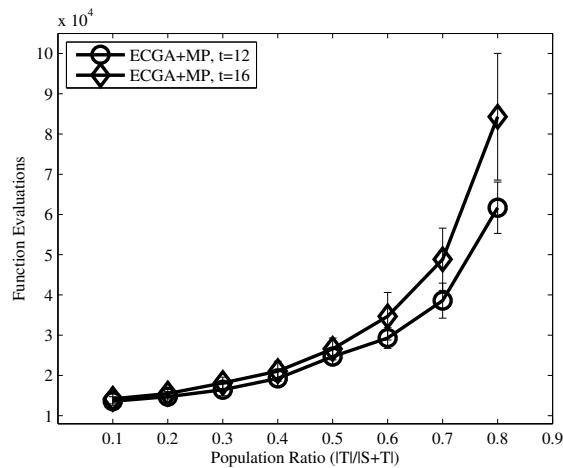(b) Generations

(c) Function Evaluations

Figure 5.9: Empirical results of the proposed method on 60-bit uniformly scaled problems with different splitting ratios between the two sub-populations. The splitting ratio $(|T|/|S + T|)$ ranging from 0.1 to 0.8 are experimented to observe the change in performance of the proposed approach.

that integrating the mechanism of model pruning into ECGA does help ECGA to deliver much better performance on problems comprising building blocks of disparate scalings while the extra costs on the uniformly scaled problem is reasonable.

Observing the empirical results, it shows that our approach works better with a stronger selection pressure if the problem at hand is with distinguishable prominence or solving priority among the constituting subproblems. In these cases, the population can be set to a fairly small size as long as the salient unconverged building blocks can be handled. On the other hand, if we are dealing with problems which are composed of subproblems of roughly equal salience, the selection pressure should be adjusted to a weaker level, and of course, a larger population size will be required since the performance gain delivered by the model pruning mechanism is limited in the situation where the linkage is always almost completely sensible during the entire optimization process.

For setting the splitting ratio, our suggestion is to adopt a ratio under 0.7. If the given problem is evidently with distinguishable prominence among the constituting subproblems, using higher splitting ratios will yield better performance. Lower ratios are more suitable if the problem at hand is composed of subproblems of roughly equal salience. If the property of the problem is completely unknown as in black-box optimization, setting the ratio to 0.5 is a reasonable choice.

Chapter 6

Conclusions

This chapter concludes this thesis. It starts with a summary of the ideas that we have introduced and the status of the research project. Following that, we provide an alternative view of our attempt and the conclusions of this study.

6.1 Summary

This thesis started at reviewing previous studies on EDAs and scaling difficulties, and then illustrated how the scaling difficulty shadows EDAs' ability in recognizing building blocks. Following that, a notion called *linkage sensibility* was introduced to describe the observation, and we use the term *sensible linkage* to refer to the problem structures that can be extracted by inspecting only the set of selected solutions. Based on that concept, we briefly defined the effectiveness of distributions estimated by probabilistic model building. That is, for a distribution to be effective, it must be

- consistent with building blocks, and

- providing good directions for further search.

After that, we proposed a general approach that prunes the built model to achieve a more desirable modeling. Finally, an implementation of the proposed approach on ECGA was introduced as well as experimented on several test functions of different scaling difficulties.

6.2 Perspectives

According to the results of experiments, using pruned model to sample partially does deliver better performance than the original, complete sampling approach if there are

disparate scales among building blocks. Moreover, for the uniformly scaled problems in which the linkage are always completely sensible, the proposed method uses just nearly twice as many function evaluations as the original approach. While this performance is already reasonable considered that only half of the population is used for model building, we would like to point out another direction of thinking concerning this constant overhead: most problems that we are dealing with are not uniformly scaled. That is, we can expect more often the case would be that the subproblems are of unequal salience. Thus, it may be more practical to look at black-box optimization problems with this caution in mind and not assume the linkage would be completely sensible.

In this study, we focused on the scaling difficulties and their influences on the ability of EDAs to appropriately identify building blocks. However, at a higher scope, our attempt was trying to resolve an important issue which was rarely addressed: what if the information contained in the given population is inevitably insufficient? The approach to solve this problem was proposed and successfully implemented for ECGA. It may be adopted and carried over to other EDAs such that more flexible and robust EDAs can be developed.
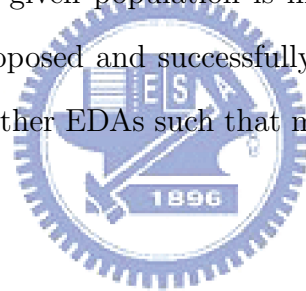
Bibliography

[1] H. Mühlenbein and G. Paaß, "From recombination of genes to the estimation of distributions I. binary parameters," in *PPSN IV: Proceedings of the 4th International Conference on Parallel Problem Solving from Nature.* London, UK: Springer-Verlag, 1996, pp. 178–187.

[2] P. Larrañaga and J. A. Lozano, *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*, ser. Genetic algorithms and evolutionary computation. Boston, MA: Kluwer Academic Publishers, October 2001, vol. 2, ISBN: 0-7923-7466-5.

[3] M. Pelikan, D. E. Goldberg, and F. G. Lobo, "A survey of optimization by building and using probabilistic models," *Computational Optimization and Applications*, vol. 21, no. 1, pp. 5–20, 2002.

[4] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning.* Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989.

[5] ——, *The Design of Innovation: Lessons from and for Competent Genetic Algorithms.* Norwell, MA, USA: Kluwer Academic Publishers, 2002.

[6] G. Harik, "Linkage learning via probabilistic modeling in the ECGA," Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign., IlliGAL Report No. 99010, 1999.

[7] J. H. Holland, *Adaptation in natural and artificial systems.* Cambridge, MA, USA: MIT Press, 1992.

[8] D. E. Goldberg, B. Korb, and K. Deb, "Messy genetic algorithms: Motivation, analysis, and first results," *Complex Systems*, vol. 3, no. 5, pp. 493–530, 1989.

[9] H. Kargupta, "SEARCH, polynomial complexity, and the fast messy genetic algorithm," Ph.D. dissertation, University of Illinois, 1995.

[10] G. Harik, "Learning gene linkage to efficiently solve problems of bounded difficulty using genetic algorithms," Ph.D. dissertation, University of Illinois, 1997.

[11] Y.-p. Chen, *Extending the scalability of linkage learning genetic algorithms: Theory and practice*, ser. Studies in Fuzziness and Soft Computing. Springer, 2006, vol. 190, ISBN: 3-540-28459-1.

[12] H. Kargupta, "The gene expression messy genetic algorithm," in *International Conference on Evolutionary Computation*, 1996, pp. 814–819.

[13] M. Munetomo and D. Goldberg, "Identifying linkage by nonlinearity check," Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign., IlliGAL Report No. 98012, 1998.

[14] M. Munetomo and D. E. Goldberg, "Identifying linkage groups by nonlinearity/non-monotonicity detection," in *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-99*, W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith, Eds., vol. 1. Orlando, Florida, USA: Morgan Kaufmann, 13-17 1999, pp. 433–440.

[15] R. B. Heckendorn and A. H. Wright, "Efficient linkage discovery by limited probing," *Evolutionary Computation*, vol. 12, no. 4, pp. 517–545, 2004.

[16] S. Baluja, "Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning," Pittsburgh, PA, USA, Tech. Rep., 1994.

[17] G. R. Harik, F. G. Lobo, and D. E. Goldberg, "The compact genetic algorithm," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, p. 287, November 1999.

[18] J. de Bonet, C. Isbell, and P. Viola, "MIMIC: Finding optima by estimating probability densities," in *Advances in Neural Information Processing Systems*, M. C. Mozer, M. I. Jordan, and T. Petsche, Eds., vol. 9. The MIT Press, 1997, pp. 424–430.

[19] S. Baluja and S. Davies, "Using optimal dependency-trees for combinational optimization," in *ICML '97: Proceedings of the Fourteenth International Conference on Machine Learning.* San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1997, pp. 30–38.

[20] M. Pelikan and H. Mühlenbein, "The bivariate marginal distribution algorithm," in *Advances in Soft Computing - Engineering Design and Manufacturing*, R. Roy, T. Furuhashi, and P. K. Chawdhry, Eds. London: Springer-Verlag, 1999, pp. 521–535.

[21] M. Pelikan, D. E. Goldberg, and E. Cantú-Paz, "BOA: The Bayesian optimization algorithm," in *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-99*, W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith, Eds., vol. I. Orlando, FL: Morgan Kaufmann Publishers, San Fransisco, CA, 13-17 1999, pp. 525–532.

[22] R. Etxeberria and P. Larrañaga, "Global optimization using bayesian networks," in *Proceedings of the Second Symposium on Artificial Intelligence (CIMAF-99)*, A. O. Rodriguez, M. S. Ortiz, and R. S. Hermida, Eds., Habana, Cuba, 1999, pp. 332–339.

[23] H. Mühlenbein and T. Mahnig, "FDA: A scalable evolutionary algorithm for the optimization of additively decomposed functions," *Evolutionary Computation*, vol. 7, no. 4, pp. 353–376, 1999.

[24] H. Mühlenbein and R. Höns, "The estimation of distributions and the minimum relative entropy principle," *Evolutionary Computation*, vol. 13, no. 1, pp. 1–27, 2005.

[25] D. Thierens, D. E. Goldberg, and Â. G. Pereira, "Domino convergence, drift and the temporal salience structure of problems," in *Proceedings of the 1998 IEEE International Conference on Evolutionary Computation.* IEEE Press, 1998, pp. 535–540.

[26] D. E. Goldberg, K. Deb, and J. H. Clark, "Genetic algorithms, noise, and the sizing of populations," *Complex Systems*, vol. 6, no. 4, pp. 333–362, 1992.

[27] D. E. Goldberg and M. Rudnick, "Genetic algorithms and the variance of fitness," *Complex Systems*, vol. 5, no. 3, pp. 265–278, 1991.

[28] D. E. Goldberg, K. Deb, and B. Korb, "Messy genetic algorithms revisited: Studies in mixed size and scale," *Complex Systems*, vol. 4, no. 4, pp. 415–444, 1990.

[29] F. G. Lobo, D. E. Goldberg, and M. Pelikan, "Time complexity of genetic algorithms on exponentially scaled problems," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000)*, D. Whitley, D. Goldberg, E. Cantú-Paz, L. Spector, I. Parmee, and H.-G. Beyer, Eds. Las Vegas, Nevada, USA: Morgan Kaufmann, 10-12 2000, pp. 151–158.

[30] Y.-p. Chen and D. E. Goldberg, "Convergence time for the linkage learning genetic algorithm," *Evolutionary Computation*, vol. 13, no. 3, pp. 279–302, 2005.

[31] K. Deb and D. E. Goldberg, "Analyzing deception in trap functions," in *Foundations of Genetic Algorithms 2*, 1993, pp. 93–108.

[32] ——, "Sufficient conditions for deceptive and easy binary functions," *Annals of Mathematics and Artificial Intelligence*, vol. 10, no. 4, pp. 385–408, 1994.

[33] J. Rissanen, "Modelling by shortest data description," *Automatica*, vol. 14, pp. 465–471, 1978.

[34] T. M. Cover and J. A. Thomas, *Elements of information theory*. New York, NY, USA: Wiley-Interscience, 1991.

[35] T. M. Mitchell, *Machine Learning*. McGraw-Hill Higher Education, 1997.