Inductive Linkage Identification: Scalability, Robustness, and Population Sizing

Chung-Yao Chuang
Jih-Yiing Lin
Ying-ping Chen

NCLab Report No. NCL-TR-2009008
October 2009

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/

Inductive Linkage Identification: Scalability, Robustness, and Population Sizing

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

October 1, 2009

Abstract

This paper proposes a linkage identification algorithm, named *inductive linkage identification* (ILI), to identify linkages, which are referred to as the interdependencies among variables. The proposed algorithm utilizes the ID3 decision tree to extract sets of variables based on the mutual relevance on the fitness differences caused by perturbation. In this study, we concentrate on the properties and characteristics of ILI, including its scalability, robustness, and population requirement. According to the experimental results, compared to other linkage learning techniques, ILI exhibits equal or better flexibility, scalability, and robustness. A theoretical population sizing model is also developed in this paper to reveal the population requirement for ILI to operate. The proposed population sizing model well agrees with the experimental results and such a model may provide an insight into perturbation-based as well as entropy-based linkage learning methodologies.

1 Introduction

Genetic algorithms (GAs) have been widely adopted in handling optimization problems for their simplicity and applicability. In addition to population-based search, GAs explore the search space by combining pieces of promising solutions and require only the information from objective functions. However, the fixed presentations and problem-independent recombination operators, inherent in the early GAs, can break combined pieces of promising solutions and thus lead to the convergence to local optima. In order to overcome this difficulty, three categories of techniques have been proposed and developed [1].

The first kind of such approaches is to evolve representations or operators in an attempt to reduce disruptions of promising subsolutions. Yet, the evolution of representations or operators often cannot match the selection rates and usually results in premature convergence. The second category is probabilistic modeling for promising solutions. Methods in this category construct probability models of promising solutions and utilize the built model to generate new solutions. Though the model construction does not require additional function evaluations, there is a trade-off between model accuracy and computational cost. The approaches in the last category apply perturbation to identify the interdependencies among decision variables. These interdependencies, referred to as *linkage*, can be used to adjust or adapt representations and/or crossover operators to alleviate the aforementioned disruption problem. The main disadvantage is that the operation of linkage identification requires extra function evaluations.

In this paper, we propose a linkage identification technique based on analyzing the fitness changes caused by perturbation. The proposed algorithm, called *inductive linkage identification*

(ILI), adopts a population-wise perturbation approach and the ID3 decision tree to recognize sets of decision variables that exhibit strong relationships. The proposed algorithm is simple, efficient and requires no a priori knowledge about subproblem sizes. Series of experiments and a theoretical population sizing model are conducted and developed in this work to give a comprehensive understanding about ILI's properties and characteristics. The proposed algorithm can handle both uniformly and exponentially scaled problems with similar numbers of function evaluations. Experimental results further show that ILI scales well as well as exhibits robustness on large problem sizes. The proposed population sizing model provides a good agreement with empirical results on scalability and sheds a light on the population requirement of perturbation-based and entropy-based linkage identification approaches.

The rest of this paper is organized as follows. Section 2 introduces the background of linkage in genetic and evolutionary algorithms. The linkage problem and different categories of linkage handling techniques are introduced. Section 3 explains how ILI works. It firstly gives the terminologies that will be used throughout this paper, then provides a review of the ID3 decision tree learning algorithm, illustrates the proposed approach with an example, and finally describes the proposed algorithm in detail. The experiments and results are presented in section 4 for observing the behavior of ILI, including its scalability and robustness. Section 5 proposes a theoretical population sizing model for ILI and empirically verifies the proposed model. Finally, section 6 concludes this paper.

2 Background

An optimization problem can be defined by specifying the set of all solutions to that problem and a measure to evaluate the quality of each 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, i.e., the objective function. The only way of learning information about the relation between the semantics of solutions and the quality measure is to sample new candidate solutions and evaluate them with the objective function.

A popular way to deal with black-box optimization problems is to apply genetic algorithms (GAs) [2, 3]. 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) [4], 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 GA computational power. The Schema theorem [2] states that short, low-order, and highly fit sub-solutions increase their share to be combined and eventually form solutions at the end. Also as stated in the building block hypothesis [3], 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 often be reliably solved by combining the pieces of promising solutions in the form of problem decomposition.

The three features, the need of only the objective function, population-based search, and exploration by combining pieces of promising solutions, make GAs particularly suitable for black-box optimization. Because of the simplicity of the idea and its wide applicability, GAs have become an increasingly important area of computational optimization. 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 disruption can lead to premature convergence. 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, probabilistic modeling for promising solutions, and perturbation methods.

2.1 Evolving Representations or Operators

In order to alleviate the disruption problem of building blocks, techniques in this category adjust the representation of solutions during the search process, and thus building blocks can be less likely to be separated by crossover operators. Generally they encode both locations and values of each variable and apply a modified recombination operator on the chromosome. The messy GA (mGA) [5], its more efficient descendant—the fast messy GA (fmGA) [6], and the linkage learning GA (LLGA) [7] are some representatives in this category. Though these techniques incorporate the forming of tight linkage in the evolution, the progress of forming tight linkage is often much slower than the selection process. Consequently, premature convergence to local optima is the main drawback of the techniques in this category.

2.2 Probabilistic Modeling for Promising Solutions

The approaches in the second category are often referred to as estimation of distribution algorithms (EDAs) [8, 9, 10]. These methods construct probabilistic models of promising solutions and utilize the built model to generate new solutions. Early EDAs, such as the population-based incremental learning (PBIL) [11] and the compact genetic algorithm (cGA) [12], assume no interaction between variables, i.e., variables are independent. Subsequent studies start from capturing pairwise interactions, such as mutual-information-maximizing input clustering (MIMIC) [13], Baluja's dependency tree approach [14], and the bivariate marginal distribution algorithm (BMDA) [15], to modeling multivariate interactions, such as the extended compact genetic algorithm (ECGA) [16], the Bayesian optimization algorithm (BOA) [17], the factorized distribution algorithm (FDA) [18], and the learning version of FDA (LFDA) [19]. The model construction processes in these algorithms do not require additional function evaluations. Thus, they can perform effectively especially for the situations in which the performance are bounded by function evaluations. However, it is difficult for them to correctly model building blocks of low salience (small fitness contributions) [20].

2.3 Perturbation Methods

The methods in the third category examine the fitness differences obtained by conducting perturbations on the variables to detect interdependencies. For example, the gene expression messy GA (GEMGA) [21] records fitness changes caused by perturbation of each variable and detects relationships according to the possibilities that the variable may compose local optima. Linkage identification by nonlinearity check (LINC) [1] identifies the linkage via detecting the nonlinearity among variables after pairwise perturbations. It assumes that nonlinearity exists within interdependent variables. 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, the two variables can be viewed as to reside within different 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) [22], adopts non-monotonicity instead of nonlinearity and detects linkage by

checking violations of the monotonicity conditions. Although perturbation methods require extra function evaluations in addition to the running of GA, they have the advantage of being able to identify building blocks of low salience. [23] generalized this category through a Walsh analysis.

An intriguing algorithm combining the ideas of EDAs and perturbation, called *the dependency detection for distribution derived from fitness differences* (D^5), was developed by [20]. D^5 detects the dependencies of variables by estimating the distributions of strings clustered according to fitness differences. For each variable, D^5 calculates fitness differences by perturbing it for the entire population and clusters the strings into sub-populations according to the obtained fitness differences. The sub-populations are examined to find the k variables with the lowest entropies, where k is a pre-defined problem complexity (the number of variables in a linkage set). These k variables are assumed to be tightly linked as a linkage set. D^5 can detect dependencies for a class of functions that are difficult for EDAs (e.g., functions contain low-salient building blocks) and requires less computational cost than other perturbation methods do. However, the major constraint is that it relies on an input parameter k which may not be available due to the limited information of the problem structure. As a side-effect to the parameter k, D^5 might be fragile in the situation where the problem is composed of subproblems of different sizes.

In this work, we propose a perturbation-based linkage identification algorithm, called *inductive linkage identification* (ILI). ILI is similar to D^5 in that the population-wise perturbation approach is adopted and different from D^5 because instead of using a clustering method to obtain a biased sub-population, a supervised learning method, ID3 [24], which is well-established in the field of machine learning, is adopted to construct a decision tree for the task of extracting linkage groups. By inspecting the learned decision tree, we can obtain a set of variables exhibiting a strong relationship with the perturbed variable. The advantages of the proposed approach are that a lower number of function evaluations is needed and no problem complexity parameter (e.g., k in D^5) is required, and as a result, ILI is robust against problems composed of different-sized subproblems. Moreover, the experiments conducted in this work also demonstrate that ILI can scale well and is robust on the growth and mixing of different problem sizes. A population sizing model that reveals some important essence of population sizing in entropy-based linkage identification approaches is also developed.

3 Inductive Linkage Identification

In this section, we first briefly review some definitions and terminologies which will be used through out this paper. Then, we give a short review of ID3, followed by the demonstration of the idea behind the proposed linkage identification technique with an example. Finally, the detailed description of ILI is presented.

3.1 Additively Decomposable Functions

Through this work, we use additively decomposable functions (ADF) as the problem model because we concentrate on the problems containing certain structures. Let $s = s_1 s_2 \cdots s_\ell$, for ℓ variables, represent a string s of length ℓ. The fitness of string s can be defined as:

$$f(\mathbf{s}) = \sum_{i=1}^{m} f_i(\mathbf{s}_{v_i}) \ ,$$

where m is the number of subfunctions f_i, and s_{v_i} is the subset variables of s that corresponds to f_i. v_i here is a vector of indexes that specifies the corresponding subset variables s_{v_i}. For example, if $v_i = (1, 2, 4, 8)$, $\mathbf{s}_{v_i} = s_1 s_2 s_4 s_8$. Let V_i be the set that contains all the elements of v_i,

we can refer to the set V_i as a linkage set. In this paper, we consider only a subclass of ADFs. We concentrate on non-overlapping sub-functions, that is, $V_i \cap V_j = \emptyset$ if $i \neq j$. In addition, the strings are assumed to be composed of binary variables.

In the remainder of this paper, our approach will be demonstrated and evaluated on test problems constructed by concatenating several trap functions. Specifically, a k-bit trap function is a function of unitation[1] 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 GAs 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 [25, 26].

3.2 Decision Tree Learning: ID3

Decision tree learning is one of the most widely used methods for inductive inference. It has been successfully applied to a broad range of tasks, from diagnosing medical cases to accessing credit risks of loan applicants. Decision tree learning approximates discrete-valued target functions and represents these functions with decision trees.

In this paper, the ID3 decision tree learning algorithm [24] is used, and we mainly utilize ID3's ability in classification. For a classification problem, a training instance is composed of a list of attribute values describing the instance and a target value that the decision tree is supposed to predict after training. In our case, as described in section 3.4, the list of attribute values is the solution string, and the target value is the fitness difference caused by perturbation.

In its simplest form, ID3 constructs the decision tree in a top-down manner without backtracking. To construct a decision tree, each attribute is evaluated using a statistical property, called the *information gain*, to measure how well it alone classifies the training instances. The best attribute is selected and used as the test at the root node of the tree. A descendant of the root is then created for each possible value of this attribute, and the training instances are split into appropriate descendant nodes. The entire process is repeated using the training instances associated with each descendant node to select the best attribute to test at that point of the tree.

The statistical property, *information gain*, of each attribute is simply the expected reduction in the impurity of instances after the instances are classified by using that attribute. The impurity of an arbitrary collection of instances is often called *entropy* in the information theory. Given a collection D, containing instances of c different target values, the entropy of D relative to this c-wise classification is defined as

$$Entropy(D) \equiv \sum_{i=1}^{c} -p_i \log_2 p_i \ , \tag{1}$$

where p_i is the proportion of D belonging to class i. In all calculations involving entropy, we define $0 \log_2 0$ to be 0.

In terms of entropy, the information gain can be defined as follows. The information gain, $Gain(D, A)$, of an attribute A relative to a collection of instances D is

$$Gain(D, A) \equiv Entropy(D) - \sum_{v \in Val(A)} \frac{|D_v|}{|D|} Entropy(D_v) \ , \tag{2}$$

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

$s_1 s_2 \cdots s_8$	f	df_1
$\bar{0}1111\ 011$	0	-5
$\bar{0}0011\ 001$	3	1
$\bar{0}0100\ 000$	5	1
$\bar{0}1001\ 111$	5	1
$\bar{1}1111\ 000$	7	5
$\bar{0}1101\ 101$	1	1
$\bar{0}0110\ 011$	2	1
$\bar{0}1101\ 110$	1	1
$\bar{0}0001\ 011$	3	1
$\bar{1}0100\ 111$	5	-1
$\bar{1}1110\ 101$	0	-1
$\bar{1}1111\ 110$	5	5
$\bar{1}1011\ 010$	1	-1
$\bar{0}1000\ 010$	4	1
$\bar{0}0100\ 010$	4	1
$\bar{0}0001\ 000$	5	1
$\bar{0}1100\ 010$	3	1
$\bar{1}0000\ 101$	3	-1
$\bar{0}0000\ 100$	5	1
$\bar{1}1011\ 110$	0	-1
$\bar{0}0011\ 001$	3	1
$\bar{0}0111\ 010$	2	1
$\bar{0}0100\ 100$	4	1
$\bar{1}0110\ 000$	3	-1
$\bar{1}1100\ 000$	3	-1
$\bar{0}1111\ 111$	3	-5
$\bar{1}0100\ 010$	3	-1
$\bar{1}0100\ 001$	3	-1
$\bar{0}1000\ 001$	4	1
$\bar{0}1111\ 110$	0	-5

(a) Original population.

$s_1 s_2 \cdots s_8$	f	df_1
$\bar{0}0000\ 100$	5	1
$\bar{0}0001\ 011$	3	1
$\bar{0}0001\ 000$	5	1
$\bar{0}0100\ 000$	5	1
$\bar{0}0100\ 010$	4	1
$\bar{0}0100\ 100$	4	1
$\bar{0}1000\ 010$	4	1
$\bar{0}1000\ 001$	4	1
$\bar{0}1001\ 111$	5	1
$\bar{0}1100\ 010$	3	1
$\bar{0}1101\ 101$	1	1
$\bar{0}1101\ 110$	1	1
$\bar{0}0011\ 001$	3	1
$\bar{0}0011\ 001$	3	1
$\bar{0}0110\ 011$	2	1
$\bar{0}0111\ 010$	2	1
$\bar{0}1111\ 011$	0	-5
$\bar{0}1111\ 111$	3	-5
$\bar{0}1111\ 110$	0	-5
$\bar{1}0000\ 101$	3	-1
$\bar{1}0100\ 111$	5	-1
$\bar{1}0100\ 010$	3	-1
$\bar{1}0100\ 001$	3	-1
$\bar{1}0110\ 000$	3	-1
$\bar{1}1100\ 000$	3	-1
$\bar{1}1110\ 101$	0	-1
$\bar{1}1111\ 000$	7	5
$\bar{1}1111\ 110$	5	5
$\bar{1}1011\ 010$	1	-1
$\bar{1}1011\ 110$	0	-1

(b) Rearranged population.

Table 1: Population of strings. s_1 is perturbed.

where $Val(A)$ is the set of all possible values for attribute A, and D_v is the subset of D for which attribute A has value v.

3.3 Exemplary Illustration

To illustrate the idea behind ILI, suppose that we are dealing with an 8-bit problem formed by concatenating two trap functions,

$$f(s_1 s_2 \cdots s_8) = f_{trap_5}(s_1 s_2 s_3 s_4 s_5) + f_{trap_3}(s_6 s_7 s_8) \ ,$$

where $s_1 s_2 \cdots s_8$ is an individual. Our goal is to identify two linkage sets $V_1 = \{1, 2, 3, 4, 5\}$ and $V_2 = \{6, 7, 8\}$.

In the beginning, a population of strings is randomly generated as listed in Table 1(a). The first column lists the solution strings, and the second column lists the fitness values of the corresponding strings. After initializing the population, we perturb the first variable s_1 ($0 \rightarrow 1$

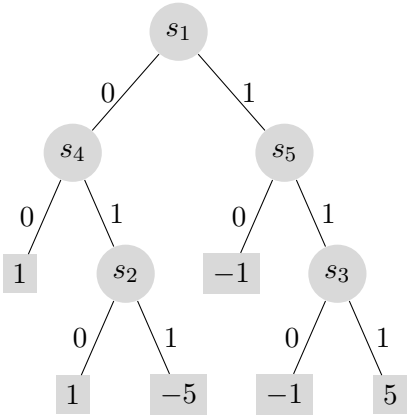


Figure 1: The ID3 decision tree constructed according to Table 1.

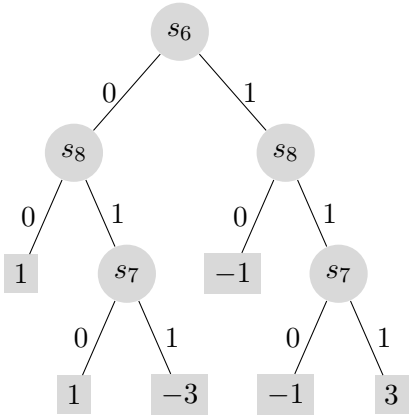


Figure 2: The ID3 decision tree constructed according to Table 2.

or $1 \rightarrow 0$) for all strings in the population in order to detect the linkage set in which the variables are related to s_1 (i.e., V_1). The third column of Table 1(a) records the fitness differences, df_1, caused by perturbations at variable s_1.

Then, we construct an ID3 decision tree by using the population of strings as the training instances. Each variable in $s_1 s_2 \cdots s_8$ is an attribute to the instances, and the target values are the fitness differences df_1. By using this setup, we can obtain an ID3 decision tree as shown in Figure 1. By gathering all the decision variables of the non-leaf nodes, we can identify a variable group, s_1, s_2, s_3, s_4, and s_5, which corresponds to linkage set V_1. As a consequence, the linkage set V_1 is correctly identified.

One might think that this result is a little too sudden. We may consider the rearranged population listed in Table 1(b) for a clearer view. In Table 1(b), strings from different blocks are bearing different patterns. For example, s_1 and s_4 of the strings from the first block are all 0's. In the fourth block, values of s_1 are 1's, and values of s_5 are 0's. Such an observation can be extended to other blocks as well. In fact, these patterns are corresponding to the paths from leaf nodes of the tree in Figure 1 to the root. To put it in another way, as we consider the fitness differences caused by perturbations as the target values of the decision tree, the ID3 algorithm selects variables showing strong relationship to the fitness differences. Thus the variables belonging to the same subfunction as the perturbed variable, s_1, tend to be selected in this mechanism.

A more accurate explanation can be given as follows. Consider the fitness difference df_1 of a certain string $\mathbf{s} = s_1 s_2 \cdots s_8$ perturbed at variable s_1:

$$\begin{aligned} df_1(\mathbf{s}) &= f(s_1 s_2 \cdots s_8) - f(\overline{s_1} s_2 \cdots s_8) \\ &= f_{trap_5}(s_1 s_2 s_3 s_4 s_5) + f_{trap_3}(s_6 s_7 s_8) \\ &\quad - f_{trap_5}(\overline{s_1} s_2 s_3 s_4 s_5) - f_{trap_3}(s_6 s_7 s_8) \\ &= f_{trap_5}(s_1 s_2 s_3 s_4 s_5) - f_{trap_5}(\overline{s_1} s_2 s_3 s_4 s_5) \ . \end{aligned} \tag{3}$$

As shown in Equation (3), the fitness difference df_1 is independent of the variables s_6, s_7, and s_8. df_1 depends on only s_1, s_2, ..., s_5. Therefore, for a sufficiently large population showing strong statistical evidences, the independent variables will not be chosen as nodes in the decision tree. On the other hand, because f_{trap_5} is a function with nonlinearity, all of the five variables tend to be identified given a sufficiently large population which contains nonlinear points of f_{trap_5}.

For the remainder of this example, since V_1 is already correctly identified, we proceed at s_6. The fitness differences after perturbations at variable s_6 are shown in Table 2. Employing the

$s_1 s_2 \cdots s_8$	f	df_6
01111 $\overline{0}$11	0	-3
00011 $\overline{0}$01	3	1
00100 $\overline{0}$00	5	1
01001 $\overline{1}$11	5	3
11111 $\overline{0}$00	7	1
01101 $\overline{1}$01	1	-1
00110 $\overline{0}$11	2	-3
01101 $\overline{1}$10	1	-1
00001 $\overline{0}$11	3	-3
10100 $\overline{1}$11	5	3
11110 $\overline{1}$01	0	-1
11111 $\overline{1}$10	5	-1
11011 $\overline{0}$10	1	1
01000 $\overline{0}$10	4	1
00100 $\overline{0}$10	4	1
00001 $\overline{0}$00	5	1
01100 $\overline{0}$10	3	1
10000 $\overline{1}$01	3	-1
00000 $\overline{1}$00	5	-1
11011 $\overline{1}$10	0	-1
00011 $\overline{0}$01	3	1
00111 $\overline{0}$10	2	1
00100 $\overline{1}$00	4	-1
10110 $\overline{0}$00	3	1
11100 $\overline{0}$00	3	1
01111 $\overline{1}$11	3	3
10100 $\overline{0}$10	3	1
10100 $\overline{0}$01	3	1
01000 $\overline{0}$01	4	1
01111 $\overline{1}$10	0	-1

Table 2: Population of strings. s_6 is perturbed.

identical procedure, an ID3 decision tree is constructed as presented in Figure 2. By inspecting the tree, we obtain the related variables s_6, s_7, and s_8 which form the size 3 linkage set V_2. The example illustrates that the proposed algorithm can handle problems composed of different-sized sub-problems.

3.4 Algorithm and Steps

The idea illustrated in previous sections forms the basis of the proposed algorithm, called *inductive linkage identification* (ILI), of which the procedure consists of the following three main steps:

1. Calculate the fitness differences caused by perturbation;
2. Construct an ID3 decision tree rooted at the perturbed variable;
3. Inspect the decision tree to obtain a linkage set.

Algorithm 1 Inductive Linkage Identification

procedure IDENTIFYLINKAGE(f, ℓ)
  Initialize a population P with n string of length ℓ.
  Evaluate the fitness of strings in P using f.
  $V \leftarrow \{1, \ldots, \ell\}$
  $m \leftarrow 0$
  while $V \neq \emptyset$ **do**
    $m \leftarrow m + 1$
    Select v in V at random.
    $V_m \leftarrow \{v\}$
    $V \leftarrow V - \{v\}$
    for each string $\mathbf{s^i} = s_1^i s_2^i \cdots s_\ell^i$ in P **do**
      Perturb s_v^i.
      $df^i \leftarrow$ fitness difference caused by perturbation.
    end for
    Construct an ID3 decision tree using (P, df) with v as root node.
    for each decision variable s_j in tree **do**
      $V_m \leftarrow V_m \cup \{j\}$
      $V \leftarrow V - \{j\}$
    end for
  end while
  return the linkage sets V_1, V_2, $\cdots$, V_m
end procedure

The three steps repeat until all the variables are divided into their corresponding linkage sets. In detail, ILI starts at initializing a population of strings. After initialization, ILI identifies one linkage set at a time using the following procedure: (1) a variable is randomly selected to be perturbed; (2) an ID3 decision tree with the perturbed variable as root is constructed according to the fitness differences caused by perturbations; (3) by inspecting the constructed tree, the variables used in the decision tree are collected and considered as a linkage set.

Algorithm 1 presents the pseudo code of the overall ILI procedure. As shown in Algorithm 1, the number of function evaluations required to accomplish the task of linkage identification is proportional to the number of the linkage sets of the given objective function. Suppose that we are dealing with an ADF f in which the length of solution strings is $\ell = k \times m$, where m is the number of subfunctions forming f, and k is the size of each subfunction. In this case, with the notation adopted by [20][2], LINC needs $\mathcal{O}(\ell^2) = \mathcal{O}(k^2 m^2)$ function evaluations, D^5 needs $\mathcal{O}(\ell) = \mathcal{O}(km)$ function evaluations, and ILI needs $\mathcal{O}(m)$ function evaluations. As a consequence, both ILI and D^5 need a number of function evaluations linear to the problem size, but ILI needs fewer evaluations by a factor of k, which is the subfunction size. The numerical results verifying this theoretical observation are included in the next section.

4 Comparative Evaluations, Scalability, and Robustness

Experimental settings and numerical results are presented in this section. The experiments are designed to indicate the population requirement for ILI to work correctly on problems composed

[2] [20] separates the discussion of population sizes and extra function evaluations used in linkage detection. To discuss the number of function evaluations needed for linkage identification, it is assumed that the population size is sufficiently large to capture all nonlinearity of the objective function. Such an assumption is the premise for perturbation methods, such as LINC, to work correctly.

of subproblems of different complexities. In this study, we focus on the problems composed of m concatenated non-overlapping trap functions as subproblems, which can be described as

$$f(\mathbf{s}) = \sum_{i=1}^{m} f_{trap_k}(s_{(i-1)k+1} \cdots s_{(i-1)k+k}) \ ,$$

where k is the size of subproblems (i.e., subproblem complexity), and m is the number of subproblems.

For each problem instance, the goal is to determine the minimum population size required by ILI to correctly identify all the linkage sets. The criterion to check whether or not a population size is sufficiently large is that ILI can work as expected in 30 consecutive, independent runs. The experimental procedure runs in a bisection style. An upper bound and a lower bound (2500 and 0 respectively in this study) are set to initialize the experiments. The population size to test is the middle value of the current upper and lower bounds. If linkage identification is successful, i.e., ILI can correctly identify all the linkage sets in 30 consecutive, independent runs, the current middle value will be the next upper bound. If linkage identification is unsuccessful, the current middle value will be the next lower bound. The procedure repeats until the difference between the upper and lower bounds is less than or equal to 2.

4.1 Comparative Evaluations

In the first series of experiments, we compare the proposed algorithms with two related methods, LINC and D^5, on binary ADFs with non-overlapping subfunctions for comparison. The numerical results on uniformly scaled functions as well as on exponentially scaled functions are also compared to demonstrate the flexibility of ILI.

The empirical comparison of the proposed approach with LINC and D^5 on uniformly scaled functions is performed on the functions composed of $trap_5$ subfunctions:

$$f(\mathbf{s}) = \sum_{i=1}^{m} f_{trap_5}(s_{5\cdot(i-1)+1} \cdots s_{5\cdot(i-1)+5}) \ ,$$

where m ranges from 20 to 180. That is, the problem size ranges from 100 bits to 900 bits.The results of ILI are compared to that of LINC and D^5 [20] and plotted in Figure 3. The number of function evaluations is much lower than that needed by LINC and grows in a relatively slow rate. It is also lower than D^5 about a factor of the subfunction size, $k = 5$. The results confirm our previous theoretical observation.

Moreover, Figure 4 presents the population requirement of ILI on exponentially scaled functions, in which the $trap_5$ function is also adopted:

$$f(\mathbf{s}) = \sum_{i=1}^{m} 2^{i-1} \times f_{trap_5}(s_{5\cdot(i-1)+1} \cdots s_{5\cdot(i-1)+5}) \ ,$$

where m ranges from 10 to 50. That is, the problem size ranges from 50 bits to 250 bits. The results are compared to that of ILI on uniformly scaled functions. It can be observed that ILI needs approximately the same number of function evaluations for problems of same sizes. That ILI is independent of different subproblem scalings can be concluded according to the observation.

4.2 Scalability

In this section, we exam the scalability of ILI on different problem sizes for both population sizes and function evaluations. The number of subproblems, m, and the size of subproblems,

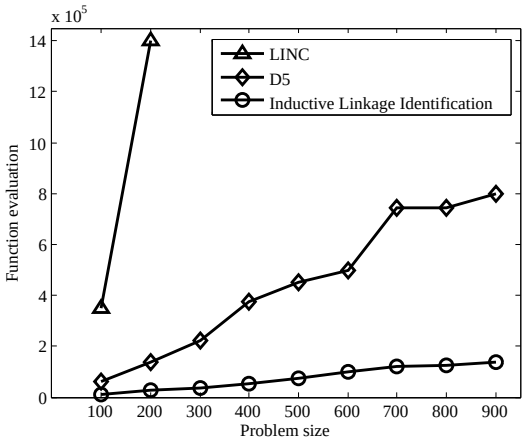


Figure 3: Numerical results of ILI compared to that of LINC and D^5 [20] on uniformly scaled problems.

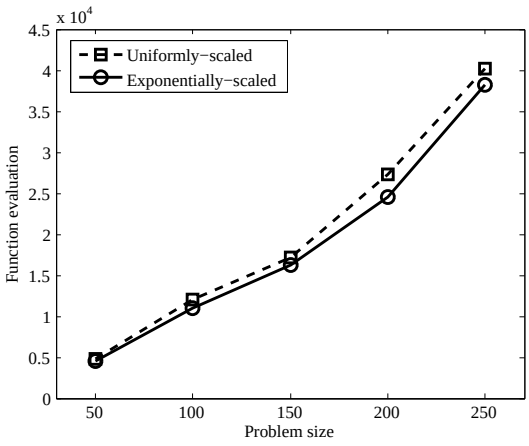


Figure 4: ILI needs approximately the same number of function evaluations for problems of same sizes.

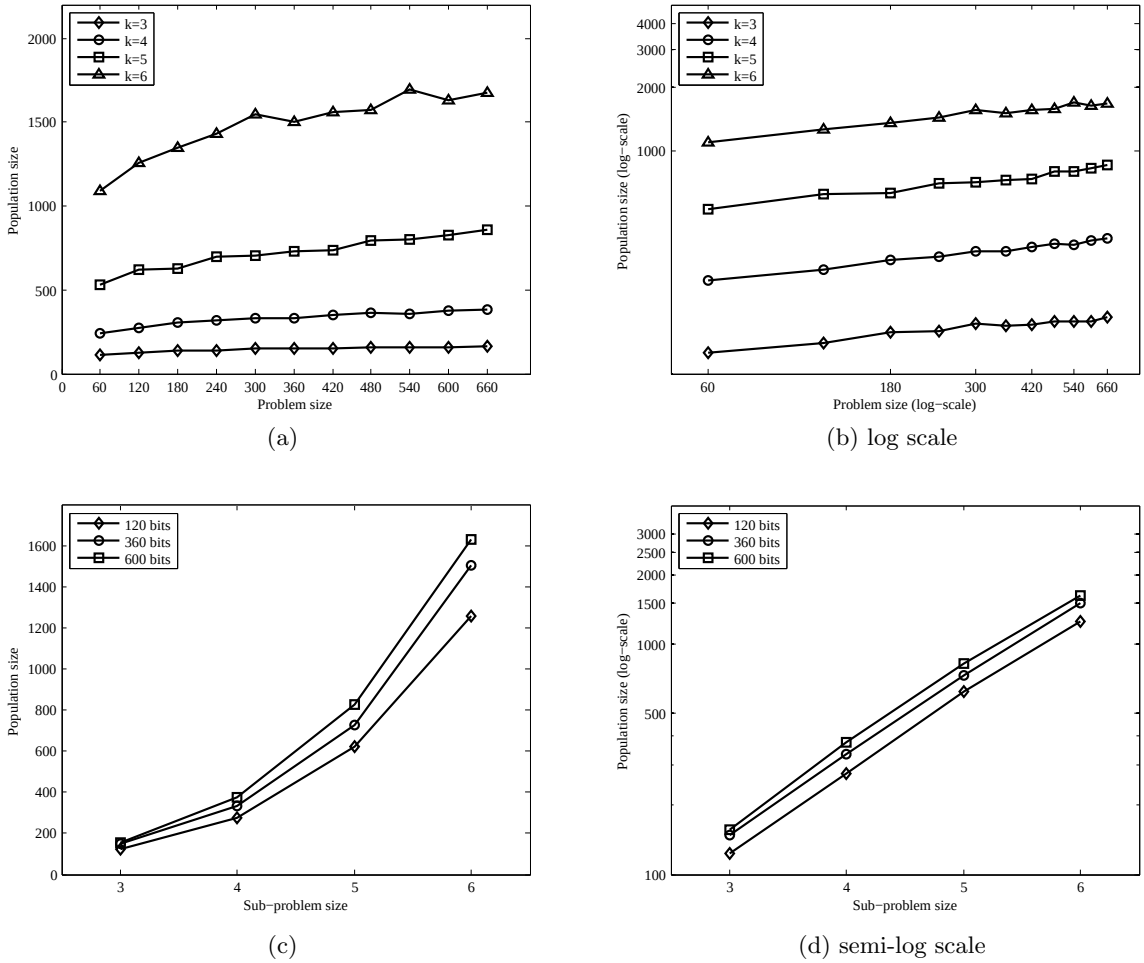
k, are two key elements of problem sizes, ℓ. To clearly identify the effect of each element, we adopt non-overlapping ADFs as testbeds. The problem sizes are the sum of all the sizes of the subproblems. In order to examine the scalability of population sizes and function evaluations on the number and size of subproblems, we empirically determine the population sizes and function evaluations required for k from 3 to 6 with various numbers of subproblems. The overall problem sizes, ℓ, are 60, 120, 180, $\cdots$, 600 bits, and m is calculated by $m = \ell/k$.

The results of population sizes and function evaluations are illustrated in Figures 5 and Figure 6, respectively. Figure 5(a) and 5(b) show that the population size required by ILI grows sub-linearly with the problem size when the subproblem complexity is fixed. This observation indicates that the population requirement of ILI is relatively insensitive to the overall problem size and grows sub-linearly with the number of subproblems. Figure 5(c) and Figure 5(d) show the scalability of ILI on population sizes against subproblem complexity. Each line in these two figures corresponds to a fixed overall problem size. The straight lines for $\ell = 120$, 360, and 600 in Figure 5(d), of which the y-axis is log-scaled, indicate that for a fixed overall problem size, the population size required by ILI grows exponentially with the complexity of subproblems. Overall, the required population size grows sub-linearly with the number of subproblems while it grows exponentially with the size of subproblems.

Figure 6 shows the numerical results of function evaluations against various problem sizes as well as subproblem sizes. The straight lines in Figures 6(a) and 6(b) indicate the liner growth of function evaluations with overall problem sizes when the subproblem size is fixed. Figures 6(c) and 6(d) illustrate the scalability of function evaluations on the subproblem complexity. The straight lines in Figure 6(d), of which the y-axis is log-scale, indicate that function evaluations grow exponentially with the complexity of subproblems because of the exponentially-growing population requirement.

4.3 Robustness

In this section, experiments are conduct to investigate the robustness of ILI against the population sizes that are smaller than the population requirement. As aforementioned, existing linkage detection/identification methods, including ILI, LINC, and D^5, basically assume to work on a sufficiently large population that is able to provide statistically significant evidence to reveal the interdependency among decision variables. However, in practice, an insufficiently large popula-



(a) (b) log scale

(c) (d) semi-log scale

Figure 5: Results of ILI for population sizes: (a, b) Population size versus problem size with different subproblem sizes; (c, d) Population size versus subproblem size with different problem sizes.

tion may be employed by the user and consequently result in the failure of linkage identification. Therefore, investigating the robustness of ILI against the population size is necessary and may provide some helpful information to ILI users.

Based on the measured population requirement in section 4.2, we decrease the population size to the 90%, 80%, ..., 50% of the requirement to see how well ILI can perform in terms of the ratio of linkage sets that can be correctly identified. Figure 7 shows the identification rate for different subproblem sizes and overall problem sizes. According to the experimental results, we can know that within a decrement of 20% of the requirement population size, ILI can still recognize about 90% of the linkage sets disregarding the problem size or subproblem complexity. Such an observation implies that ILI is quite robust and responds well to population sizes that are not large enough.

5 Population Sizing for ILI

In this section, we theorize the linkage identification mechanism of ILI and accordingly derive the population sizing model which can indicate a sufficiently large population to ensure successful

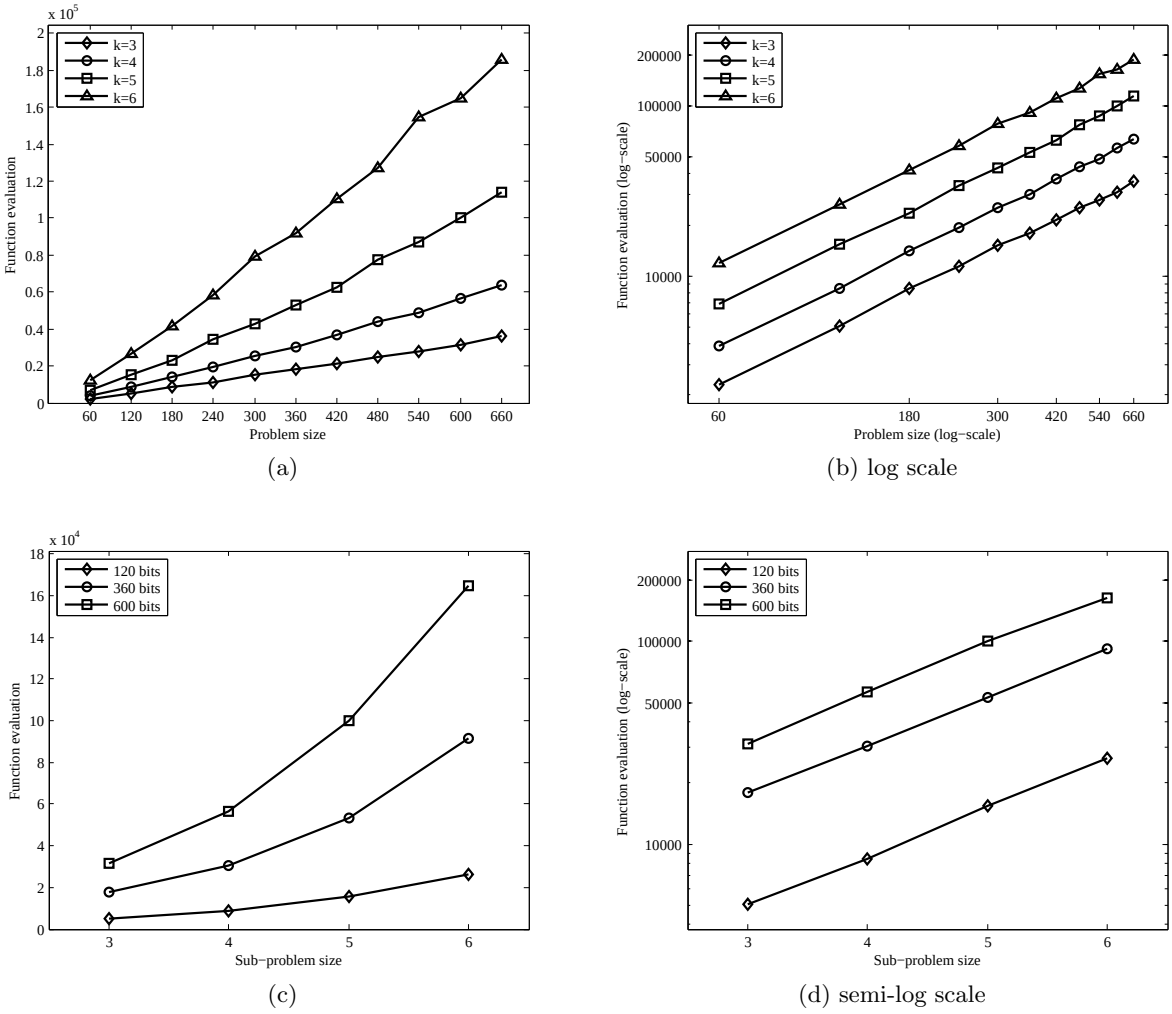


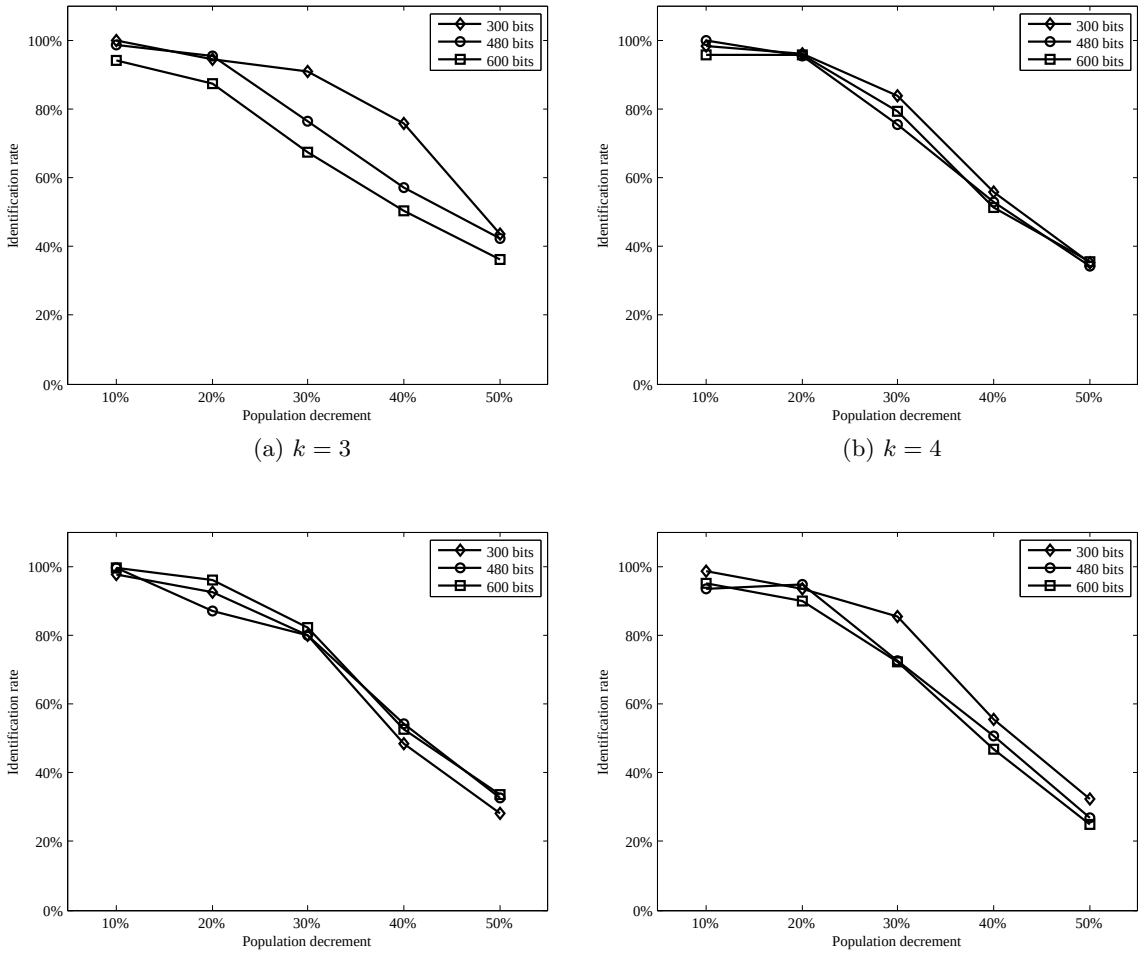
Figure 6: Results of ILI for function evaluations: (a, b) Function evaluation versus problem size with different subproblem sizes; (c, d) Function evaluation versus subproblem size with different overall problem sizes.

linkage identification. We will firstly establish a population sizing model by analyzing the core mechanism of ILI and then empirically verify the derived model. Finally, we will discuss several aspects and implications of the proposed model.

5.1 Model Derivation

Since ILI adopts the ID3 decision tree to identify linkage, we firstly investigate the relationship between the population size, n, and the probability of selecting a wrong decision variable in the decision tree, p_{terr}. We assume that the objective function consists of non-overlapping subfunctions and the sampling of objective function is noise free. Since every variable of each individual is generated at random, the subpopulation size of individuals with a same substring $s_i s_{i+1} \cdots s_{i+r-1} = a_0 a_1 \cdots a_{r-1}$, denoted as n_{s^r}, is a binomial distribution with $n =$ population size and $p = 2^{-r}$. When n is sufficiently large, an excellent approximation of such a binomial distribution can be obtained via the normal distribution:

$$N(np, np(1-p)) \ .$$



(a) $k = 3$ (b) $k = 4$

(c) $k = 5$ (d) $k = 6$

Figure 7: Numerical results of ILI robustness test on subproblems of different sizes.

For a more detailed illustration, we start from using the following function as a demonstrative example:

$$f(\mathbf{s}) = \sum_{i=1}^{m} f_{trap_3}(s_{3\cdot(i-1)+1} \cdots s_{3\cdot(i-1)+3}) \tag{4}$$

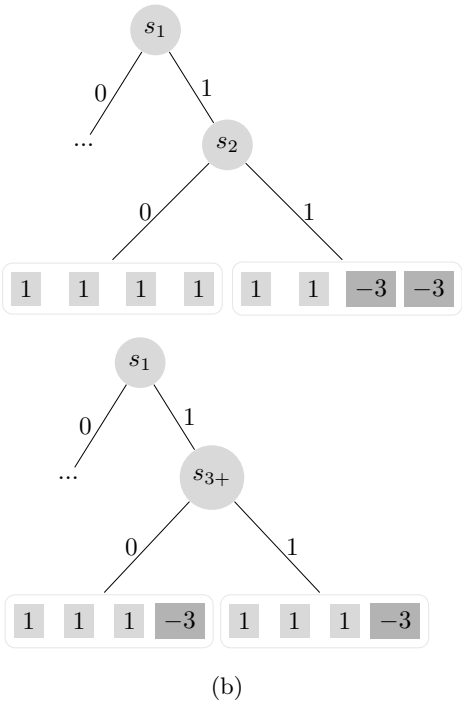
$$= f_{trap_3}(s_1 s_2 s_3) + \sum_{i=2}^{m} f_{trap_3}(s_{3\cdot(i-1)+1} \cdots s_{3\cdot(i-1)+3}) \,. \tag{5}$$

To identify the linkage set $V_1 = \{1, 2, 3\}$, the first variable s_1 is perturbed. Figure 8(a) lists the fitness differences of individuals categorized by $s_1 s_2 s_3 s_{3+}$ values. For example, the first row indicates the fitness difference of any individual with its $s_1 s_2 s_3 s_{3+} = \overline{0}000$ is 1. The f_o column denotes the original fitness value, and the f_n column denotes the new fitness value after perturbation. The $\overline{0}$ denotes that the corresponding variable has been perturbed from 1 to 0, and s_{3+} denotes a specified variable other than s_1, s_2, and s_3. With a sufficient population size n, it is reasonable to approximate the subpopulation size of each $s_1 s_2 s_3 s_{3+}$-typed individuals as a normal distribution with a mean of $2^{-4} \cdot n$.

Figure 8(b) illustrates the expected fitness difference distribution in subpopulations classified by decision variables $s_1 s_2$ and $s_1 s_{3+}$, respectively. Since the fitness difference distributions are

s_1	s_2	s_3	s_{3+}	f_o	f_n	df_1
$\overline{0}$	0	0	0	1	2	1
$\overline{0}$	0	0	1	1	2	1
$\overline{0}$	0	1	0	0	1	1
$\overline{0}$	0	1	1	0	1	1
$\overline{0}$	1	0	0	0	1	1
$\overline{0}$	1	0	1	0	1	1
$\overline{0}$	1	1	0	3	0	-3
$\overline{0}$	1	1	1	3	0	-3
$\overline{1}$	0	0	0	2	1	-1
$\overline{1}$	0	0	1	2	1	-1
$\overline{1}$	0	1	0	1	0	-1
$\overline{1}$	0	1	1	1	0	-1
$\overline{1}$	1	0	0	1	0	-1
$\overline{1}$	1	0	1	1	0	-1
$\overline{1}$	1	1	0	0	3	3
$\overline{1}$	1	1	1	0	3	3

(a)



(b)

Figure 8: Fitness differences and decision tree construction for scenario I.

similar for $s_1 = 0$ and $s_1 = 1$, for simplicity, we omit the left sibling $s_1 = 0$. At the top of the tree, all the individuals with their $s_1 s_2 = 10$ have $df_1 = 1$, and the $s_1 s_2 = 11$ individuals have roughly half $df_1 = 1$ and half $df_1 = -3$. Meanwhile, at the bottom, $df_1 = -3$ individuals is expected to distribute equally in each subpopulations with a quarter share. This fitness difference distribution symmetry inherent in the $s_1 s_{3+}$ tree grounds a higher entropy state than that of the asymmetric $s_1 s_2$ tree. According to the expected fitness difference distribution and the information gain defined in Equation (2), taking s_2 as a tree node can achieve a lower entropy state than taking s_{3+} can. Likewise, taking the variables in the linkage set as decision tree nodes can achieve lower entropy states than taking others can. Thus, selecting wrong variables during the decision tree construction does not often occur.

Of course, there are still chances for a decision tree to take a wrong variable as an internal node. By observing the decision trees shown in Figure 8(b), we can see that a wrong variable will be selected when one of the subpopulation of $s_1 s_{3+}$ does not contain $df_1 = -3$ individuals and the size of the other subpopulation is less than that of $s_1 s_2 = 11$ subpopulation. Figure 9(a) illustrates one of such scenarios, in which $s_1 s_2 s_3 s_{3+} = \overline{0}110$ is absent. The corresponding $s_1 s_2$ and $s_1 s_{3+}$ trees are illustrated in Figure 9(b). Since the $df_1 = -3$ individuals in both trees are identical, once the subpopulation size of $s_1 s_{3+} = 11$ individuals is less than that of $s_1 s_2 = 11$, selecting s_{3+} as a node will achieve a lower entropy state and thus will introduce a wrong variable into the linkage set.

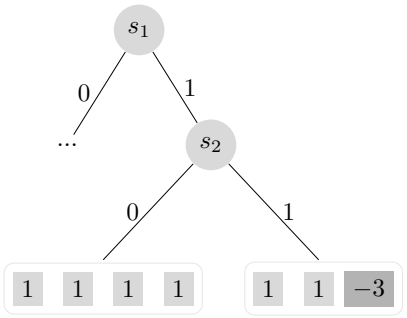
In general, since there are around half of the population in the $s_1 = 1$ sibling, the subpopulation size of $s_1 s_{3+} = 11$ individuals, $n_{s_{3+}}$, can be approximated with a normal distribution as

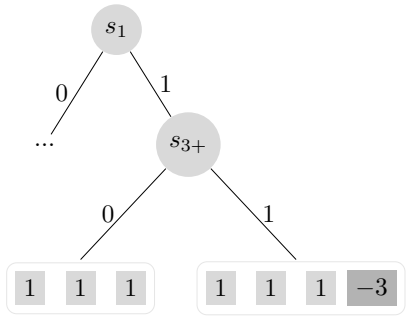
$$N(\frac{2^2 n}{2(2^3-1)}, \frac{2^2(2^2-1)n}{2(2^3-1)^2}) = N(\frac{4n}{14}, \frac{12n}{98}) \; .$$

It is in fact a binomial distribution with population size $= n/2$ and $p = 4/7$, which corresponds

s_1	s_2	s_3	s_{3+}	df_1
$\overline{0}$	0	0	0	1
$\overline{0}$	0	0	1	1
$\overline{0}$	0	1	0	1
$\overline{0}$	0	1	1	1
$\overline{0}$	1	0	0	1
$\overline{0}$	1	0	1	1
~~$\overline{0}$~~	~~1~~	~~1~~	~~0~~	~~-3~~
$\overline{0}$	1	1	1	-3
$\overline{1}$	0	0	0	-1
$\overline{1}$	0	0	1	-1
$\overline{1}$	0	1	0	-1
$\overline{1}$	0	1	1	-1
$\overline{1}$	1	0	0	-1
$\overline{1}$	1	0	1	-1
$\overline{1}$	1	1	0	3
$\overline{1}$	1	1	1	3

(a)





(b)

Figure 9: Fitness differences and decision tree construction for scenario II.

to the proportion of $s_1 s_{3+} = 11$ individuals in the whole population. The subpopulation size of $s_1 s_2 = 11$ individuals in $s_1 = 1$ sibling, n_{s_2}, can also be approximated with a normal distribution as

$$N(\frac{(2^2-1)n}{2(2^3-1)}, \frac{2^2(2^2-1)n}{2(2^3-1)^2}) = N(\frac{3n}{14}, \frac{12n}{98}) \; . \tag{6}$$

Similarly, n_{s_3} has the same distribution. In this scenario, when $n_{s_{3+}}$ is less than the largest among $n_{s_{i \in V_1}}$, it is possible for s_{3+} to be selected as a decision tree node. Moreover, when $n_{s_{3+}}$ is less than the smallest among $n_{s_{i \in V_1}}$, s_{3+} definitely would be taken as a decision variable. Since the distribution of each $n_{s_{i \in V_1}}$ is identical, the largest among $n_{s_{i \in V_1}}$ can be considered as the largest number sampled from the normal distribution described by Equation (6). In other words, it is a second order statistic. The smallest among $n_{s_{i \in V_1}}$ is a first order statistic. Therefore, p_{terr} of the $s_1 = 1$ sibling, when $s_1 s_2 s_3 s_{3+} = \overline{0}110$ is absent, can be estimated as

$$\Phi(\frac{\mu_{X_{(1:2)}} - \mu_{s_{3+}}}{\sigma_{s_{3+}}}) \leq p_{terr} \leq \Phi(\frac{\mu_{X_{(2:2)}} - \mu_{s_{3+}}}{\sigma_{s_{3+}}}) \; ,$$

where Φ denotes the cumulative distribution function of the standard normal function, $\mu_{X_{(1:2)}}$ denotes the mean of the first order statistic in a sample of size two while $\mu_{X_{(2:2)}}$ denotes the mean of the second order statistic. $\mu_{s_{3+}}$ denotes the mean of $n_{s_{3+}}$, and $\sigma_{s_{3+}}$ denotes the standard deviation of $n_{s_{3+}}$.

Now we consider a more general case of a linkage set of k variables, $V = \{1, 2, \cdots, k\}$, and revisit Figure 8(a) for a closer look at the fitness difference induced by perturbing a bit in a trap function. In the case of alternating a bit from one to zero, one is added to the fitness of each individual except those who originally have the fitness value k. These exceptions have a new fitness of zero, and hence fitness difference $-k$. In the case of alternating a bit from zero

to one, this reduces one from the fitness of each individual except those who originally have a fitness value of zero. These exceptions have a new fitness of k, and hence fitness difference k. As depicted, only those $s_2 s_3 \cdots s_k = 11 \cdots 1$ individuals have fitness $\pm k$. In this way, their decision tree on $s_{i \in V}$ and s_{k+} resembles those in Figure 8(b). As the discussion for $k = 3$ example, there are two scenarios in which the $s_1 s_{k+}$ tree may achieve a lower entropy state in the $s_1 = 1$ sibling: when either $s_1 s_2 \cdots s_k s_{k+} = \overline{0}1 \cdots 10$ or $s_1 s_2 \cdots s_k s_{k+} = \overline{0}1 \cdots 11$ individuals are absent. Therefore, we can similarly approximate the subpopulation size of $s_1 s_{k+} = 11$ individuals in $s_1 = 1$ sibling, $n_{s_{k+}}$, with a normal distribution as

$$N(\frac{2^{k-1}}{2(2^k-1)}n, \frac{2^{k-1}(2^{k-1}-1)}{2(2^k-1)^2}n) \ .$$

The subpopulation size of $s_1 s_{i \in V} = 11$ individuals, n_{s_V}, can be approximated with a normal distribution as

$$N(\frac{(2^{k-1}-1)}{2(2^k-1)}n, \frac{2^{k-1}(2^{k-1}-1)}{2(2^k-1)^2}n) \ . \tag{7}$$

Likewise, p_{terr} on the $s_1 = 1$ sibling tree when $s_1 s_2 \cdots s_{k+} = \overline{0}1 \cdots 10$ is absent can be estimated as

$$\Phi(\frac{\mu_{X_{(1:k-1)}} - \mu_{s_{k+}}}{\sigma_{s_{k+}}}) \leq p_{terr} \leq \Phi(\frac{\mu_{X_{(k-1:k-1)}} - \mu_{s_{k+}}}{\sigma_{s_{k+}}}) \ ,$$

where Φ denotes the cumulative distribution function of the standard normal function, $\mu_{X_{(1:k-1)}}$ denotes the mean of the first order statistic in a sample of size $k-1$ while $\mu_{X_{(k-1:k-1)}}$ denotes the mean of the $(k-1)$-th order statistic. $\mu_{s_{k+}}$ denotes the mean of $n_{s_{k+}}$, and $\sigma_{s_{k+}}$ denotes the standard deviation of $n_{s_{k+}}$. In the following sections for the verification purpose, we will use the term

$$\Phi(\frac{\mu_{X_{(1:k-1)}} - \mu_{s_{k+}}}{\sigma_{s_{k+}}}) \tag{8}$$

as a lower bound of p_{terr} to compute an upper bound of the population size and the term

$$\Phi(\frac{\mu_{X_{(k-1:k-1)}} - \mu_{s_{k+}}}{\sigma_{s_{k+}}}) \tag{9}$$

as an upper bound of p_{terr} to compute a lower bound of the population size.

Since the two scenarios for selecting wrong decision variables are symmetric, p_{terr} is identical. For each scenario to occur, all the individuals in the subpopulation of $s_1 = 1$ sibling should not contain the absent substring. Because there are roughly half of the population in the $s_1 = 1$ sibling and the probability for an individual to contain the absent substring is 2^{-k}, we can estimate the probability as $(1-2^{-k})^{n/2}$. Thus, the total probability of selecting a wrong decision variable s_{k+} in the $s_1 = 1$ sibling is

$$2 \cdot (1-2^{-k})^{n/2} \cdot p_{terr} \ .$$

The probability for one sibling to identify linkage correctly is

$$1 - 2 \cdot (1-2^{-k})^{n/2} \cdot p_{terr} \ .$$

Since there are two siblings from the root, $(\ell - k)s_{k+}$ candidates, and m decision trees, we can calculate the probability for ILI to correctly identify the linkage set as

$$p_\alpha = [1 - 2 \cdot (1-2^{-k})^{n/2} \cdot p_{terr}]^{2m(\ell-k)} \ . \tag{10}$$

$\mu_{2:2}$	$\mu_{3:3}$	$\mu_{4:4}$	$\mu_{5:5}$	$\mu_{1:2}$	$\mu_{1:3}$	$\mu_{1:4}$	$\mu_{1:5}$
0.564	0.846	1.029	1.163	−0.564	−0.846	−1.029	−1.163

Table 3: Mean values of the normal order statistics for different k.

5.2 Empirical Verification of the Model

In section 4, the population sizes required by ILI to correctly identify all the linkage sets on different problem sizes are empirically measured. Since we conduct the experiment by using 30 consecutive and independent successful runs as the judgment criterion, the 95% confidence interval of p_α is 0.884 to 1.0 according to statistics [27]. Thus, we select $p_\alpha = 0.942$, the middle point of the interval, for the proposed model described by Equation (10) to estimate the population sizes. In order to conduct a thorough verification of the derived model, the estimated population sizes are compared for k, the subproblem complexity, from 3 to 6. Table 3 lists the means of normal order statistics for different k's [28]. $\mu_{x:n}$ denotes the mean of the x-th order statistic in a sample of size n when the distribution is standard normal.

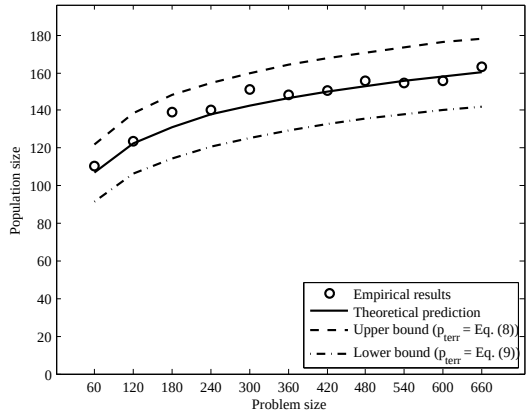
Hence, the mean values of the i-th order statistics in a sample of size $k-1$ from the subpopulation size distribution of linkage set variables can be calculated as

$$\mu_{X_{i:k-1}} = \mu_{s_{ls}} + \mu_{i:k-1}\sigma_{s_{ls}} \ ,$$

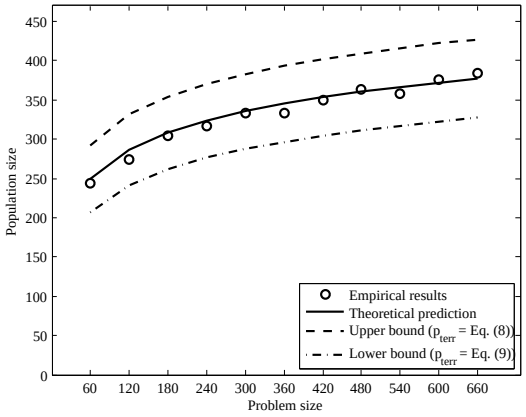
where $\mu_{s_{ls}}$ and $\sigma_{s_{ls}}$ are the mean and standard deviation of Equation (7), respectively. Applying the aforementioned setting of p_α to Equation (10), the estimated population sizes as well as the corresponding empirical results are illustrated in Figure 10. The circle marks represent the empirically obtained population sizes, and the solid lines are predicted population sizes according to the proposed population sizing model (Equation (10)). Figure 11 further illustrates the population requirement of ILI for problems of different subproblem complexity. In Figure 11, the marks represent the obtained population sizes for problems of lengths 120, 360, and 600 bits, and the lines are predicted population sizes. Figures 10 and 11 indicate that the proposed population sizing model is able to provide a very good approximation for the population sizes required by ILI to identify the linkage sets in trap functions for different overall problem sizes as well as subproblem sizes.

5.3 Discussion

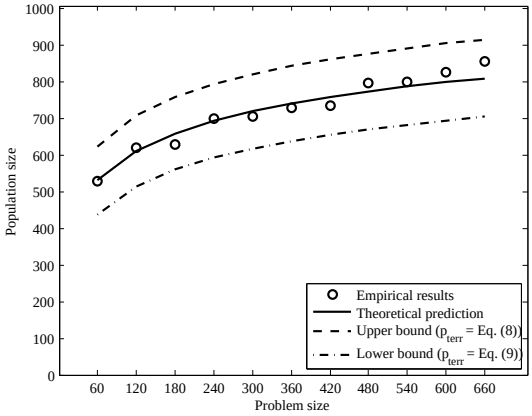
Although we developed a population sizing model based on the properties and characteristics of trap functions, several aspects actually support that the proposed model provides an approximated upper bound of population sizes for a broad range of popular test functions. Observing the fitness difference list given in Table 4, we can find that the fitness difference distributions of $trap_4$ and $nith_4$ are similar. Moreover, taking a further investigation on the fitness difference distribution of $valley_4$ and $trap_4$, the p_{terr} of $valley_4$, though with a slight difference, should be similar to that of $trap_4$. On the other hand, p_{terr} of $tmmp_4$ should be smaller because there are more fitness difference classes in each subpopulation and variables in the linkage set are easier to be selected. In general, any of these test functions shall hold a p_{terr} smaller than or equal to that of $trap_4$. Figure 12 shows the population sizes required by ILI on problems composed of different subproblem types ($k = 4$) as well as the computed theoretical bounds. All the population sizes are well within the bounds and imply that the derived population sizing model can also be used to compute population sizes for other functions.



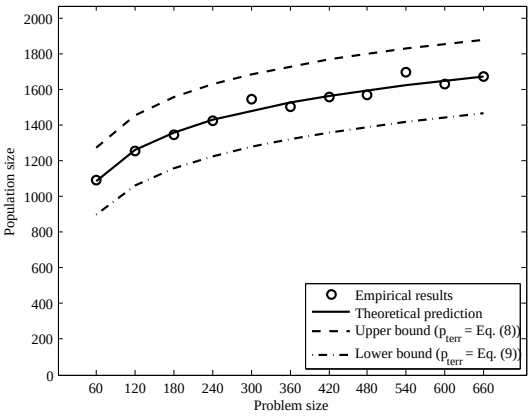
(a) $k = 3$



(b) $k = 4$



(c) $k = 5$



(d) $k = 6$

Figure 10: Population sizing model: Theoretical prediction vs. empirical results.

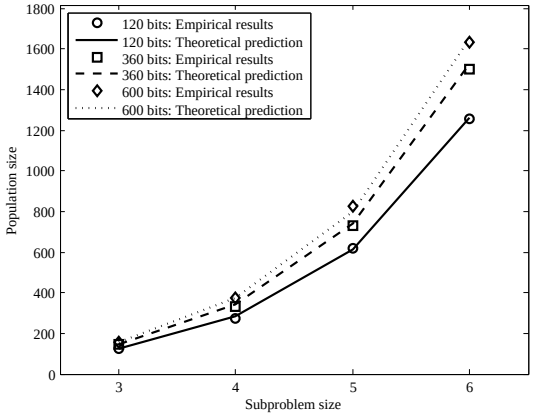


Figure 11: Population sizing model: Theoretical prediction vs. empirical results for different k.

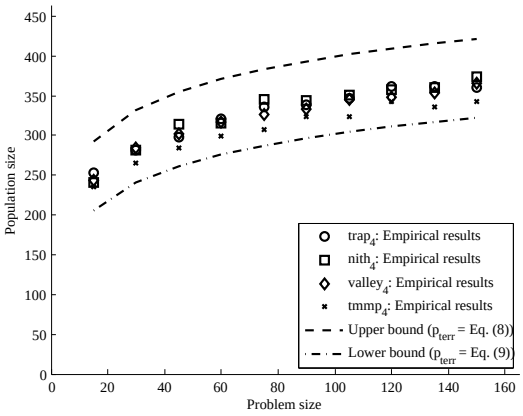


Figure 12: Population sizing model: Theoretical prediction vs. empirical results for different subproblem types.

substring				$trap_4$			$nith_4$			$valley_4$			$tmmp_4$		
s_1	s_2	s_3	s_4	f_o	f_n	df_1	f_o	f_n	df_1	f_o	f_n	df_1	f_o	f_n	df_1
$\overline{0}$	0	0	0	2	3	1	0	0	0	2	4	2	1	0	-1
$\overline{0}$	0	0	1	1	2	1	0	0	0	0	2	2	2	1	-1
$\overline{0}$	0	1	0	1	2	1	0	0	0	0	2	2	2	1	-1
$\overline{0}$	0	1	1	0	1	1	0	0	0	2	0	-2	1	2	1
$\overline{0}$	1	0	0	1	2	1	0	0	0	0	2	2	2	1	-1
$\overline{0}$	1	0	1	0	1	1	0	0	0	2	0	-2	1	2	1
$\overline{0}$	1	1	0	0	1	1	0	0	0	2	0	-2	1	2	1
$\overline{0}$	1	1	1	4	0	-4	4	0	-4	4	2	-2	4	1	-3
$\overline{1}$	0	0	0	3	2	-1	0	0	0	4	2	-2	0	1	1
$\overline{1}$	0	0	1	2	1	-1	0	0	0	2	0	-2	1	2	1
$\overline{1}$	0	1	0	2	1	-1	0	0	0	2	0	-2	1	2	1
$\overline{1}$	0	1	1	1	0	-1	0	0	0	0	2	2	2	1	-1
$\overline{1}$	1	0	0	2	1	-1	0	0	0	2	0	-2	1	2	1
$\overline{1}$	1	0	1	1	0	-1	0	0	0	0	2	2	2	1	-1
$\overline{1}$	1	1	0	1	0	-1	0	0	0	0	2	2	2	1	-1
$\overline{1}$	1	1	1	0	4	4	0	4	4	2	4	2	1	4	3

Table 4: Fitness differences of different sub functions

6 Conclusion

In this paper, we proposed a linkage learning approach, called inductive linkage identification (ILI), that adopts the ID3 decision tree construction algorithm to learn the linkage group to which the perturbed variable belongs. The proposed approach is simple and concise because only perturbation and the ID3 algorithm are utilized to extract the linkage information. In addition to its simplicity, compared to other related, existing linkage detection methods, ILI needs fewer function evaluations to accomplish its task and similar numbers of function evaluations on uniformly exponentially scaled problems. In order to investigate the scalability and robustness of ILI, series of experiments were conducted in this study. The empirical results indicate that the required population size and function evaluations of ILI grow linearly or sub-linearly with the problem size, while they grow exponentially with the subproblem complexity. It implies that if the size of subproblems is fixed, ILI does exhibit good scalability on the problem size. In the experiments for robustness, we found that ILI is rather robust because the successful rate drops slightly as the population size decreases. Finally, a theoretical population sizing model was developed in this paper. The proposed model, derived based on a statistical basis, not only well agreed with the empirical results but also revealed some insights of population sizing in perturbation-based and entropy-based linkage identification methods.

Acknowledgments

The work was supported in part by the National Science Council of Taiwan under Grant NSC 98-2221-E-009-072. The authors are grateful to the National Center for High-performance Computing for computer time and facilities.

References

[1] 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.

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

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

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

[5] 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.

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

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

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

[9] P. Larrañaga and J. A. Lozano, *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation.* Boston, MA: Kluwer Academic Publishers, 2001.

[10] 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.

[11] 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.

[12] 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.

[13] J. de Bonet, C. Isbell, and P. Viola, "MIMIC: Finding optima by estimating probability densities," in *Advances in Neural Information Processing Systems*, vol. 9. The MIT Press, 1997, p. 424.

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

[15] M. Pelikan and H. Mühlenbein, "The bivariate marginal distribution algorithm," in *Advances in Soft Computing - Engineering Design and Manufacturing.* London, UK: Springer-Verlag, 1999, pp. 521–535.

[16] 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.

[17] 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)*, vol. I. Orlando, FL: Morgan Kaufmann Publishers, San Fransisco, CA, 13-17 1999, pp. 525–532.

[18] 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.

[19] 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.

[20] M. Tsuji, M. Munetomo, and K. Akama, "Linkage identification by fitness difference clustering," *Evolutionary Computation*, vol. 14, no. 4, pp. 383–409, 2006.

[21] H. Kargupta, "The gene expression messy genetic algorithm," in *Proceedings of the 1996 International Conference on Evolutionary Computation (ICEC-96)*, 1996, pp. 814–819.

[22] 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)*, vol. 1. Orlando, Florida, USA: Morgan Kaufmann, 13-17 1999, pp. 433–440.

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

[24] J. R. Quinlan, "Induction of decision trees," in *Readings in knowledge acquisition and learning: automating the construction and improvement of expert systems.* San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993, pp. 349–361.

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

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

[27] D. Zwillinger and S. Kokoska, *CRC Standard Probability and Statistics Tables and Formulae.* FL, USA: CRC Press LLC, 2000.

[28] B. C. Arnold, N. Balakrishnan, and H. N. Nagaraja, *A First Course in Order Statistics.* NY, USA: John Wiley and Sons, Inc., 1993.