Recognizing Problem Decomposition with Inductive Linkage Identification: Population Requirement vs. Subproblem Complexity

Chung-Yao Chuang
Ying-ping Chen

NCLab Report No. NCL-TR-2008003
July 2008

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

Recognizing Problem Decomposition with Inductive Linkage Identification: Population Requirement vs. Subproblem Complexity

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

July 2008

Abstract

In genetic and evolutionary algorithms, the goal of linkage identification is to detect strong relationships among the decision variables of the objective function. If such relations can be identified, the crossover operator can accordingly mix and recombine the identified sub-solutions without destroying them. In our previous study, we have proposed a linkage identification technique, called *inductive linkage identification* (ILI), which integrates the mechanisms of perturbation and decision tree induction. With the proposed technique, linkage information of the objective function can be acquired via constructing an ID3 decision tree to model the mapping from solution strings to their corresponding fitness changes caused by perturbations and then inspecting the constructed decision tree for the decision variables exhibiting strong interdependencies with one another. In this study, we observe the behavior of ILI on decomposable problems of different subproblem complexities and make an attempt to understand the population requirement of the proposed linkage identification technique. The experimental results demonstrate that the population size required by ILI to correctly learn linkage grows sub-linearly with the problem size while grows exponentially with the complexity of constituting subproblems.

1 Introduction

The encoding scheme adopted in genetic and evolutionary algorithms is crucial to successful applications. If the decision variables bearing strong relationship are loosely coded in the representation, unless carefully designed mechanisms are adopted for compensation, problem-independent crossover operators tend to disrupt promising sub-solutions, which are often referred to as building blocks (BBs), rather than to properly mix them. However, problem-specific domain knowledge to avoid this pitfall is not always available. For the situations with insufficient linkage information, specifically devised methods are needed to detect the problem structure and to identify the dependencies among variables.

To tackle the building block disruption problem, a variety of methods have been proposed in the literature. These methods can be roughly classified into three categories:

1. Evolving representations and/or operators;
2. Building probabilistic models on promising solutions;
3. Perturbing variables and analyzing fitness changes.

The goal of the techniques in the first class is to adapt the representation during the search process such that the promising sub-solutions are less likely to be separated by crossover. In this line of research, the messy GA (mGA) [1] and its more efficient descendant—the fast messy GA (fmGA) [2]—detect linkage by exploiting building blocks. One of the major issue of these techniques is that the reordering mechanism often acts too slow and loses the race against selection, resulting in premature convergence. Another technique in this category, the linkage learning GA (LLGA) [3], employs a two-point crossover over a circular representation of strings to maintain tight linkage. While LLGA works well on exponentially scaled problems, it is inefficient in handling uniformly scaled problems [3] [4].

The approaches in the second class are often referred to as estimation of distribution algorithms (EDAs) [5]. These methods build probabilistic models on selected individuals and sample the built model to generate offspring individuals. Early EDAs, such as the population-based incremental learning (PBIL) [6] and the compact genetic algorithm (cGA) [7], assume no interaction between variables. Subsequent studies start from modeling pairwise interactions, such as mutual-information-maximizing input clustering (MIMIC) [8], Baluja's dependency tree approach [9], and the bivariate marginal distribution algorithm (BMDA) [10], to capturing multivariate interactions, such as the extended compact genetic algorithm (ECGA) [11], the Bayesian optimization algorithm (BOA) [12], the factorized distribution algorithm (FDA) [13], and the learning version of FDA (LFDA) [14]. The model building process usually requires no additional function evaluations. Thus, these methods can perform effectively especially for the situations in which the performance are bounded by fitness function evaluations. However, it is difficult for them to correctly model low salience building blocks [15].

In the third category, the methods analyze the fitness differences caused by perturbing the variables to detect dependencies. For example, the gene expression messy GA (GEMGA) [16] detects the sets of tightly linked variables represented by weights assigned to each solution with perturbation. Linkage identification by nonlinearity check (LINC) [17] detects nonlinearity by using pairwise perturbations to capture the linkage information. The descendant of LINC, linkage identification by non-monotonicity detection (LIMD) [18], uses non-monotonicity instead of nonlinearity and detects linkage by checking the monotonicity condition violations. Combining the ideas of EDAs and perturbation methods, dependency detection for distribution derived from fitness differences (D^5) [15] detects the variable dependencies by estimating the distributions of strings clustered with fitness differences. Although perturbation methods require extra function evaluations, they have the advantage of being able to identify low salience building blocks.

In our previous work, a new linkage identification technique based on perturbation, called *inductive linkage identification* (ILI), was proposed [19], in which ID3 [20], a supervised learning method well-established in the field of machine learning, was adopted to construct a decision tree to predict the fitness changes. The advantages of ILI include requiring fewer function evaluations than other perturbation methods and no need for setting the problem complexity parameter. In this paper, we make further attempts to inspect the behavior and the characteristics of ILI. In particular, we perform controlled experiments using problems with different subproblem complexities and observe the growth of required population sizes for correctly identifying the underlying problem structure.

The rest of this paper is organized as follows. In section 2, the background of the linkage in GAs and the decomposability of problems is briefly introduced. Section 3 describes ILI in detail with an illustrative example. Section 4 shows the empirical results, followed by the summary and conclusions in section 5.

2 Linkage and Building Blocks

In this section, we briefly review the definitions used in the remainder of this paper. As stated in [21], "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 relationship among variables is often referred to as *linkage*. In order to obtain the complete linkage information of one variable pair, the fitness contribution or optimal setting of the two variables shall be examined on all possible value combinations of the other variables. Although in general, obtaining the complete linkage information is computationally expensive, linkage should be estimated with a reasonable amount of efforts if the problem is decomposable.

According to the Schema theorem [22], short, low-order, and highly fit sub-solutions increase their market share to be combined. Also stated in the building block hypothesis, genetic algorithms implicitly decompose a problem into sub-problems by processing building blocks. It is considered that combining sub-solutions is essential for genetic algorithms as well as consistent with human innovation [23]. Such a condition leads to the proposal of a problem model, called the *additively decomposable function* (ADF), written as a sum of low-order sub-functions.

Let a string $\mathbf{s}$ of length ℓ represent a solution string, where $\mathbf{s} = s_1 s_2 \cdots s_\ell$. We assume that $\mathbf{s} = s_1 s_2 \cdots s_\ell$ is a permutation of the decision variables $\mathbf{x} = x_1 x_2 \cdots x_\ell$ determined by the encoding scheme in use. The fitness of solution string $\mathbf{s}$ is 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 for f_i. Each 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 also a sum of other sub-functions, it can be replaced by those sub-functions. Thus, each sub-function f_i can be considered as a non-linear function and we refer the number of variables to a particular sub-function as the complexity of that sub-function.

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

3 Inductive Linkage Identification

In this section, we will introduce the idea of *inductive linkage identification* (ILI), proposed in our previous study [19], with an illustrative example. Following that, the detailed description of ILI will be provided.

Let's consider a trap function [24, 25] of size k:

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

where u is the number of ones in the string $s_1 s_2 \cdots s_k$. Figure 1 visualizes such a function.

Now, suppose that we are dealing with an eight-bit problem

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

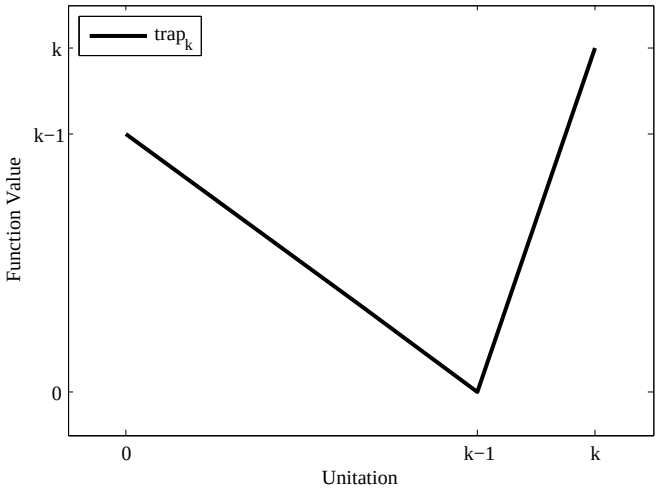


Figure 1: Trap function of size k.

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

In the beginning, a population of individuals is randomly generated as listed in Table 1(a). The first column lists the individuals, and the second column lists the corresponding fitness values. As an arbitrary choice, we perturb variable s_1 ($0 \rightarrow 1$ or $1 \rightarrow 0$) for all the individuals in order to identify the linkage set in which the variables are related to s_1 (that is, V_1). The fitness differences caused by perturbations at variable s_1, df_1, are recorded in the third column of Table 1(a).

Then, an ID3 decision tree [20] is constructed by using the population of individuals as the training instances. Each decision variable in $s_1 s_2 \cdots s_8$ is an attribute of the instances, and the target values are the fitness differences df_1. By setting up this configuration, an ID3 decision tree shown in Figure 2 can be obtained. Gathering all the variables on the non-leaf nodes, we can identify a linkage group: s_1, s_2, and s_3 which are the decision variables corresponding to linkage set V_1. As a consequence, linkage set V_1 is correctly identified.

To further illustrate this example, we may consider the rearranged population listed in Table 1(b). In Table 1(b), the individuals from different sections bear different patterns. For example, s_1 and s_3 of the strings from the first section are all 0's. In the fourth section, values of s_1 are 1's, and values of s_3 are 0's. Such an observation can be extended to other sections as well. These patterns are corresponding to the paths from the leaf nodes to the root node of the tree in Figure 2. To put it in another way, because during tree construction, the ID3 algorithm selects the variables showing strong relationship to the target values, i.e., the fitness differences caused by perturbations, the variables belonging to the same sub-function as the perturbed variable, s_1, tend to be selected.

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

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

As shown in Equation (1), fitness difference df_1 is independent of variables s_4, s_5, $\cdots$, s_8. df_1 depends on only s_1, s_2, and s_3. Therefore, for a sufficiently large population showing some

$s_1 s_2 \cdots s_8$	f	df_1
$\bar{0}11$ 01111	0	-3
$\bar{0}01$ 00011	3	1
$\bar{0}00$ 00100	5	1
$\bar{1}11$ 01001	5	3
$\bar{0}00$ 11111	7	1
$\bar{1}01$ 01101	1	-1
$\bar{0}11$ 00110	2	-3
$\bar{1}10$ 01101	1	-1
$\bar{0}11$ 00001	3	-3
$\bar{1}11$ 10100	5	3
$\bar{1}01$ 11110	0	-1
$\bar{1}10$ 11111	5	-1
$\bar{0}10$ 11011	1	1
$\bar{0}10$ 01000	4	1
$\bar{0}10$ 00100	4	1
$\bar{0}00$ 00001	5	1
$\bar{0}10$ 01100	3	1
$\bar{1}01$ 10000	3	-1
$\bar{1}00$ 00000	5	-1
$\bar{1}10$ 11011	0	-1
$\bar{0}01$ 00011	3	1
$\bar{0}10$ 00111	2	1
$\bar{1}00$ 00100	4	-1
$\bar{0}00$ 10110	3	1
$\bar{0}00$ 11100	3	1
$\bar{1}11$ 01111	3	3
$\bar{0}10$ 10100	3	1
$\bar{0}01$ 10100	3	1
$\bar{0}01$ 01000	4	1
$\bar{1}10$ 01111	0	-1

(a) Original population.

$s_1 s_2 \cdots s_8$	f	df_1
$\bar{0}00$ 11111	7	1
$\bar{0}00$ 00100	5	1
$\bar{0}00$ 00001	5	1
$\bar{0}10$ 01000	4	1
$\bar{0}10$ 00100	4	1
$\bar{0}00$ 10110	3	1
$\bar{0}00$ 11100	3	1
$\bar{0}10$ 01100	3	1
$\bar{0}10$ 10100	3	1
$\bar{0}10$ 00111	2	1
$\bar{0}10$ 11011	1	1
$\bar{0}01$ 01000	4	1
$\bar{0}01$ 00011	3	1
$\bar{0}01$ 00011	3	1
$\bar{0}01$ 10100	3	1
$\bar{0}11$ 00001	3	-3
$\bar{0}11$ 00110	2	-3
$\bar{0}11$ 01111	0	-3
$\bar{1}00$ 00000	5	-1
$\bar{1}10$ 11111	5	-1
$\bar{1}00$ 00100	4	-1
$\bar{1}10$ 01101	1	-1
$\bar{1}10$ 01111	0	-1
$\bar{1}10$ 11011	0	-1
$\bar{1}01$ 10000	3	-1
$\bar{1}01$ 01101	1	-1
$\bar{1}01$ 11110	0	-1
$\bar{1}11$ 01001	5	3
$\bar{1}11$ 10100	5	3
$\bar{1}11$ 01111	3	3

(b) Rearranged population.

Table 1: Population of solution strings.

significant statistical evidence, the independent variables will not be chosen as the decision attributes in the decision tree. On the other hand, because f_{trap_3} is a nonlinear function, all the three variables tend to be identified given a sufficiently large population containing nonlinear points of f_{trap3}.

For the rest part of this example, since V_1 is already identified, we arbitrarily choose another variable which is not in V_1, say, s_4. The fitness differences after perturbations at variable s_4 are shown in Table 2(a). With the same procedure, an ID3 decision tree is constructed and presented in Figure 3. By inspecting the tree, we obtain the related variables s_4, s_5, $\cdots$, s_8 which form linkage set V_2 of size 5. The example also illustrates that ILI can handle problems composed of subproblems of various complexities (i.e., sub-function sizes).

The idea of ILI illustrated previously can be formalized as pseudo-code and presented in Algorithm 1. ILI mainly consists of the following three steps:

1. Calculate the fitness differences by perturbations;

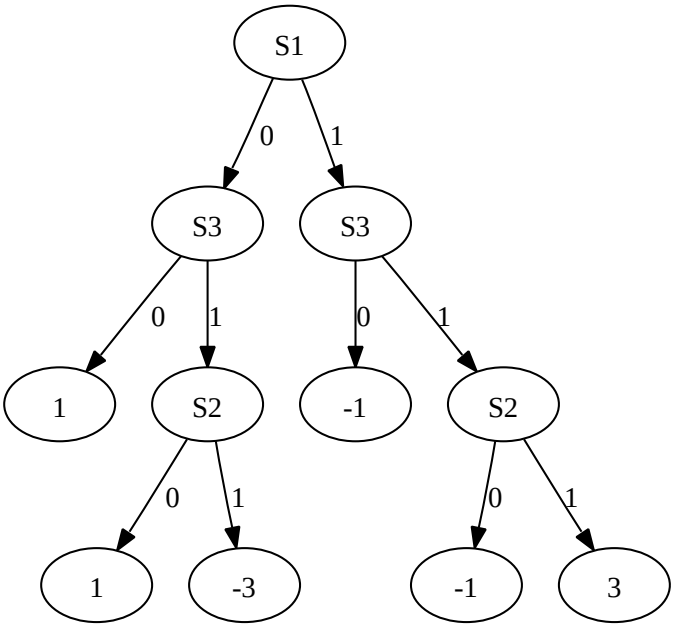


Figure 2: An ID3 decision tree constructed according to Table 1.

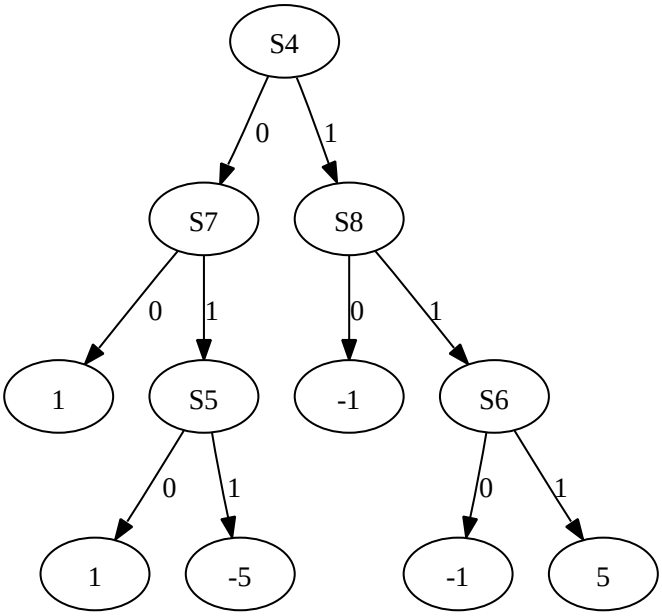


Figure 3: An ID3 decision tree constructed according to Table 2.

2. Construct an ID3 tree rooted at the perturbed variable;

3. Inspect the decision tree to obtain a linkage set.

The three steps repeat until all the variables of the objective function are included in their corresponding linkage sets. In detail, ILI starts at initializing a population of individuals. 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 specified as root is constructed according to the fitness differences caused by perturbations; (3)

$s_1 s_2 \cdots s_8$	f	df_4
011 $\overline{0}$1111	0	-5
001 $\overline{0}$0011	3	1
000 $\overline{0}$0100	5	1
111 $\overline{0}$1001	5	1
000 $\overline{1}$1111	7	5
101 $\overline{0}$1101	1	1
011 $\overline{0}$0110	2	1
110 $\overline{0}$1101	1	1
011 $\overline{0}$0001	3	1
111 $\overline{1}$0100	5	-1
101 $\overline{1}$1110	0	-1
110 $\overline{1}$1111	5	5
010 $\overline{1}$1011	1	-1
010 $\overline{0}$1000	4	1
010 $\overline{0}$0100	4	1
000 $\overline{0}$0001	5	1
010 $\overline{0}$1100	3	1
101 $\overline{1}$0000	3	-1
100 $\overline{0}$0000	5	1
110 $\overline{1}$1011	0	-1
001 $\overline{0}$0011	3	1
010 $\overline{0}$0111	2	1
100 $\overline{0}$0100	4	1
000 $\overline{1}$0110	3	-1
000 $\overline{1}$1100	3	-1
111 $\overline{0}$1111	3	-5
010 $\overline{1}$0100	3	-1
001 $\overline{1}$0100	3	-1
001 $\overline{0}$1000	4	1
110 $\overline{0}$1111	0	-5

(a) Original population.

$s_1 s_2 \cdots s_8$	f	df_4
100 $\overline{0}$0000	5	1
000 $\overline{0}$0001	5	1
000 $\overline{0}$0100	5	1
111 $\overline{0}$1001	5	1
010 $\overline{0}$0100	4	1
100 $\overline{0}$0100	4	1
010 $\overline{0}$1000	4	1
001 $\overline{0}$1000	4	1
011 $\overline{0}$0001	3	1
010 $\overline{0}$1100	3	1
101 $\overline{0}$1101	1	1
110 $\overline{0}$1101	1	1
001 $\overline{0}$0011	3	1
001 $\overline{0}$0011	3	1
011 $\overline{0}$0110	2	1
010 $\overline{0}$0111	2	1
111 $\overline{0}$1111	3	-5
011 $\overline{0}$1111	0	-5
110 $\overline{0}$1111	0	-5
111 $\overline{1}$0100	5	-1
101 $\overline{1}$0000	3	-1
010 $\overline{1}$0100	3	-1
001 $\overline{1}$0100	3	-1
000 $\overline{1}$0110	3	-1
000 $\overline{1}$1100	3	-1
101 $\overline{1}$1110	0	-1
010 $\overline{1}$1011	1	-1
110 $\overline{1}$1011	0	-1
000 $\overline{1}$1111	7	5
110 $\overline{1}$1111	5	5

(b) Rearranged population.

Table 2: Population of solution strings.

by inspecting the constructed tree, the variables used in the decision tree are collected and considered as a linkage set.

As clearly 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 problem. 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 sub-functions forming f, and k is the size of each sub-function. ILI needs $\mathcal{O}(m)$ function evaluations. As a consequence, ILI needs a number of function evaluations growing linearly with the problem size if the population is sufficiently large. However, it is still unknown that how large the population will be needed by ILI to correctly identify the linkage groups of a given problem. In this study, we aim to empirically understand the relationship between the population requirement and the complexity of subproblems.

Algorithm 1 Inductive Linkage Identification

procedure IDENTIFYLINKAGE(f, ℓ, n)
  Initialize a population P with n strings 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 $s_v^{(i)}$.
      $df^{(i)} \leftarrow$ calculate the fitness difference.
    end for
    Construct an ID3 tree using (P, df) with v as root.
    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

4 Experiments and Results

Experiment settings and empirical results are presented in this section. The experiments are designed to reveal the population requirement for ILI to work correctly on problems composed 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. In the experiments, k ranges from 3 to 6. The total problem sizes, ℓ, are 60, 120, 180, $\cdots$, 600 bits, and m is calculated by $m = \ell/k$.

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 a population size is enough 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 bound and lower bound. If the identification is successful, i.e., ILI can correctly identify all the linkage sets in 30 consecutive and independent runs, the current middle value will be the next upper bound. If the identification is unsuccessful, the current middle value will be the next lower bound. The procedure repeats until the difference between the upper bound and the lower bound is less than or equal to 2.

The experimental results are shown in Figures 4 and 5. Figure 4 shows that the population size required by ILI grows sub-linearly with the problem size if the subproblem complexity is fixed. The results indicate that the population requirement of ILI is relatively insensitive to the

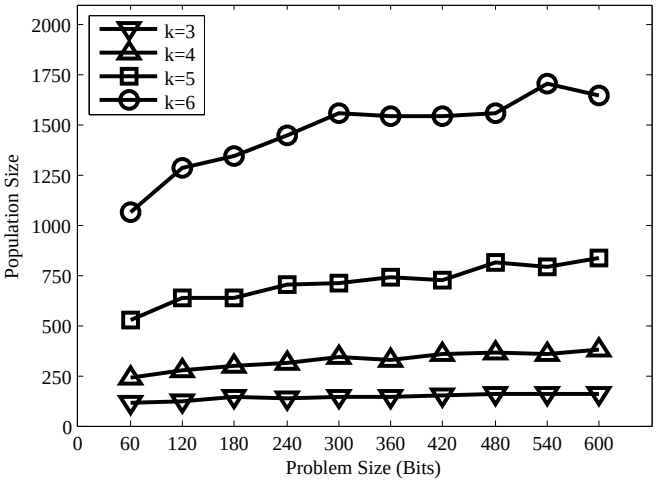


Figure 4: The population size required by inductive linkage identification (ILI) to correctly identify all the linkage sets in 30 consecutive, independent runs for problems of different total sizes. For a fixed subproblem complexity, i.e., k, the population size grows sub-linearly with the problem size.

total problem size and also grows sub-linearly with the number of subproblems. However, the straight lines for $\ell = 120$, 360, and 600 in Figure 5(b), of which the y-axis is log-scaled, indicate that for a fixed problem size, the population size required by ILI grows exponentially with the complexity of subproblems.

5 Conclusions and Future Works

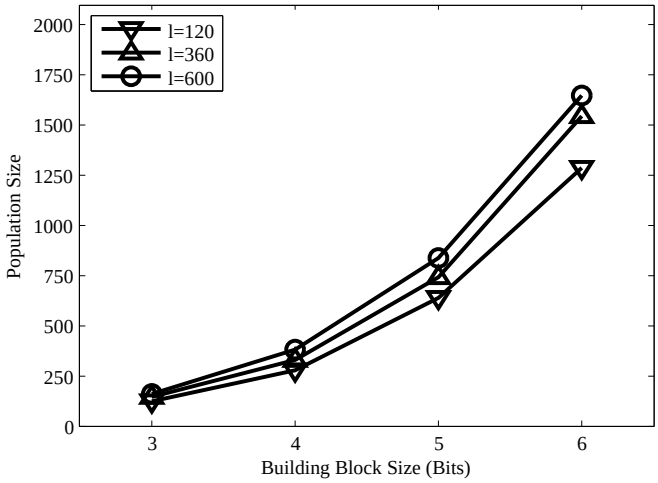
In our previous work, we proposed *inductive linkage identification* (ILI) to identify linkage sets of problems. ILI utilizes the technique of ID3 to estimate linkage sets. It starts with perturbing a randomly chosen variable for all the individuals and then records the fitness differences caused by perturbations. Based on the fitness differences, an ID3 decision tree is constructed to identify the linkage group to which the perturbed variable belongs.

In this study, we concentrated on understanding the behavior and characteristics of ILI. Particularly, we observed the growth of the population size required by ILI in order to correctly identify the decomposition of problems of various subproblem complexities. A bisection method was adopted to determine the minimum population size with which ILI could successfully accomplish the linkage identification task in 30 consecutive and independent runs. The numerical results demonstrated that the required population size grows sub-linearly with the number of subproblems while it grows exponentially with the size of subproblems.

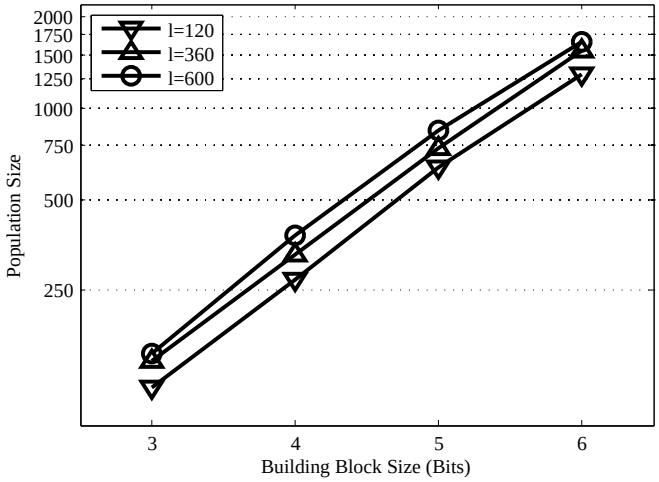
In order to provide theoretical explanations for the reason why the population size required by ILI grows exponentially with the size of subproblems, a population sizing model is being developed and may reveal certain intrinsic properties of ILI or even linkage learning mechanisms. As for the practical use, the performance and behavior of ILI on problems composed of different types of subproblems other than traps will be examined and explored as future work.

Acknowledgments

The work was supported in part by the National Science Council of Taiwan under Grant NSC-96-2221-E-009-196 and Grant NSC-96-2627-B-009-001 as well as by the ATU Plan (Aiming for



(a) Population size vs. Subproblem complexity.



(b) Population size vs. Subproblem complexity (semilog-scale).

Figure 5: The population size required by inductive linkage identification (ILI) to correctly identify all the linkage sets in 30 consecutive, independent runs for problems composed of subproblems of various complexities. For a fixed problem size, i.e., ℓ, the population size grows exponentially with the subproblem complexity.

the Top University and Elite Research Center Development Plan) of the National Chiao Tung University and Ministry of Education, Taiwan. 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] Y.-p. Chen, *Extending the scalability of linkage learning genetic algorithms: Theory and practice*, ser. Studies in Fuzziness and Soft Computing. Springer, 2006, vol. 190, ISBN: 3-540-28459-1.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

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

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

[17] 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.

[18] 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.

[19] C.-Y. Chuang and Y.-p. Chen, "Linkage identification by perturbation and decision tree induction," in *Proceedings of 2007 IEEE Congress on Evolutionary Computation (CEC 2007)*, 2007, pp. 357–363.

[20] 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.

[21] E. D. de Jong, R. Watson, and D. Thierens, "On the complexity of hierarchical problem solving," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2005).* New York, NY, USA: ACM Press, 2005, pp. 1201–1208.

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

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

[24] D. H. Ackley, *A connectionist machine for genetic hillclimbing.* Norwell, MA, USA: Kluwer Academic Publishers, 1987.

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