Sensibility of Linkage Information and Effectiveness of Estimated Distributions

Chung-Yao Chuang
Ying-ping Chen

NCLab Report No. NCL-TR-2008006
August 2008

Natural Computing Laboratory (NCLab)
Department of Computer Science
National Chiao Tung University
329 Engineering Building C
1001 Ta Hsueh Road
HsinChu City 300, TAIWAN
http://nclab.tw/

Sensibility of Linkage Information and Effectiveness of Estimated Distributions

Chung-Yao Chuang and Ying-ping Chen
Department of Computer Science
National Chiao Tung University
HsinChu City 300, Taiwan
{cychuang, ypchen}@nclab.tw

August 25, 2008

Abstract

Probabilistic model building performed by estimation of distribution algorithms (EDAs) 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 on the extended compact genetic algorithm (ECGA) and experimented on several sets of additively separable problems with different scaling setups. 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.

1 Introduction

Estimation of distribution algorithms (EDAs) [1, 2, 3] 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 automatically capture the likely structure of promising solutions and exploit the identified problem regularities to facilitate further search. It is presumed that EDAs can detect the structure of the problem by recognizing the regularities within the promising solutions. However, for certain problems, EDAs are unable to identify the entire structure of the problem at a given time 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 paper, we start from observing the evolutionary process of an EDA when 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 inconsistent with the actual problem structure. Furthermore, the proposed approach is implemented on the extended compact genetic algorithm [4] and experimented on several sets of additively separable problems with different scaling difficulties [5] to demonstrate the applicability.

In the next section, we will briefly review the research topics concerning this study. After that, section 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 in section 4 to resolve such an issue and enforce the accurate processing during the optimization process. In section 5, an implementation of the proposed approach on the extended compact genetic algorithm will be detailed. Section 6 presents the empirical results, followed by the discussion and observations on the results in section 7. Finally, section 8 concludes this paper.

2 Background

Genetic algorithms (GAs) [6, 7] 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 to implicitly process a large number of partial solutions has been recognized as an important source of the computational power of GAs. According to the Schema theorem [6], short, low-order, and highly fit sub-solutions increase their share to be combined, and also as stated in the building block hypothesis [7], 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.

However, 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 this building block disruption problem, various techniques have been proposed. In this study, we focus on one line of such efforts 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) [8] and the compact genetic algorithm (cGA) [9], 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) [10], Baluja's dependency tree approach [11], and the bivariate marginal distribution algorithm (BMDA) [12], to modeling multivariate interactions, such as the extended compact genetic algorithm (ECGA) [4], the Bayesian optimization algorithm (BOA) [13], the estimation of Bayesian network algorithm (EBNA) [14], the factorized distribution algorithm (FDA) [15], and the learning version of FDA (LFDA) [16]. By detecting dependencies among variables through probabilistic modeling, these approaches can capture the structure of the problem and thus avoid the disruption of identified partial solutions.

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 ob-

Gen.	Marginal Product Model
1	$[\boldsymbol{s_1\ s_2\ s_3\ s_4}]\ [s_5\ s_{10}\ s_{16}]\ [s_6\ s_7]\ [s_8\ s_9\ s_{12}]\ [s_{11}\ s_{14}\ s_{15}]\ [s_{13}]$
2	$[\cancel{s_1}]\ [\cancel{s_2}]\ [\cancel{s_3}]\ [\cancel{s_4}]\ [\boldsymbol{s_5\ s_6\ s_7\ s_8}]\ [s_9\ s_{13}\ s_{16}]\ [s_{10}\ s_{14}\ s_{15}]\ [s_{11}s_{12}]$
3	$[\cancel{s_1}]\ [\cancel{s_2}]\ [\cancel{s_3}]\ [\cancel{s_4}]\ [\cancel{s_5}]\ [\cancel{s_6}]\ [\cancel{s_7}]\ [\cancel{s_8}]\ [\boldsymbol{s_9\ s_{10}\ s_{11}\ s_{12}}]\ [s_{13}\ s_{16}]\ [s_{14}\ s_{15}]$
4	$[\cancel{s_1}]\ [\cancel{s_2}]\ [\cancel{s_3}]\ [\cancel{s_4}]\ [\cancel{s_5}]\ [\cancel{s_6}]\ [\cancel{s_7}]\ [\cancel{s_8}]\ [\cancel{s_9}]\ [\cancel{s_{10}}]\ [\cancel{s_{11}}]\ [\cancel{s_{12}}]\ [\boldsymbol{s_{13}\ s_{14}\ s_{15}\ s_{16}}]$

Table 1: Marginal product models built by ECGA when an exponentially scaled problem is being solved. Each group of variables represents a marginal model in which a marginal distribution resides. The converged variables are crossed out.

served that building blocks with higher marginal fitness contributions—salient building blocks—converge before those with lower marginal fitness contributions. This sequential convergence behavior is referred to as domino convergence [17]. 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 than other parts. Such a situation can pose two types of difficulties. Firstly, because the processing on the population is statistical in nature, the disparate building block scaling can cause inaccurate processing of less salient building blocks [18, 19]. 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 proper, selective pressures. Other previous studies on this topic include the explicit role of scale in a systematic experimental setting [20], a theoretical model on convergence behavior of exponentially scaled problems [17], an extension of that model to building blocks of length more than one variable [21] and a convergence model of linkage learning genetic algorithm (LLGA) [22] on problems of different scaling setups [23].

Although the aforementioned 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 the accurate processing on building blocks and prevent random drifting from taking place. In the remainder of this paper, our approach will be demonstrated and evaluated on the test problems constructed by concatenating several trap functions. A k-bit trap function is a function of unitation[1] which can be expressed as

$$f_{trap_k}(s_1 s_2 \cdots s_k) = \begin{cases} k, & \text{if u = k} \\ k-1-u, & \text{otherwise} \end{cases},$$

where u is the number of ones in the binary string $s_1 s_2 \cdots s_k$. 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 [24, 25].

3 Linkage Sensibility

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 they differ in their representative power, capture the likely structures of good solutions by processing the population-wise statistics collected from

[1] A function of which the function value depends only on the number of ones in the binary input string.

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 by Harik [4], 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.

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 cannot 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 formed by concatenating four 4-bit trap functions as subproblems,

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

where $s_1 s_2 \cdots s_{16}$ is a solution string. Note that different from 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. Suppose that we choose ECGA [4], which uses a class of multivariate probabilistic models called marginal product models (MPMs)[2], to tackle 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 1. In this table, the variables enclosed by the same pair of brackets are considered dependent and modeled jointly. 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 1 are only partially correct in each generation. More specifically, in each generation, only the most salient building block on which the population have not converged is correctly modeled. 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.

From the above example, 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 cannot "see" the entire structure of the problem from the selected set of solutions because disparate scale 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*. Based on this notion, we 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 cannot 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 often the case that the constituting subproblems are weighted significantly 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 section 2 further worsens the situation. These situations and issues are usually handled by

[2]See section 5.1 for a more detailed description on ECGA and marginal product models.

increasing population sizes when EDAs are adopted. However, we may deal with these situations in another way if it is possible to distinguish sensible linkage from insensible linkage.

4 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. Hence, the crucial criteria 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 marginal distributions from the built probabilistic model, we can perform partial sampling using only these marginal 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 $[s_1\ s_2\ s_3\ s_4]$ in the first generation and $[s_5\ s_6\ s_7\ s_8]$ in the second generation, 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 $s_1s_2s_3s_4$ according to the marginal distribution and keep $s_5s_6\cdots s_{16}$ unchanged. In the second generation, we re-sample only s_1 to s_8 according to the marginal distributions and keep $s_9s_{10}\cdots s_{16}$ with the same values (note that $s_1s_2s_3s_4$ are converged). In this way, we do not have to resort to increasing the population size to deal with the problems caused by the disparate building block scaling.

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. It may be wise to adopt a complementary approach – to identify those marginal distributions that are *not* likely to be effective. If there is a way to identify the ineffective distributions, we can bypass them and use only the rest of the probabilistic model and thus, approximate the result of knowing effective distributions. Our idea is that we can split the entire population into two sub-populations, use only one of the sub-populations for building probabilistic model, and utilize the other sub-population to collect some statistics for possible indications of ineffectiveness of certain marginal distributions in the probabilistic model built on the first sub-population. That is, with some appropriate heuristics or criterion, we can prune the likely ineffective portions of the model.

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

5 ECGA with Model Pruning

This section starts with a brief review of the extended compact genetic algorithm (ECGA) [4]. Based on the idea of detecting the inconsistency of statistics gathered from two sub-populations of the same source, 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.

5.1 Extended Compact Genetic Algorithm

ECGA 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

$[s_1]$	$[s_2\ s_4]$	$[s_3]$
$P(s_1 = 0) = 0.4$	$P(s_2 = 0, s_4 = 0) = 0.2$	$P(s_3 = 0) = 0.5$
$P(s_1 = 1) = 0.6$	$P(s_2 = 0, s_4 = 1) = 0.1$	$P(s_3 = 1) = 0.5$
	$P(s_2 = 1, s_4 = 0) = 0.1$	
	$P(s_2 = 1, s_4 = 1) = 0.6$	

Table 2: An example of marginal product model that defines a probability distribution over four variables. The variables enclosed in the same brackets are modeled jointly, and each variable subset is considered independent of other variable subsets.

adopted that variable subsets are enclosed in brackets. Table 2 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,

$$\begin{aligned} P(s_1s_2s_3s_4 = 0101) &= P(s_1 = 0) \times P(s_2 = 1, s_4 = 1) \times P(s_3 = 0) \\ &= 0.4 \times 0.6 \times 0.5 \ . \end{aligned}$$

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

In ECGA, both the structure and the parameters of the model are searched and optimized in a greedy fashion 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 [26], 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 descriptive yet not over-complicated 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 the selection operator) with an ideal compression scheme is 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 [27], 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 the 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.

5.2 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 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. 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, $i = 1 \ldots m$, in the built model M, for the ith marginal model, $i = 1 \ldots m$, 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 an expected length more than k_i bits. Based on the earlier reasoning, such a condition indicates that the marginal model is likely ineffective because T' does not agree on this part of the model. Otherwise, it should be able to compress the partial solutions in T'.

Further explained from a machine learning perspective [28], a good model should generalize well to unseen instances. Otherwise, it captures coincidental regularities among the training

Algorithm 1 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 two sub-populations S and T at random.
    S' ← apply t-wise tournament selection on S.
    T' ← apply t-wise tournament selection on T.
    M ← build the MPM on S' with greedy search.
    M' ← prune M based on the inconsistency with T'.
    for each remaining marginal distribution D in M' do
        for each solution s = s_1 s_2 ··· s_ℓ in P do
            Change the values in s partially by sampling D.
        end for
    end for
end while
```

data. If 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 subject 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 subject to proper selection pressures appear to be random, and such a situation brings about the phenomenon of random drifting mentioned in section 2. By its nature, 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 $p_{ij} = 0$. In the present work, 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 (i.e., $\sum_j p_{ij} = 1$).

5.3 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 scale exists among different parts of the problem.

The procedure is presented in Algorithm 1. 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 operator and 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 described previously. 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 satisfied.

A prominent difference between the above process and the regular EDAs is that the sampling might not include all variables. As introduced in section 4, 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

Gen.	Marginal Product Model (Before & After Pruning)	
1	Before	$[s_1\ s_2\ s_3\ s_4]\ [s_5\ s_{13}\ s_{16}]\ [s_6\ s_7\ s_{12}]\ [s_8\ s_{11}]\ [s_9\ s_{10}]\ [s_{14}\ s_{15}]$
	After	$[s_1\ s_2\ s_3\ s_4]$
2	Before	$\cancel{[s_1]}\ \cancel{[s_2]}\ \cancel{[s_3]}\ \cancel{[s_4]}\ [s_5\ s_6\ s_7\ s_8]\ [s_9\ s_{14}]\ [s_{10}\ s_{15}]\ [s_{11}\ s_{13}\ s_{16}]\ [s_{12}]$
	After	$\cancel{[s_1]}\ \cancel{[s_2]}\ \cancel{[s_3]}\ \cancel{[s_4]}\ [s_5\ s_6\ s_7\ s_8]$
3	Before	$\cancel{[s_1]}\ \cancel{[s_2]}\ \cancel{[s_3]}\ \cancel{[s_4]}\ \cancel{[s_5]}\ \cancel{[s_6]}\ \cancel{[s_7]}\ \cancel{[s_8]}\ [s_9\ s_{10}\ s_{11}\ s_{12}]\ [s_{13}\ s_{14}]\ [s_{15}\ s_{16}]$
	After	$\cancel{[s_1]}\ \cancel{[s_2]}\ \cancel{[s_3]}\ \cancel{[s_4]}\ \cancel{[s_5]}\ \cancel{[s_6]}\ \cancel{[s_7]}\ \cancel{[s_8]}\ [s_9\ s_{10}\ s_{11}\ s_{12}]$
4	Before	$\cancel{[s_1]}\ \cancel{[s_2]}\ \cancel{[s_3]}\ \cancel{[s_4]}\ \cancel{[s_5]}\ \cancel{[s_6]}\ \cancel{[s_7]}\ \cancel{[s_8]}\ \cancel{[s_9]}\ \cancel{[s_{10}]}\ \cancel{[s_{11}]}\ \cancel{[s_{12}]}\ [s_{13}\ s_{14}\ s_{15}\ s_{16}]$
	After	$\cancel{[s_1]}\ \cancel{[s_2]}\ \cancel{[s_3]}\ \cancel{[s_4]}\ \cancel{[s_5]}\ \cancel{[s_6]}\ \cancel{[s_7]}\ \cancel{[s_8]}\ \cancel{[s_9]}\ \cancel{[s_{10}]}\ \cancel{[s_{11}]}\ \cancel{[s_{12}]}\ [s_{13}\ s_{14}\ s_{15}\ s_{16}]$

Table 3: Marginal product models before and after pruning when the 16-bit exponentially scaled problem is solved with the proposed approach.

until the sufficient linkage information is available. In this way, better performance in terms of function evaluations can be achieved if disparate scale exists among different parts of the problem.

To see that the proposed method meets its design purpose, Table 3 lists the models before and after pruning when the earlier exponentially scaled problem is solved by Algorithm 1. It can be seen that the proposed approach appropriately removes the ineffective parts during each stage of the optimization process. To further illustrate the behavior and effect of the proposed approach, the algorithm is applied to another problem with a different scaling called *overloaded scaling*[3],

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

where $s_1 s_2 \cdots s_{16}$ is a solution string. The overloaded cases are those with two scales, 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. 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.

6 Experiments

The experiments are designed for observing the behavior of the proposed approach on sets of problems with different scaling difficulties. Because ECGA is limited in handling overlapped building blocks, we use only test problems that are additively separable. 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,

[3] As mentioned by Goldberg [5], the word "overloaded" is a reference to the application of this idea in the early messy GA work [20], 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.

Gen.		Marginal Product Model (Before & After Pruning)
1	Before	$[s_1\ s_2\ s_3\ s_4]\ [s_5\ s_6\ s_7\ s_8]\ [s_9\ s_{16}]\ [s_{10}\ s_{14}\ s_{15}]\ [s_{11}\ s_{13}]\ [s_{12}]$
	After	$[s_1\ s_2\ s_3\ s_4]\ [s_5\ s_6\ s_7\ s_8]$
2	Before	$[s_1\ s_2\ s_3\ s_4]\ [s_5\ s_6\ s_7\ s_8]\ [s_9\ s_{13}\ s_{14}]\ [s_{10}\ s_{12}]\ [s_{11}\ s_{15}]\ [s_{16}]$
	After	$[s_1\ s_2\ s_3\ s_4]\ [s_5\ s_6\ s_7\ s_8]$
3	Before	$\cancel{[s_1]}\ \cancel{[s_2]}\ \cancel{[s_3]}\ \cancel{[s_4]}\ \cancel{[s_5]}\ \cancel{[s_6]}\ \cancel{[s_7]}\ \cancel{[s_8]}\ [s_9\ s_{10}\ s_{11}\ s_{12}]\ [s_{13}\ s_{14}\ s_{15}\ s_{16}]$
	After	$\cancel{[s_1]}\ \cancel{[s_2]}\ \cancel{[s_3]}\ \cancel{[s_4]}\ \cancel{[s_5]}\ \cancel{[s_6]}\ \cancel{[s_7]}\ \cancel{[s_8]}\ [s_9\ s_{10}\ s_{11}\ s_{12}]\ [s_{13}\ s_{14}\ s_{15}\ s_{16}]$
4	Before	$\cancel{[s_1]}\ \cancel{[s_2]}\ \cancel{[s_3]}\ \cancel{[s_4]}\ \cancel{[s_5]}\ \cancel{[s_6]}\ \cancel{[s_7]}\ \cancel{[s_8]}\ [s_9\ s_{10}\ s_{11}\ s_{12}]\ [s_{13}\ s_{14}\ s_{15}\ s_{16}]$
	After	$\cancel{[s_1]}\ \cancel{[s_2]}\ \cancel{[s_3]}\ \cancel{[s_4]}\ \cancel{[s_5]}\ \cancel{[s_6]}\ \cancel{[s_7]}\ \cancel{[s_8]}\ [s_9\ s_{10}\ s_{11}\ s_{12}]\ [s_{13}\ s_{14}\ s_{15}\ s_{16}]$

Table 4: Marginal product models before and after pruning when the 16-bit problem of the overloaded scaling is solved by the proposed approach.

three sets of test functions are constructed using f_{trap_4} as the elemental function:

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

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

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

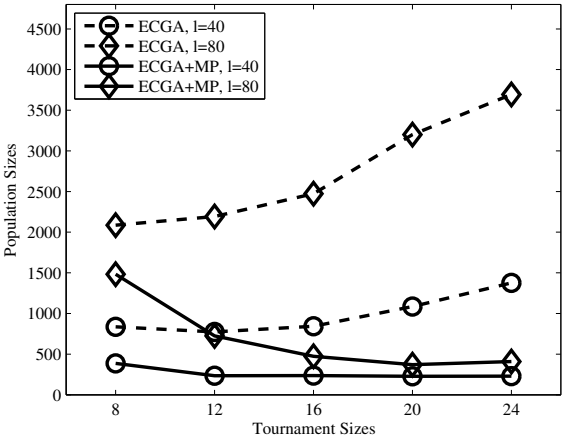
Furthermore, various selection pressures are also taken into considerations to make a more thorough observation. For simplicity, the splitting of population is disjoint and the stopping criterion is set such that a run is terminated when all solutions in the population converge to the same fitness value.

6.1 Effect of Selection Pressure

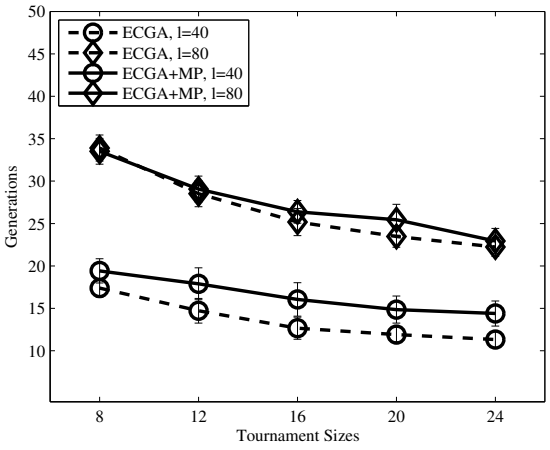
This section describes the experiments that are designed for observing the effect of selection pressure on both the original ECGA and the ECGA combined with the proposed approach. The purpose to perform these experiments are twofold. Firstly, because the proposed approach will be compared with the original ECGA in the next sets of experiments, in order to make a fair and meaningful comparison, it must be ensured that the selection pressure is set properly for the original ECGA. Secondly, the appropriate selection pressure is relatively important to the well-functioning of the proposed approach because the pruning mechanism is designed according to the statistical inconsistencies between the two sub-populations.

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.

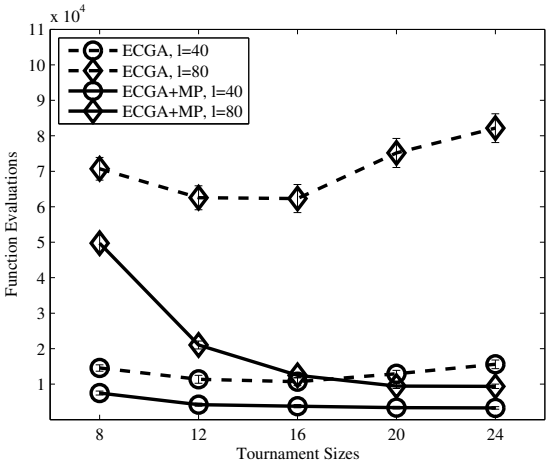
The results on exponentially, power-law, and uniformly scaled problems are presented in Figures 1, 2, and 3, respectively. It can be observed that for all three scalings, the original ECGA performs better under tournament size 12 or 16. On the other hand, although tournament sizes 12 and 16 also work well for the proposed method, the best tournament size varies from one scaling to another.



(a) Population Sizes

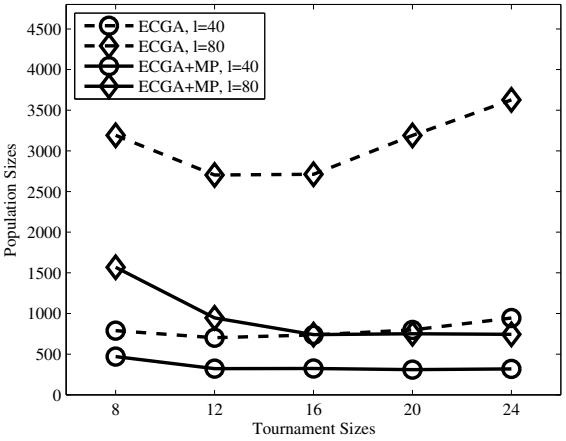


(b) Generations

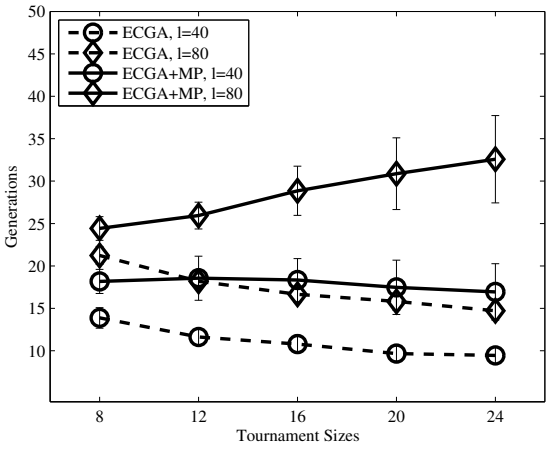


(c) Function Evaluations

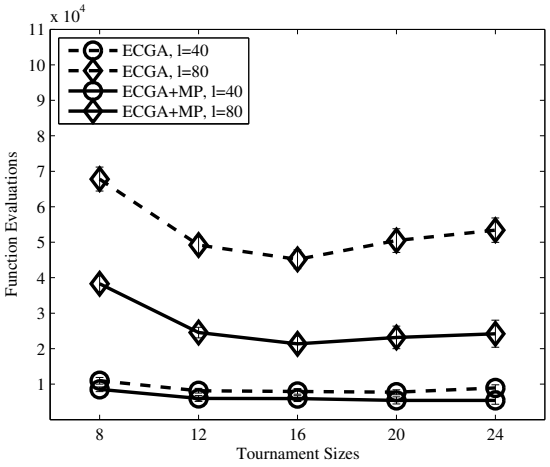
Figure 1: Empirical results of the proposed method and original ECGA on 40 and 80-bit *exponentially scaled problems*. Five tournament sizes ranging from 8 to 24 are experimented to observe the behavior of the algorithms under different selection pressures.



(a) Population Sizes

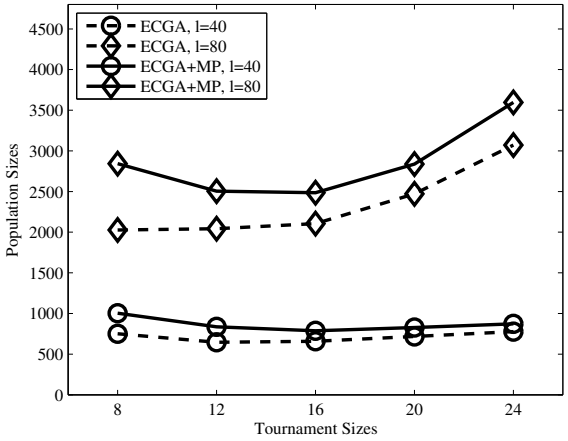


(b) Generations

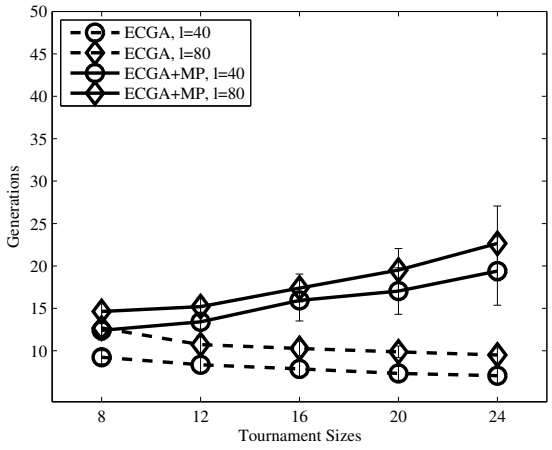


(c) Function Evaluations

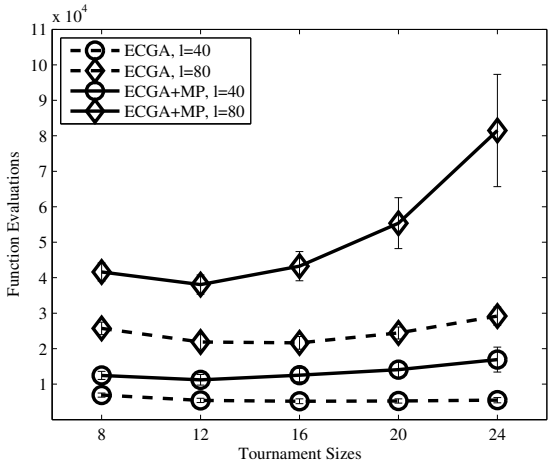
Figure 2: Empirical results of the proposed method and original ECGA on 40 and 80-bit *power-law scaled problems*. Five tournament sizes ranging from 8 to 24 are experimented to observe the behavior of the algorithms under different selection pressures.



(a) Population Sizes



(b) Generations



(c) Function Evaluations

Figure 3: Empirical results of the proposed method and original ECGA on 40 and 80-bit *uniformly scaled problems*. Five tournament sizes ranging from 8 to 24 are experimented to observe the behavior of the algorithms under different selection pressures.

6.2 Impact on Population Requirement

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 setups. 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 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 4. The minimum population sizes required by the proposed method are much lower than that needed by the original ECGA and grow in a relatively slow rate. The same situation is also observed in the function evaluations that our approach works remarkably well.

Figure 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 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 building blocks, the verification on the built model may just be useless. The results also show that the function evaluations used by the proposed method are about twice the number of that needed by the original ECGA.

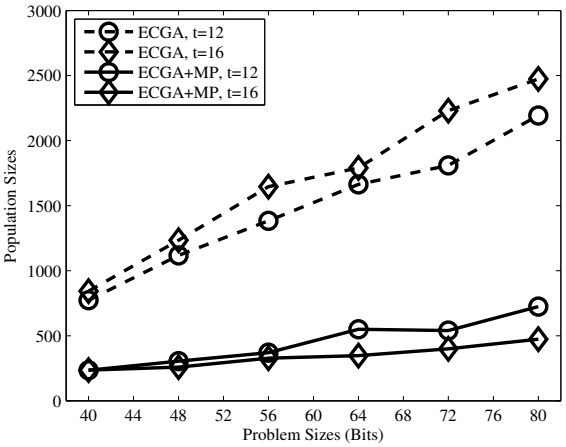
6.3 Building vs. Verifying

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 behavior 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.

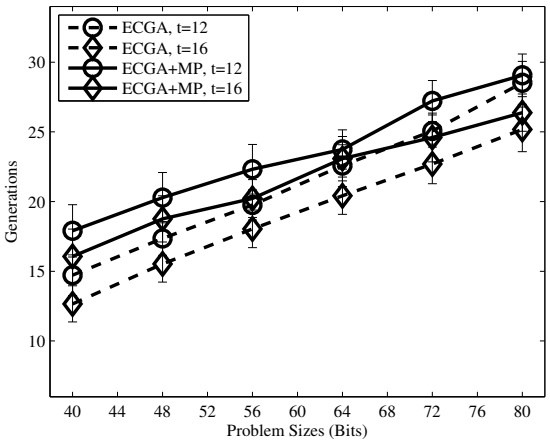
The empirical results on exponentially scaled problems are shown in Figure 7. For both tournament sizes, the required population size decreases as the splitting ratio increases. However, the generation increases with the splitting ratio. The combined effect is that the minimum required function evaluation appears when the splitting ratio is 0.6, and the required function evaluations grow when the splitting ratio either increases or decreases.

Figure 8 presents the results on power-law scaled problems. Different from the previous 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.

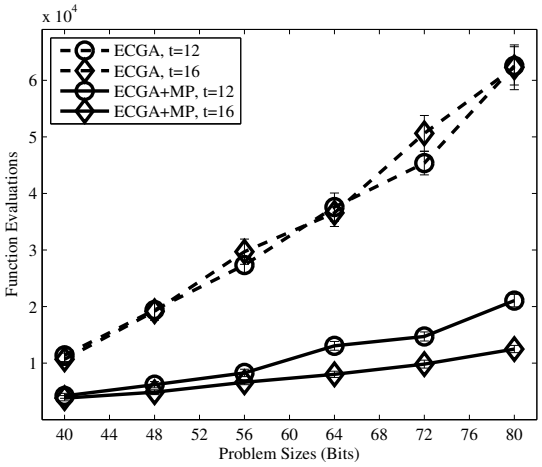
Figure 9 shows the results on uniformly scaled problems. As expected, Figures 9(a), 9(b), and 9(c) all shares a common pattern that the population size, the generation, and the function evaluation increase with the splitting ratio. It is because in the uniformly scaled case, the linkage



(a) Population Sizes

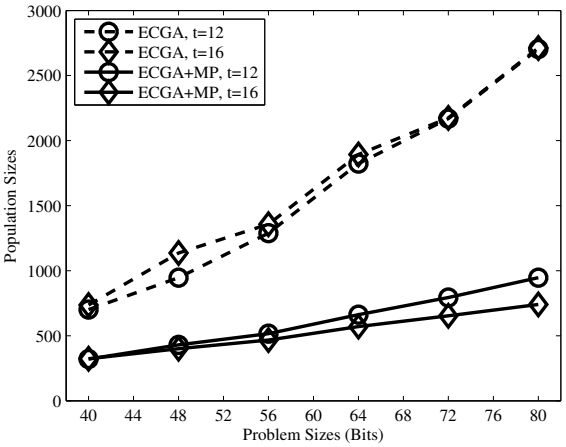


(b) Generations

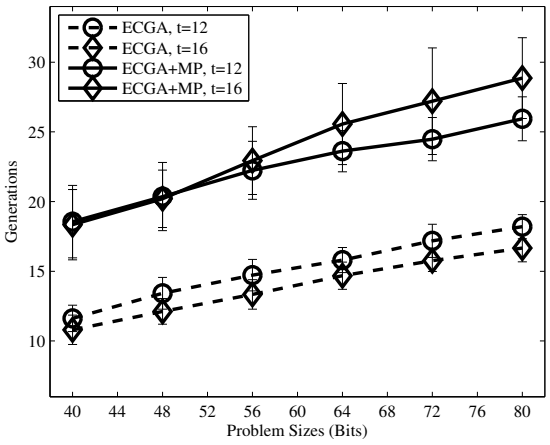


(c) Function Evaluations

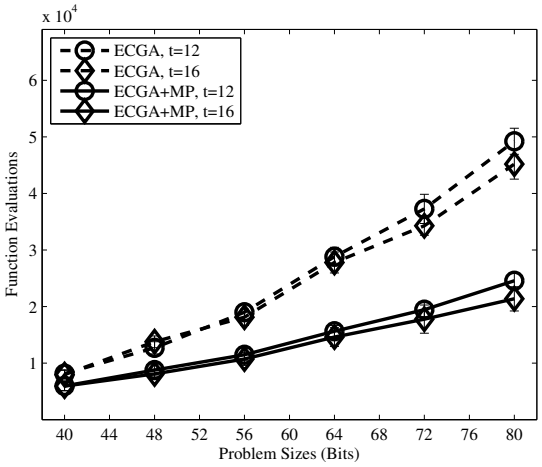
Figure 4: Empirical results of the proposed method compared to the original ECGA on *exponentially scaled problems* using tournament sizes $t = 12$ and $t = 16$. The problem sizes ranging from 40 to 80 bits are experimented to observe the performance of the algorithms.



(a) Population Sizes

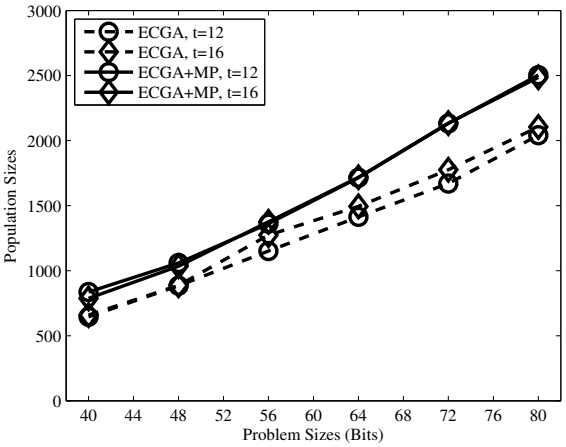


(b) Generations

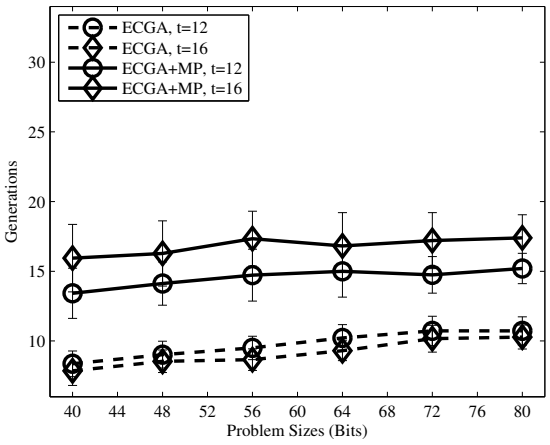


(c) Function Evaluations

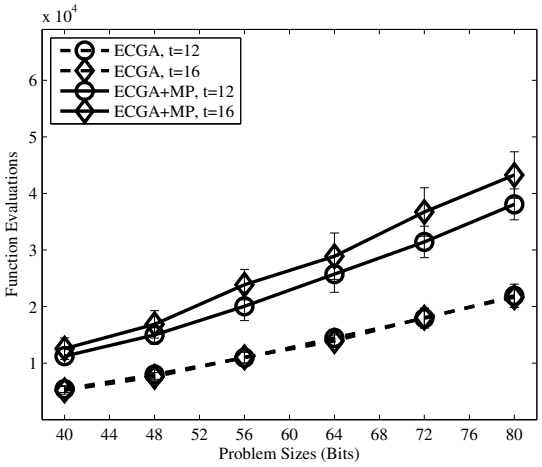
Figure 5: Empirical results of the proposed method compared to the original ECGA on *power-law scaled problems* using tournament sizes $t = 12$ and $t = 16$. The problem sizes ranging from 40 to 80 bits are experimented to observe the performance of the algorithms.



(a) Population Sizes



(b) Generations



(c) Function Evaluations

Figure 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.

is always completely sensible, and there is no need to verify and prune the built probabilistic model.

7 Discussion

As aforementioned, we utilized the existence of disparate scales 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 shown in Figures 4(c) and 5(c), the proposed approach does improve the original ECGA on the test problems where disparate scales exist among building blocks. In this section, we discuss some properties of the proposed approach and possible extension of this work.

7.1 Parameter Settings in Action

The empirical results suggested that, different from the original ECGA, 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 compared to the original ECGA and we speculate that our approach only needs a population that is large enough for the salient unconverged building blocks to be handled properly. 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.

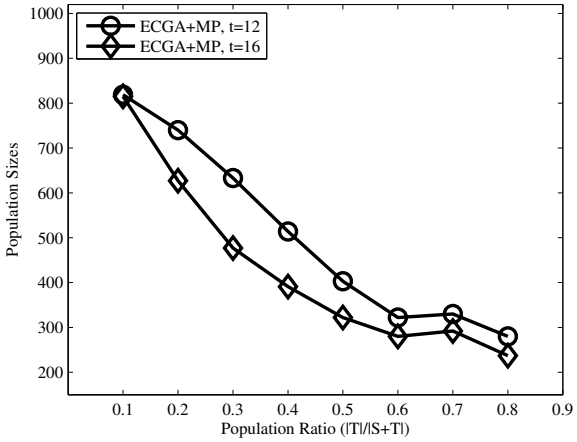
For setting the splitting ratio, our suggestion is to adopt a ratio under 0.7. If the given problem is evidently with distinguishable prominence among the constituting subproblems, using higher splitting ratios will yield better performance. Lower ratios are more suitable if the problem at hand is composed 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.

7.2 Overhead in Uniformly Scaled Problems

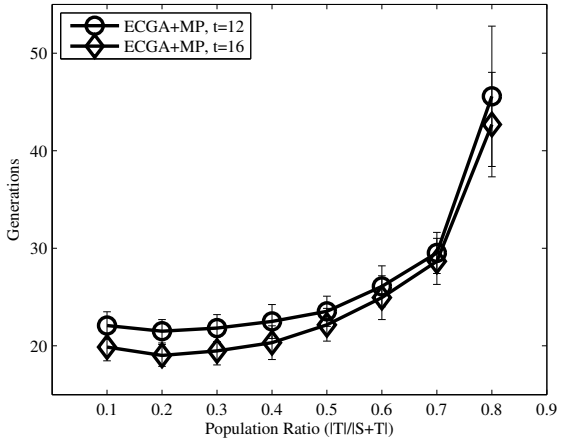
Empirical results presented in Figure 6(c) also show that on the uniformly scaled cases, the proposed approach uses nearly twice as many function evaluations as the original ECGA. We speculate that this double expenditure is a general property of the proposed approach when dealing with uniformly scaled problems.

This speculation can be explained by a reverse thinking on a hypothetical situation described as follows. Suppose that given a uniformly scaled problem, the original ECGA with an appropriate selection pressure needs a population of size n to handle that problem properly. Now, consider the proposed approach for the same problem. If we use a population of size $2n$, then in our algorithm, the entire population will be divided into two sub-populations each of size n, assuming that the splitting of population is disjoint and of equal size. If the original ECGA is capable of detecting the accurate problem structure using a population of size n, then in our algorithm, a sub-population of size n will also do the job. In the ideal case, there will be no statistical inconsistency between the built model and the set of promising solutions selected from the second sub-population. As a result, we wasted half of the population in the pruning mechanism which contributed to the extra cost compared to the original ECGA.

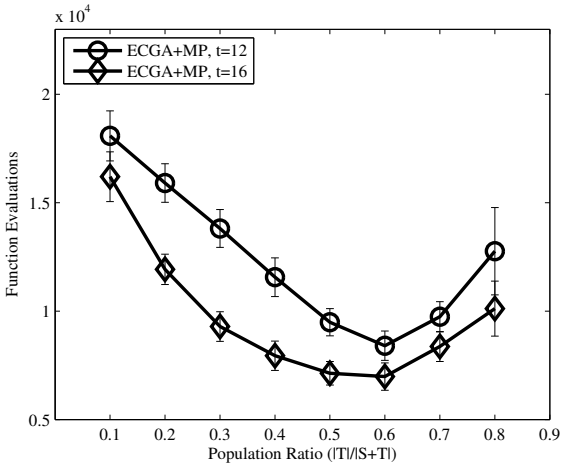
However, the above inference cannot fully explain the obtained results. As illustrated in Figure 6(a) the minimum population sizes needed by the proposed method is not exactly twice



(a) Population Sizes

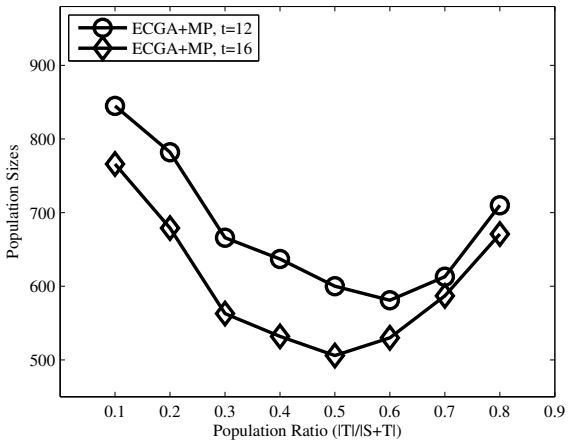


(b) Generations

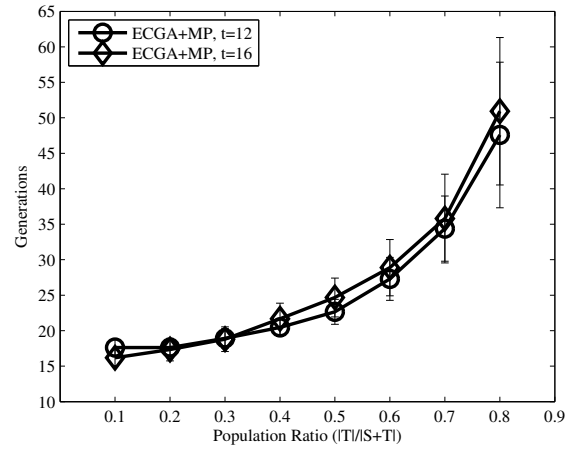


(c) Function Evaluations

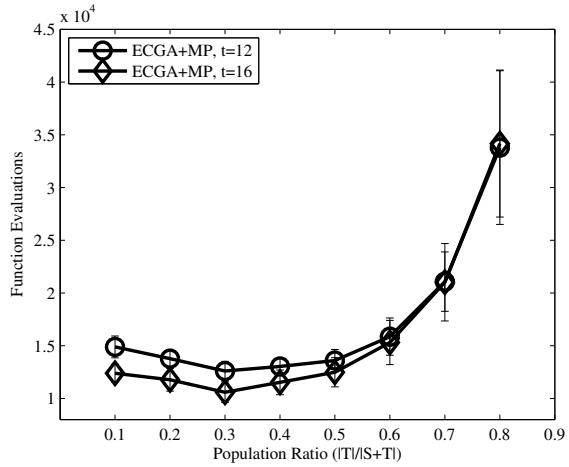
Figure 7: Empirical results of the proposed method on 60-bit *exponentially scaled problems* with different splitting ratios between the two sub-populations. The splitting ratio ($|T|/|S+T|$) ranging from 0.1 to 0.8 are experimented to observe the change in performance of the proposed approach.



(a) Population Sizes

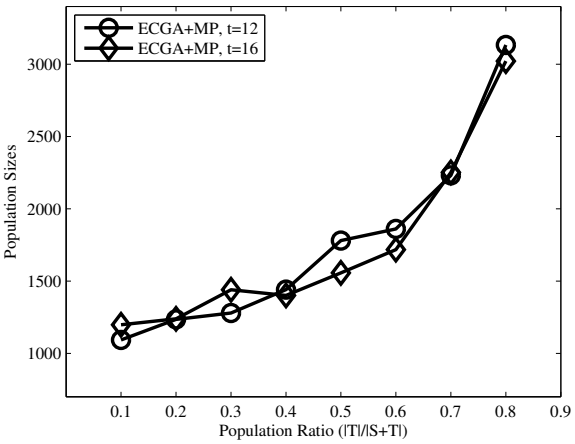


(b) Generations

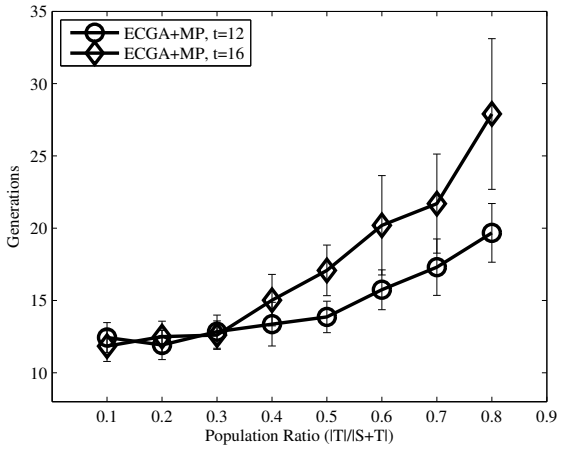


(c) Function Evaluations

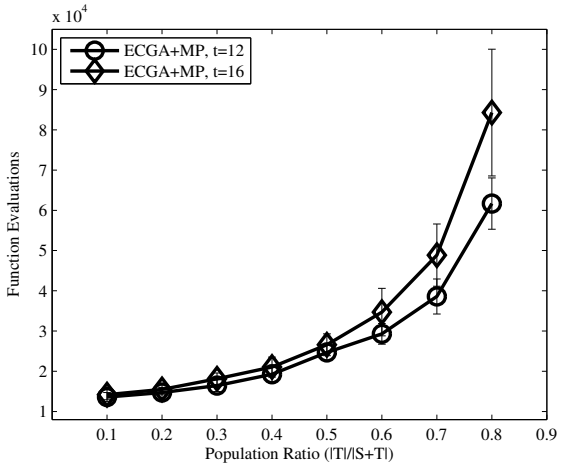
Figure 8: Empirical results of the proposed method on 60-bit *power-law scaled problems* with different splitting ratios between the two sub-populations. The splitting ratio ($|T|/|S+T|$) ranging from 0.1 to 0.8 are experimented to observe the change in performance of the proposed approach.



(a) Population Sizes



(b) Generations



(c) Function Evaluations

Figure 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.

of that required by the original ECGA. In fact, the numbers are much lower than twice of that needed by the original ECGA. On the other hand, our approach uses more generations compared to the original ECGA because the sub-population for model building was not sufficiently large for all problem structures to be detected properly in the beginning of the process. In this situation, the processing was slowed down because the pruning mechanism removed parts of the model that exhibit statistical inconsistencies. As a consequence, the originally expected simultaneous processing of building blocks was not fully achieved which resulted in the delay of convergence. Nevertheless, spending more generations seems to yield an equivalent usage of function evaluations as the hypothetical case described above, and we think that the pruning mechanism introduced an additional interaction between population sizes and generations. Further empirical or theoretical studies are needed to investigate the interaction between these factors.

7.3 Pruning Network-based Probabilistic Models

In this work, we have introduced a technique to prune a given marginal product model based on the statistics collected from a reserved set of solutions. It is possible to extend this idea to design pruning mechanisms for other EDAs. For example, consider EDAs that use network-based probabilistic models with Bayesian information criterion (BIC) [29] as model scoring metrics like EBNA [14] and a variant of BOA [30]. In the binary case, BIC assigns a given network structure B of ℓ variables a score

$$\begin{aligned} S(B) &= \sum_{i=1}^{\ell} \left(-n \times H(X_i|\Pi_i) - 2^{|\Pi_i|} \frac{log_2 n}{2} \right) \\ &= -\sum_{i=1}^{\ell} n \times H(X_i|\Pi_i) - \sum_{i=1}^{\ell} 2^{|\Pi_i|} \frac{log_2 n}{2} \ , \end{aligned}$$

where X_i's, $i = 1 \dots \ell$, are variables and $H(X_i|\Pi_i)$ is the conditional entropy of X_i given its parent Π_i in the network; n is the population size. The conditional entropy $H(X_i|\Pi_i)$ is given by

$$H(X_i|\Pi_i) = -\sum_{x_i,\pi_i} p(x_i,\pi_i) log_2 p(x_i|\pi_i) \ ,$$

where $p(x_i,\pi_i)$ is the probability of instances with $X_i = x_i$ and $\Pi_i = \pi_i$; $p(x_i|\pi_i)$ is the conditional probability of instances with $X_i = x_i$ given that $\Pi_i = \pi_i$.

The term $\sum_{i=1}^{\ell} n \times H(X_i|\Pi_i)$ is of the same function as the compressed population complexity (C_p) in ECGA because $H(X_i|\Pi_i)$ denotes the average number of bits required to store a value of X_i with compression given the information of Π_i. Thus, we can check if a variable X_i should be pruned away or not by the following inequality

$$-\sum_{x_i,\pi_i} q(x_i,\pi_i) log_2 p(x_i|\pi_i) > 1 \ ,$$

where $q(x_i,\pi_i)$ is the frequency of $X_i = x_i$ and $\Pi_i = \pi_i$ observed in the set of solutions selected from the reserved sub-population. Using the idea described in section 5.2, if this inequality holds, then X_i should be removed because it encodes a one-bit partial solution to a bit string of an expected length more than one bit.

However, despite the similarities in ideas, some technical complications remain to be overcome before we can finish the design of a pruning mechanism for network-based probabilistic models. For instance, what if a variable which we intend to prune is a parent node of some other variables? In summary, pruning network-based probabilistic models is potentially feasible and requires further investigations.

8 Summary and Conclusions

This paper started at reviewing previous studies on EDAs and scaling difficulties. It 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 the concept, we briefly defined the effectiveness of distributions estimated by probabilistic model building and proposed a general approach to achieve a more effective modeling. Finally, an implementation of the proposed approach on ECGA was introduced as well as experimented on several test functions of different scaling difficulties.

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 for EDAs: 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.

Acknowledgments

The authors are grateful to the National Center for High-performance Computing for computer time and facilities.

References

[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] 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.

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

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

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

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

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

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

[20] 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.

[21] 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.

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

[23] 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.

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

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

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

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

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

[29] G. Schwarz, "Estimating the dimension of a model," *The Annals of Statistics*, vol. 6, no. 2, pp. 461–464, 1978.

[30] M. Pelikan, D. E. Goldberg, and K. Sastry, "Bayesian optimization algorithm, decision graphs, and occam's razor," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, L. Spector, E. D. Goodman, A. Wu, W. B. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. H. Garzon, and E. Burke, Eds. San Francisco, California, USA: Morgan Kaufmann, 2001, pp. 519–526.