Linkage Identification by Perturbation and Decision Tree Induction

Chung-Yao Chuang
Ying-ping Chen

NCLab Report No. NCL-TR-2007008
April 2007

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/`

Linkage Identification by Perturbation and Decision Tree Induction

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

April 2, 2007

Abstract

The purpose of linkage identification in genetic and evolutionary algorithms is to detect the tightly linked variables of the fitness function. If such linkage information is not known a priori and can be obtained through computation, the crossover or recombination operator can accordingly mix discovered sub-solutions effectively without disrupting them. In this paper, we propose a linkage identification technique, called *inductive linkage identification* (ILI), employing perturbation with decision making. With the proposed scheme, the linkage information can obtained by constructing an ID3 decision tree to learn the mapping from population of solutions to their corresponding fitness differences caused by perturbation and inspecting the constructed decision tree for variables exhibiting strong interdependencies with one another. The numerical results show that the proposed technique can accomplish the identical linkage identification tasks with a lower number of function evaluations compared to similar methods proposed in the literature. Moreover, the proposed technique is shown able to handle both uniformly scaled and exponentially scaled problems.

1 Introduction

The encoding of solutions is of vital importance to the success of applying genetic and evolutionary algorithms. If the variables bearing strong relationship are encoded loosely on the representation, unless certain sophisticated mechanism is adopted for compensation, crossover tends to cause disruptions of promising sub-solutions, which are often referred to as building blocks (BBs), rather than properly mixing them. However, the knowledge of the problem to solve is not always sufficient to avoid this pitfall. For the situations with insufficient linkage information, some specifically designed techniques are needed to detect the structure of the fitness function and to identify the interdependent variables.

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

1. Evolving representations or operators;
2. Probabilistic modeling for promising solutions;
3. Perturbation methods.

The objective of the first class of techniques is to manipulate the representation of solutions during the search process such that members of the promising sub-solutions are less likely to be separated by crossover operators. Various reordering and mapping operators were proposed. In this line of research, the messy GA (mGA) [1] and its more efficient descendant—the fast messy GA (fmGA) [2]—identify linkage by exploiting building blocks. The problem of techniques in this category is that reordering operators are often too slow and lose the race against selection, resulting in the premature convergence to local optima. Another technique in this category, the linkage learning GA (LLGA) [3], employs a two-point crossover over circular representation of strings to maintain tight linkage. While LLGA works well on exponentially scaled problems, it is inefficient to handle uniformly scaled problems.

The approaches in the second category are often referred to as estimation of distribution algorithms (EDAs) [4]. 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) [5] and the compact genetic algorithm (cGA) [6], 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) [7], Baluja's dependency tree approach [8], and the bivariate marginal distribution algorithm (BMDA) [9], to modeling multivariate interactions, such as the extended compact genetic algorithm (ECGA) [10], the Bayesian optimization algorithm (BOA) [11], the factorized distribution algorithm (FDA) [12], and the learning version of FDA (LFDA) [13]. The model construction processes in these algorithms require no additional function evaluations. Thus, they can perform effectively especially for the situations in which the performance are bounded on fitness function evaluations. However, it is difficult for them to correctly model low salience (small fitness contribution) building blocks.

The methods in the third category examine the fitness differences by conducting perturbations on the variables to detect dependencies among them. For example, the gene expression messy GA (GEMGA) [14] employs a perturbation method to detect the sets of tightly linked variables represented by weight values assigned to each solution. GEMGA records fitness changes caused by perturbation of every variable for strings in the population and detects relations among variables according to the possibilities that the variables may construct the local optima. Linkage identification by nonlinearity check (LINC) [15] detects nonlinearity by using pairwise perturbations in order to identify the linkage information. It assumes that nonlinearity exists within variables to form a building block. If the fitness difference by simultaneous perturbations at a pair of variables is equal to the sum of fitness differences by perturbation at each variable in the pair, then these variables can be viewed as to reside within different and independent subproblems, and therefore, these variables can be optimized separately. Linkage information identified by LINC is represented as sets of variables. Each set contains tightly linked variables forming a building block and such a set is called a linkage set. The descendant of LINC, linkage identification by non-monotonicity detection (LIMD) [16], adopts non-monotonicity instead of nonlinearity and detects linkage by checking violations of the monotonicity conditions. Although perturbation methods require extra fitness function evaluations in addition to the running of GA, they have the advantage of being able to identify low salience building blocks. Heckendorn and Wright [17] generalized this category through a Walsh analysis.

An interesting algorithm combining an EDA and a perturbation method, called the dependency detection for distribution derived from fitness differences (D^5), was developed by Tsuji et al. [18]. 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 perturbations at that variable for the entire population, then cluster 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 the pre-defined problem complexity (the number of variables in a linkage set). These k variables are assumed to be tightly linked to form a linkage set. D^5 can detect dependencies of a class of functions that are difficult for EDAs (i.e. functions contain low salience building blocks) and requires less computational cost than other perturbation methods does. However, its major constraint is that it relies on an input parameter k which may or may not be available due to the limited information to 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 paper, we propose a new linkage identification technique based on perturbation, called *inductive linkage identification* (ILI). Similar to D^5, the population-wise perturbation approach is adopted, but different from D^5, instead of using clustering to obtain a biased sub-population, we use a supervised learning method well-established in the field of machine learning, ID3 [19], to construct a decision tree for the task of predicting the fitness difference after perturbation based on some parts of the solution. By inspecting the learned tree, we can obtain a set of variables exhibiting strong relationship with the perturbed variable. The contributions of the proposed approach include no problem complexity parameter (that is, k in D^5) is required, problems composed of different-sized building blocks can be appropriately handled, and a lower number of function evaluations for identifying linkage.

The rest of this paper is organized as follows. In section 2, the background of the linkage in GA and the decomposability of problems is briefly introduced. Section 3 gives a review of the ID3 decision tree learning algorithm. In section 4, we illustrate the proposed approach by using an example. Section 5 describes our algorithm in detail. Section 6 shows the empirical results. Finally, section 7 concludes this paper.

2 Linkage and Building Blocks

In this section, we briefly review some definitions and terminologies which will be used through out this paper. As stated in [20], “two variables in a problem are interdependent if the fitness contribution or optimal setting for one variable depends on the setting of the other variable,” and such relationships between variables is often referred to as linkage in GA. In order to obtain the full linkage information of a pair of variables, the fitness contribution or optimal setting of these two variables shall be examined on all possible settings of the other variables.

Although obtaining the full linkage information is computationally expensive, linkage should be estimated using a reasonable amount of efforts if the problem at hand is decomposable. According to the Schema theorem [21], short, low-order and highly fit substrings increase their share to be combined, and also stated in the building block hypothesis, GAs implicitly decompose a problem into sub-problems by processing building blocks. It is considered that combining small parts is important for GAs and consistent with human innovation [22]. These lead to a problem model called the additively decomposable function (ADF), which can written as a sum of low-order sub-functions.

Let a string $\mathbf{s}$ of length ℓ be described as a series of variables, $\mathbf{s} = s_1 s_2 \cdots s_\ell$. We assume that $\mathbf{s} = s_1 s_2 \cdots s_\ell$ is a permutation of the problem variables $\mathbf{x} = x_1 x_2 \cdots x_\ell$ to represent the used encoding scheme. The fitness of string $\mathbf{s}$ is the defined as

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

where m is the number of sub-functions, f_i is the i-th sub-function, and $\mathbf{s}_{v_i}$ is the substring to f_i. The v_i is a vector specifying the substring $\mathbf{s}_{v_i}$. For example, if $v_i = (1, 2, 4, 8)$, $\mathbf{s}_{v_i} = s_1 s_2 s_4 s_8$. If f_i is a simple sum of more elemental sub-functions, f_i can be replaced by those sub-functions. Each of the sub-function f_i can be considered as a nonlinear function.

By eliminating the ordering property of v_i , we can obtain a set V_i containing the elements of v_i. The variables from the same set of V_i should be interdependent because f_i is nonlinear. Thus, we refer to the set V_i as a linkage set. A related term, building blocks (BBs), is referred to as the candidate solutions of some sub-function f_i. In this paper, only a subclass of the ADFs is considered. We concentrate on non-overlapping sub-functions. That is, $V_i \cap V_j = \emptyset$ if $i \neq j$. The strings is composed of binary variables.

3 Decision Tree Learning: ID3

Decision tree learning is one of the most widely used and practical methods for inductive inference. It has been successfully applied to a broad range of tasks from learning to diagnose medical cases to learning to assess credit risks of loan applicants. Decision tree learning approximates discrete-valued target functions, in which the learned function is represented by a decision tree.

In this paper, the ID3 decision tree learning algorithm [19] is used and we consider only its ability in classification problems. In 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, the list of attribute values is the population of solution strings, and the target value is the fitness difference after perturbation.

In its most basic form, ID3 constructs the decision tree top-down 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 the appropriate descendant node. The entire process is then repeated using the training instances associated with each descendant node to select the best attribute to test at that node.

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

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

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

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

$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Val(A)} \frac{|S_v|}{|S|} Entropy(S_v) \ ,$$

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

4 Exemplary Illustration

Before describing the proposed linkage identification technique in detail, in this section, we first illustrate the idea behind the algorithm by using the following example. Consider a trap function of size k[1]:

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

[1]The proposed algorithm does not require this parameter of problem complexity, but for explanation, we use the k-traps as sub-problems.

$s_1s_2\cdots s_8$	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.

$s_1s_2\cdots s_8$	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.

where u is the number of ones in the string $s_1s_2\cdots s_k$. Suppose that we are dealing with an eight-bit problem

$$f(s_1s_2\cdots s_8) = f_{trap_5}(s_1s_2s_3s_4s_5) + f_{trap_3}(s_6s_7s_8) ,$$

where $s_1s_2\cdots s_8$ 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 ($0 \to 1$ or $1 \to 0$) for all strings in the population in order to detect the linkage set in which the variables are related to s_1 (that is, V_1). The third column of Table 1(a) records the fitness differences, df_1, after the 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 $s_1s_2\cdots s_8$ is an attribute to the instances, and the target values are

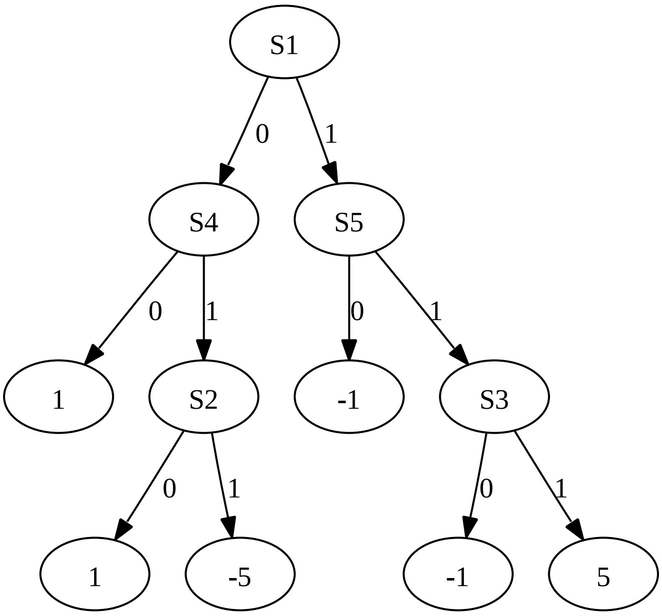

Figure 1: An ID3 decision tree constructed according to Table 1(a).

the fitness differences df_1. By having this setup, we can obtain an ID3 decision tree as shown in Figure 1. By gathering all decision variables of the non-leaf nodes, we can identify as a group s_1, s_2, s_3, s_4, and s_5 which are the variables corresponding to linkage set V_1. As a consequence, the linkage set V_1 is correctly identified.

Readers might think this result 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, because during the construction of the decision tree, the ID3 algorithm selects variables that show strong relationship to the target values, i.e., the fitness differences after perturbation, the variables belonging to the same sub-function as the perturbed variable, s_1, tend to be selected under 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:

$$\begin{aligned} df_1(\mathbf{s}) &= f(s_1 s_2 \cdots s_8) - f(\overline{s_1} s_2 \cdots s_8) \\ &= f_{trap_5}(s_1 s_2 s_3 s_4 s_5) + f_{trap_3}(s_6 s_7 s_8) - f_{trap_5}(\overline{s_1} s_2 s_3 s_4 s_5) - f_{trap_3}(s_6 s_7 s_8) \\ &= f_{trap_5}(s_1 s_2 s_3 s_4 s_5) - f_{trap_5}(\overline{s_1} s_2 s_3 s_4 s_5)\ . \end{aligned} \tag{1}$$

As shown in Equation (1), the fitness difference df_1 is independent of the variables s_6, s_7, and s_8. df_1 depends only on s_1, s_2, …, s_5. Therefore, for a large enough population showing strong statistical evidences, the independent variables will not be chosen as decision variables in the decision tree. On the other hand, because f_{trap_5} is a function with nonlinearity, all five variables tends to be identified given a large enough population which contains nonlinear points of f_{trap5}.

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 is shown in Table 2. Employing the 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

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

Table 2: Population of strings.

example illustrates that the proposed algorithm can handle problems composed of different-sized sub-problems.

5 Inductive Linkage Identification

In this section, the idea demonstrated in the previous section is formalized as an algorithm, which is called *inductive linkage identification* (ILI) and presented in Algorithm 1. ILI consists mainly the following three steps:

1. Calculate the fitness differences by perturbations;
2. Construct an ID3 decision tree;
3. Examine the decision tree to obtain a linkage set.

The three steps repeat until all the variables are classified into a linkage set.

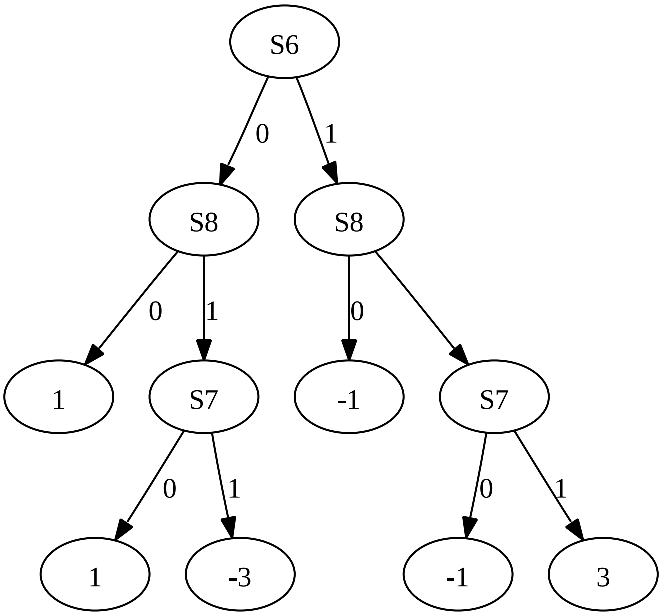

Figure 2: An ID3 decision tree constructed according to Table 2.

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 is constructed according to the fitness differences after perturbations; (3) by inspecting the tree, a linkage set of the variables used in the tree can be collected.

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 ← {1, ..., ℓ}
    m ← 0
    while V ≠ ∅ do
        m ← m + 1
        Select v in V at random.
        V_m ← {v}
        V ← V − {v}
        for each string s^i = s^i_1 s^i_2 ··· s^i_ℓ in P do
            Perturb s^i_v.
            df^i ← fitness difference after perturbation.
        end for
        Construct an ID3 decision tree using (P, df).
        for each decision variable s_j in tree do
            V_m ← V_m ∪ {j}
            V ← V − {j}
        end for
    end while
    return the linkage sets V_1, V_2, ···, V_m
end procedure
```

As clearly shown in Algorithm 1, the number of fitness function evaluations required to accomplish the task of linkage identification is proportional to the number of the linkage sets of the problem. Suppose that we are dealing with an ADF f in which the length of solution string 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, LINC needs $\mathcal{O}(\ell^2) = \mathcal{O}(k^2m^2)$ 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 evaluations growing linearly with the problem size, but ILI needs fewer evaluations by a factor of the building-block size k. The numerical results presented in the next section verify the theoretical computation.

6 Numerical Experiments

The empirical results are presented in this section. The experiments are designed to show the behavior of the proposed technique, ILI, on binary ADFs with non-overlapping subfunctions. For the considered problems, the scalability of the proposed algorithm is investigated and compared to LINC and D^5. Furthermore, the numerical results on uniformly scaled functions and exponentially scaled functions are also presented to examine the flexibility of ILI.

6.1 Uniformly Scaled Functions

The section describes the experimental settings and results of the proposed algorithm on uniformly scaled functions. The experiment 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 population sizes are set empirically to obtain all linkage sets correctly in 10 consecutive and independent runs. The results are shown in Figure 3. The results of ILI are compared to that of LINC and D^5 [18]. The number of function evaluations needed by ILI grows linearly with the problem size and is much lower than that needed by LINC. It is also lower by a factor of the building-block size than that needed by D^5. The results verify the theoretical computation presented in the previous section.

6.2 Exponentially Scaled Functions

In this section, the results for exponentially scaled functions are presented. As in the experiment on uniformly scaled functions, the $trap_5$ function is used as a subfunction to compose more complicated functions:

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

where m ranges from 10 to 50. That is, the problem size ranges from 50 bits to 250 bits.

The population sizes are set empirically to obtain all linkage sets correctly in 10 consecutive and independent runs. The results are shown in Figure 4. The results of ILI 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 the same size of problems. It indicates that ILI is independent of different building block scalings.

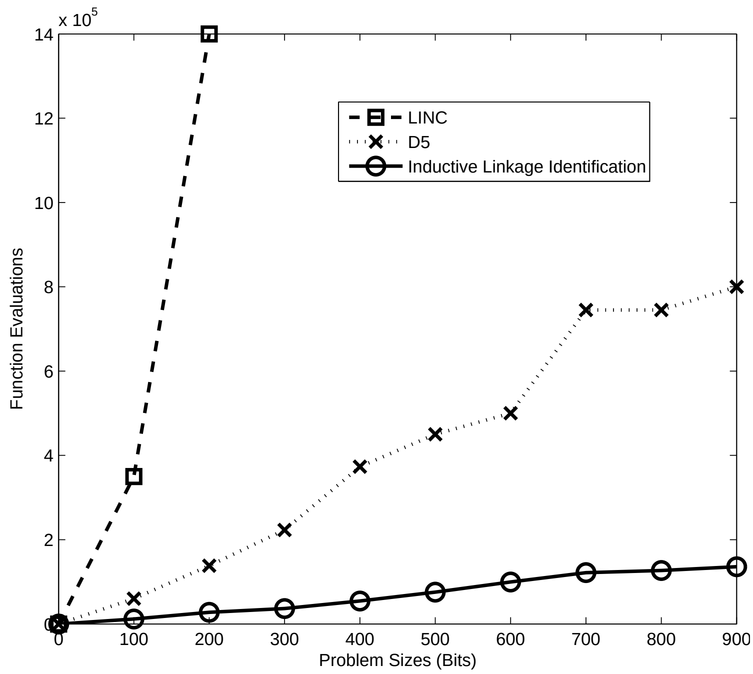

Figure 3: Numerical results of inductive linkage identification on uniformly scaled functions. The results are compared to that of LINC and D^5 [18]. The number of function evaluations needed by ILI grows linearly with the problem size.

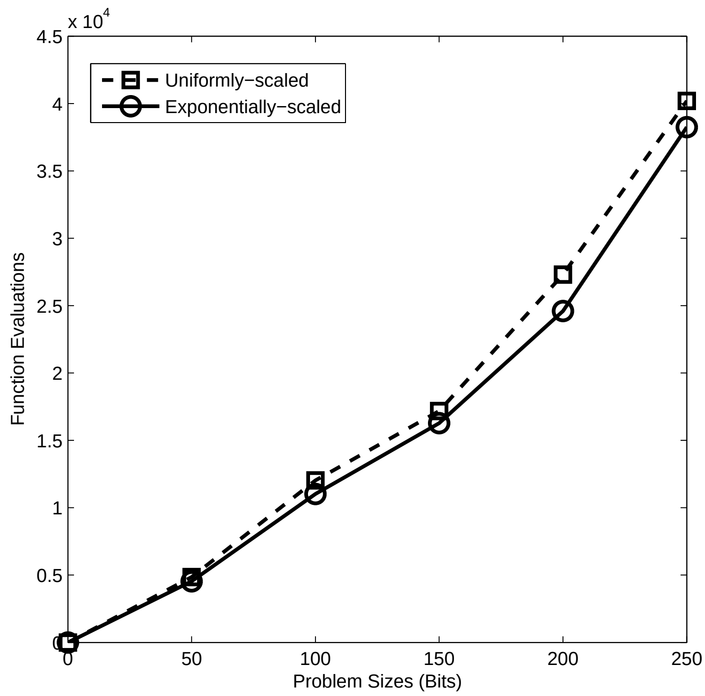

Figure 4: Numerical results of inductive linkage identification on exponentially scaled functions. The results compared to that of ILI on uniformly scaled functions.

7 Summary and Conclusions

In this paper, we proposed an algorithm, called *inductive linkage identification* (ILI), to identify linkage for a class of problems. The algorithm utilized a supervised learning model, ID3, as the task-force to estimate linkage sets. After obtaining the fitness differences by conducting perturbation, an ID3 decision tree was constructed according to the gained information, and a linkage group was identified based on the created tree. The performance of the proposed algorithm was discussed and compared to that of other techniques. The results demonstrated that ILI was able to identifying linkage groups at a lower computational cost.

ILI improves the previous methods, including LINC and D^5, in two aspects. First, the number of function evaluations for identifying linkage sets is reduced. Second, the algorithm requires no input parameter regarding the information of the problem structure or problem complexity, such as the building block size k. Moreover, the function evaluations required by the proposed algorithm is proportional to the number of linkage sets in the problem.

The proposed technique can be used in two possible areas. First, it can serve as a preprocessing step of a running GA. By obtaining the information of linkage sets, the crossover operator can be designed to perform effective mixing of sub-solutions. Second, it can be used as a tool for understanding the structure of totally unknown or partially understood problems.

Acknowledgments

The work was partially sponsored by the National Science Council of Taiwan under grants NSC-95-2221-E-009-092 and NSC-95-2627-B-009-001 as well as by the MOE ATU Program. The authors are grateful to the National Center for High-performance Computing for computer time and facilities.

References

[1] 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.

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

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

[4] 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.

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

[6] G. R. Harik, F. G. Lobo, and D. E. Goldberg, "The compact genetic algorithm," *IEEE-EC*, vol. 3, no. 4, p. 287, November 1999.

[7] 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, p. 424.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

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

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

[15] 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.

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

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

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

[19] 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.

[20] E. D. de Jong, R. Watson, and D. Thierens, "On the complexity of hierarchical problem solving," in *GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation.* New York, NY, USA: ACM Press, 2005, pp. 1201–1208.

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

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