International Journal of Systems Science
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/tsys20

Inductive linkage identification on building blocks of different sizes and types

Ying-ping Chen [a], Chung-Yao Chuang [a] & Yuan-Wei Huang [a]
[a] Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan
Published online: 11 Apr 2011.

To cite this article: Ying-ping Chen, Chung-Yao Chuang & Yuan-Wei Huang (2012) Inductive linkage identification on building blocks of different sizes and types, International Journal of Systems Science, 43:12, 2202-2213, DOI: 10.1080/00207721.2011.566639

To link to this article: http://dx.doi.org/10.1080/00207721.2011.566639

International Journal of Systems Science
Vol. 43, No. 12, December 2012, 2202–2213

Taylor & Francis
Taylor & Francis Group

Inductive linkage identification on building blocks of different sizes and types

Ying-ping Chen*, Chung-Yao Chuang and Yuan-Wei Huang

Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan

(*Received 24 March 2010; final version received 17 February 2011*)

The goal of linkage identification is to obtain the dependencies among decision variables. Such information or knowledge can be applied to design crossover operators and/or the encoding schemes in genetic and evolutionary methods. Thus, promising sub-solutions to the problem will be disrupted less likely, and successful convergence may be achieved more likely. To obtain linkage information, a linkage identification technique, called *Inductive Linkage Identification* (ILI), was proposed recently. ILI was established upon the mechanism of perturbation and the idea of decision tree learning. By constructing a decision tree according to decision variables and fitness difference values, the interdependent variables will be determined by the adopted decision tree learning algorithm. In this article, we aim to acquire a better understanding on the characteristics of ILI, especially its behaviour under problems composed of different-sized and different-type building blocks (BBs) which are not overlapped. Experiments showed that ILI can efficiently handle BBs of different sizes and is insensitive to BB types. Our experimental observations indicate the flexibility and the applicability of ILI on various elementary BB types that are commonly adopted in related experiments.

Keywords: inductive linkage identification; ILI; linkage learning; BBs; genetic algorithms; evolutionary computation

1. Introduction

Previous studies (Goldberg, Korb, and Deb 1989; Harik 1997) on genetic algorithms (GAs), which are widely utilised to handle control and engineering problems (Wang 2009; Li and Li 2010; Gladwin, Stewart, and Stewart 2011), have shown that the encoding scheme of solutions is one of the key factors to the success of GAs by demonstrating that simple GAs fail to handle problems of which the solutions are represented with loose encodings while genetic algorithms capable of learning linkage succeed. If strongly related variables, which are usually referred to as building blocks (BBs), are arranged loosely with the adopted representation, they are likely to be disrupted by crossover operations. Such a condition contributes to the divergence of population, instead of the convergence towards optimal solutions. Although encoding strongly related variables tightly or making crossover operators aware of such relationships could mitigate the problem and improve the GA performance (Stonedahl, Rand, and Wilensky 2008), both measures require the foreknowledge of the target problem, which is often not the case in which evolutionary algorithms are adopted.

In order to overcome the BB disruption problem, a variety of techniques have been proposed and developed in the past two decades and can be roughly classified into three categories (Munetomo and Goldberg 1998; Chen, Yu, Sastry, and Goldberg 2007):

(1) Evolving representations or operators;
(2) Probabilistic modelling for promising solutions;
(3) Perturbation methods.

The objective of the techniques in the first class is to make individual promising sub-solutions separated and less likely to be disrupted by crossover via manipulating the representation of solutions during optimisation. Various reordering and mapping operators have been proposed in the literature, such as self-crossover (Pal, Nandi, and Kundu 1998), which is proven able to generate any arbitrary permutation of the symbols, the messy GA (mGA) (Goldberg et al. 1989), and the fast mGA (fmGA) (Kargupta 1995), which is the more efficient descendant of mGA. The difficulty faced by these methods is that the reordering operator usually reacts too slow and loses the race against selection. Therefore, premature convergence at local optima occurs. Another technique, the linkage learning GA (LLGA) proposed by Harik (1997), uses circular structures as the representation with two-point crossover such that the tight linkage might be more likely preserved. LLGA works well while the shares of BBs are exponentially apportioned in the total fitness,

*Corresponding author. Email: ypchen@cs.nctu.edu.tw

ISSN 0020–7721 print/ISSN 1464–5319 online

http://dx.doi.org/10.1080/00207721.2011.566639
http://www.tandfonline.com

which are usually referred to as exponentially scaled problems. However, it is inefficient when applied to uniformly scaled problems.

The methods in the second category are often referred to as the estimation of distribution algorithms (EDAs) (Mühlenbein and Paaß 1996; Larrañaga and Lozano 2001; Pelikan, Goldberg, and Lobo 2002). These approaches describe the dependencies among variables in a probabilistic manner by constructing a probabilistic model from selected solutions and then sample the built model to generate new solutions. Early EDAs began with assuming no interactions among variables, such as the population-based incremental learning (PBIL) (Baluja 1994) and the compact GA (cGA) (Harik, Lobo, and Goldberg 1999). Subsequent studies started to model pairwise interactions, e.g. the mutual-information input clustering (MIMIC) (de Bonet, Isbell, and Viola 1997), Baluja's dependency tree approach (Baluja and Davies 1997), and the bivariate marginal distribution algorithm (BMDA) (Pelikan and Mühlenbein 1999). Multivariate dependencies were then exploited, and more general interactions were modelled. Example methods include the extended compact GA (ECGA) (Harik 1999), the Bayesian optimisation algorithm (BOA) (Pelikan, Goldberg, and Cantú-Paz 1999), the factorised distribution algorithm (FDA) (Mühlenbein and Mahnig 1999) and the learning version of FDA (LFDA) (Mühlenbein and Höns 2005). Since model constructing in these methods requires no additional fitness evaluations, EDAs are usually considered efficient in the traditional viewpoints of evolutionary computation, especially when fitness evaluations involve time-consuming simulations. However, the model constructing mechanism itself is sometimes computationally expensive with a large population size, which usually occurs in evolutionary methods. The difficulty which EDAs often face is that the BBs contributing less to the total fitness are likely ignored rather than recognised.

Approaches in the third category observe the fitness differences caused by perturbing variables to detect dependencies. In the literature, the gene expression messy GA (GEMGA) (Kargupta 1996) models the sets of tightly linked variables as weights assigned to solutions and employs a perturbation method to detect them. GEMGA observes the fitness changes caused by perturbations on every variable for strings in the population and detects interactions among variables according to how likely the variables compose optimal solutions. Assuming that nonlinearity exists within a BB, the linkage identification by nonlinearity check (LINC) (Munetomo and Goldberg 1998) perturbs a pair of variables and observes the presence of nonlinearities to identify linkages. If the sum of fitness differences of perspective perturbations on two variables is equal to the fitness difference caused by simultaneously perturbing the two variables, linearity is confirmed, and thus, these two variables are considered independent. Instead of non-linearity, the descendant of LINC, linkage identification by non-monotonicity detection (LIMD) (Munetomo and Goldberg 1999), adopts non-monotonicity to detect interactions among variables. Compared to EDAs, the low salience BBs are unlikely ignored in these approaches. However, since obtaining fitness differences requires extra function evaluations, perturbation methods are usually considered demanding more computational efforts to detect linkages. In addition to empirical studies Heckendorn and Wright (2004) generalised these methods through Walsh analysis to obtain theoretical resource requirements. Zhou, Sun, and Heckendorn (2007) and Zhou, Heckendorn, and Sun (2008) later extended this study from the binary domain to high-cardinality domains.

An interesting approach combining the ideas of EDAs and perturbation methods, called *the dependency detection for distribution derived from fitness differences* (D^5), was developed by Tsuji, Munetomo, and Akama (2006). 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 on that variable in the entire population and clusters the strings into sub-populations according to the obtained fitness differences. The sub-populations are examined to find k variables with the lowest entropies, where k is an algorithmic parameter for problem complexity, i.e. the number of variables in a linkage set. The determined k variables are considered forming a linkage set. D^5 can detect dependencies for a class of functions that are difficult for EDAs, e.g. functions containing low salience BBs, and requires less computational cost than other perturbation methods do. However, its major constraint is that it relies on parameter k, which may not be available due to the limited information of the problem structure. As a side-effect to parameter k, D^5 might be fragile in the situation where the problem is composed of subproblems of different sizes. Moreover, Ting, Zeng, and Lin (2010) recently utilised another data mining technique, Apriori Algorithm, to learn potential association rules between decision variables for linkage discovery. They reported that their proposal can improve D^5 in terms of solution quality and efficiency.

In our previous work, we proposed *inductive linkage identification* (ILI) based on perturbations and the integration with the Interative Dichotomiser (ID3) (Quinlan 1986) algorithm, which is widely used in machine learning. ILI is an unsupervised method without any parameter for the complexity of BBs.

Its scalability and efficiency against the increasing problem sizes have been demonstrated (Chuang and Chen 2007, 2008; Huang and Chen 2009, 2010). Compared to the conventional perturbation methods, such as LINC and LIMD, ILI utilises a data mining technique to analyse objective functions. Compared to D^5, which uses clustering, and the method proposed by Ting et al. (2010), which uses Apriori algorithm, ILI adopts the ID3 algorithm and behaves quite differently. In this article, we aim to address more detailed characteristics of ILI in order to gain deeper insights and better understandings of linkage learning. In particular, problems constructed by non-overlapped BBs of different sizes and sub-functions are studied and experimented upon. Our experimental results indicate that ILI holds the properties of robustness and efficiency when facing various configurations of BBs.

The remainder of this article is organised as follows. In Section 2, the background of linkage leaning in GA and decomposability of problems is briefly introduced. Section 3 gives an introduction to ILI, including a review of the ID3 decision tree learning algorithm, an example illustrating the proposed approach, and an algorithmic description of ILI. Section 4 presents the experiments conducted in this study and the results revealing the behaviour of ILI. Finally, Section 5 summaries and concludes this article.

2. Linkage and BBs

In this section, we briefly review the definitions and terminologies which will be used through out this article. As stated by de Jong, Watson, and Thierens (2005), '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 between variables is often referred to as *linkage* in the GA literature. In order to obtain the full linkage information of a pair of variables, the fitness contribution or optimal setting of these two variables will 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 target problem is decomposable. According to the Schema theorem (Holland 1992), short, low-order and highly fit substrings increase their share to be combined. Also stated in the BB hypothesis, GAs implicitly decompose a problem into sub-problems by processing BBs. It is considered that combining small parts is important for GAs and is consistent with human innovation (Goldberg 2002). These lead to a problem model called the *additively decomposable function* (ADF), which can be 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 decision variables $\mathbf{x} = x_1 x_2 \cdots x_\ell$ to represent the encoding scheme adopted by GA users. The fitness of string $\mathbf{s}$ is then defined as

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

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. 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 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 belonging to the same set of V_i is regarded as interdependent because f_i is nonlinear. Thus, we refer to the set V_i as a linkage set. A related term, BBs, is referred to as the candidate solutions to sub-function f_i. In this article, only a subclass of the ADFs is considered. We concentrate on non-overlapping sub-functions. That is, $V_i \cap V_j = \emptyset$ if $i \neq j$. In addition, the strings are assumed to be composed of binary variables.

3. Inductive linkage learning

In this section, the ideas behind ILI will be presented. Then, the ID3 algorithm, which is proposed and widely utilised in the field of machine learning, will be briefly introduced. An example is given to illustratively explain the mechanism of ILI, followed by the pseudo code.

In ILI, linkage learning is regarded as the issue of decision tree learning. As an illustration, the fitness difference can be derived in the following equation within the ADF model:

$$\begin{aligned} f(s_1 s_2 \cdots s_8) &= f_1(s_1 s_2 s_3 s_4 s_5) + f_2(s_6 s_7 s_8) \\ \mathrm{d}f_1(\mathbf{s}) &= f(s_1 s_2 \cdots s_8) - f(\overline{s_1} s_2 \cdots s_8) \\ &= (f_1(s_1 s_2 s_3 s_4 s_5) + f_2(s_6 s_7 s_8)) \\ &\quad - (f_1(\overline{s_1} s_2 s_3 s_4 s_5) + f_2(s_6 s_7 s_8)) \\ &= f_1(s_1 s_2 s_3 s_4 s_5) - f_1(\overline{s_1} s_2 s_3 s_4 s_5). \end{aligned} \tag{2}$$

Equation (2) indicates that the fitness difference $\mathrm{d}f_1$ should be affected only by the bits belonging to the same sub-functions as the perturbed bits s_1, which are $s_1 s_2 \cdots s_5$. Since certain fitness difference values are respectively caused by particular bits arranged in some

permutation of the sub-function where the perturbed variable belongs, we can consider the task as finding which values of variables will result in the corresponding fitness differences.

We found that this kind of tasks is similar to decision making in machine learning: given a condition composed of attributes, an agent (algorithm) should learn to make a decision with the given training instances. When the decision-making method is adopted for conducting linkage learning, decision variables are regarded as attributes and the fitness difference values stand for class labels. With this simple and direct mapping, linkage learning in GAs can potentially be handled with certain well-developed methods in machine learning.

3.1. *Decision tree learning: ID3*

The ID3 algorithm was proposed by Quinlan (1986) for the purpose of constructing a decision tree on a set of training instances. In its basic form, ID3 constructs a decision tree in a top–down manner without backtracking. When a decision tree is being constructed, each attribute is evaluated using a statistical property, called the *information gain*, to measure how well the attribute alone classifies the training instances. The best attribute, which leads to the highest information gain, is accordingly selected and used as the root node of the tree. A descendant node of the root is created for each possible value of the selected attribute, and the training instances are split into appropriate descendant branches. The entire process is repeated on the training instances associated with each descendant node.

The statistical property, information gain, of each attribute is simply the expected reduction in the impurity of instances after classifying the instances with the selected attribute. The impurity of an arbitrary collection of instances is 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

$$Entropy(D) \equiv \sum_{i=1}^{c} -p_i \log_2 p_i, \tag{3}$$

where p_i is the proportion of D belonging to class i. For simplicity, in all the calculations involving entropy, we define $0\log_2 0$ to be 0. In terms of entropy, the information gain, $Gain(D, A)$, of an attribute A relative to a collection of instances D, is defined as

$$Gain(D, A) \equiv Entropy(D) - \sum_{v \in Val(A)} \frac{|D_v|}{|D|} Entropy(D_v), \tag{4}$$

where $Val(A)$ is the set of all possible values for attribute A and D_v is the subset of D of which attribute A has value v. In summary, ID3 can be described as the pseudo code given in Algorithm 1.

Algorithm 1: Pseudo code of ID3

```
procedure ID3(D)
  Stop if no further classification is need
  for each attribute A do
    Calculate Gain(D, A)
  end for
  Select the attribute with the highest information
  gain as a tree node
  for each possible value v of the selected attribute
  do
    Create a branch for D_v, the subset of D of
    which the selected attribute has value v
    Call ID3(D_v) to construct this subtree
  end for
end procedure
```

In the proposal of ILI (Chuang and Chen 2007), the ID3 algorithm is adopted as a classification and relationship extraction mechanism. Linkage learning is then achieved by a sequence of decision tree constructions. In a classification problem, a training instance is composed of a list of attributes describing the instance and a target value which the decision tree is supposed to predict after training. For the purpose of linkage identification, the list of attributes is the solution string, and the target value is the fitness difference caused by perturbations.

3.2. *Exemplary illustration*

This section illustrates the idea that linkage learning is considered as decision learning with an example. We consider a trap function of size k defined as the following:

$$\begin{aligned} f_{trap_k}(s_1 s_2 \cdots s_k) &= trap_k\left(u = \sum_{i=1}^{k} s_i\right) \\ &= \begin{cases} k, & \text{if } u = k; \\ k - 1 - u, & \text{otherwise,} \end{cases} \end{aligned} \tag{5}$$

where u is the number of ones in the string $s_1 s_2 \cdots s_k$. Suppose that we are dealing with an eight-bit problem

$$f(s_1 s_2 \cdots s_8) = f_{trap_5}(s_1 s_2 s_3 s_4 s_5) + f_{trap_3}(s_6 s_7 s_8), \tag{6}$$

where $s_1 s_2 \cdots s_8$ is a solution string. In the black-box optimisation scenario, the structural decomposition of the objective function is unknown. Our goal here is to

Table 1. Population perturbed at s_1.

$s_1s_2\cdots s_8$	f	df_1	$s_1s_2\cdots s_8$	f	df_1
$\bar{0}$0001 011	3	1	$\bar{1}$0010 110	2	−1
$\bar{0}$0011 111	5	1	$\bar{1}$0011 000	1	−1
$\bar{0}$0100 001	3	1	$\bar{1}$0101 010	1	−1
$\bar{0}$0100 100	3	1	$\bar{1}$0101 100	1	−1
$\bar{0}$0101 111	5	1	$\bar{1}$0101 110	1	−1
$\bar{0}$0110 111	5	1	$\bar{1}$0110 101	1	−1
$\bar{0}$1000 111	6	1	$\bar{1}$0111 100	0	−1
$\bar{0}$1010 011	2	1	$\bar{1}$1001 011	1	−1
$\bar{0}$1101 010	1	1	$\bar{1}$1001 111	4	−1
$\bar{0}$1101 100	1	1	$\bar{1}$1010 011	1	−1
$\bar{0}$1101 101	1	1	$\bar{1}$1011 001	0	−1
$\bar{0}$1110 110	1	1	$\bar{1}$1011 010	0	−1
$\bar{0}$1111 001	0	−5	$\bar{1}$1100 000	1	−1
$\bar{0}$1111 110	0	−5	$\bar{1}$1100 010	1	−1
$\bar{0}$1111 111	3	−5	$\bar{1}$1100 011	1	−1
$\bar{1}$0000 010	3	−1	$\bar{1}$1110 101	0	−1
$\bar{1}$0001 010	2	−1	$\bar{1}$1111 010	5	5
$\bar{1}$0010 101	2	−1	$\bar{1}$1111 011	5	5

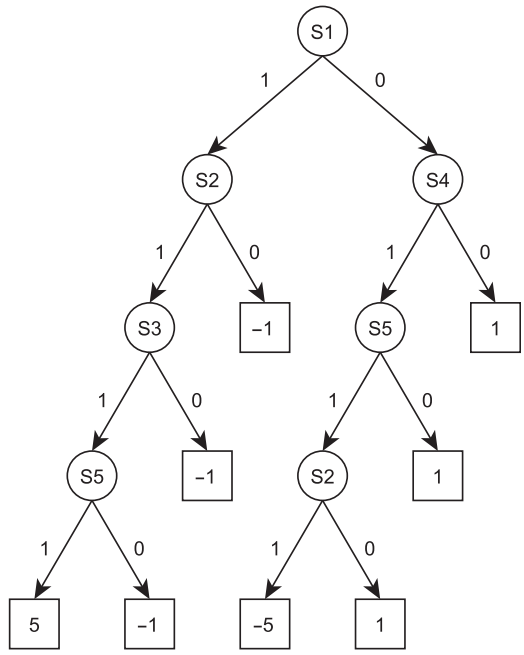

Figure 1. ID3 decision tree constructed according to Table 1.

identify the two linkage sets $V_1=\{1,2,3,4,5\}$ and $V_2=\{6,7,8\}$, which correspond to the problem structural decomposition.

In the beginning, a population of strings is randomly generated as listed in Table 1. 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 ($0\rightarrow 1$ or $1\rightarrow 0$) for all strings in the population in order to detect the variables interdependent on s_1. Note that the choice of first operating on s_1 in this example is not mandatory. Any un-grouped decision variable in the encoding may be chosen as the root node. The third column of Table 1 records the fitness differences, df_1, caused by perturbations at variable s_1.

Then, we construct an ID3 decision tree by using the perturbed population of strings as the training instances and the perturbed variable s_1 as the tree root. Variables in $s_1s_2\cdots s_8$ are regarded as attributes of the instances, and the fitness differences df_1 are the target values/class labels. Corresponding to Table 1, an ID3 decision tree shown in Figure 1 is constructed. By gathering all the decision variables on the non-leaf nodes, we can identify a group of s_1, s_2, s_3, s_4 and s_5. As a consequence, linkage set V_1 is correctly identified.

For the remainder of this example, since s_1, s_2, s_3, s_4 and s_5 are already identified as linkage set V_1, we proceed at s_6. The fitness differences after perturbing variable s_6 are shown in Table 2. Conducting the same procedure, an ID3 decision tree presented in Figure 2 is obtained. By gathering all the decision variables used in the decision tree, we obtain variables s_6, s_7 and s_8, which form linkage set V_2. Because all the decision variables are classified into their respective linkage sets, the linkage detecting task is accomplished. ILI finally reports two linkage sets, $V_1=\{s_1,s_2,s_3,s_4,s_5\}$ and $V_2=\{s_6,s_7,s_8\}$.

Table 2. Population perturbed at s_6.

$s_1s_2\cdots s_8$	f	df_6	$s_1s_2\cdots s_8$	f	df_6
11100 $\bar{0}$00	1	0	10101 $\bar{1}$00	1	0
10011 $\bar{0}$00	1	0	01101 $\bar{1}$00	1	0
11011 $\bar{0}$01	0	0	00100 $\bar{1}$00	3	0
01111 $\bar{0}$01	0	0	10010 $\bar{1}$01	2	0
00100 $\bar{0}$01	3	0	10110 $\bar{1}$01	1	0
11111 $\bar{0}$10	5	0	11110 $\bar{1}$01	0	0
10101 $\bar{0}$10	1	0	01101 $\bar{1}$01	1	0
11100 $\bar{0}$10	1	0	01110 $\bar{1}$10	1	0
10001 $\bar{0}$10	2	0	01111 $\bar{1}$10	0	0
11011 $\bar{0}$10	0	0	01110 $\bar{1}$10	1	0
10000 $\bar{0}$10	3	0	10101 $\bar{1}$10	1	0
01101 $\bar{0}$10	1	0	01111 $\bar{1}$10	0	0
00001 $\bar{0}$11	3	−3	10010 $\bar{1}$10	2	0
00001 $\bar{0}$11	3	−3	00011 $\bar{1}$11	5	3
11010 $\bar{0}$11	1	−3	00011 $\bar{1}$11	5	3
11001 $\bar{0}$11	1	−3	01000 $\bar{1}$11	6	3
11111 $\bar{0}$11	5	−3	00101 $\bar{1}$11	5	3
11100 $\bar{0}$11	1	−3	11001 $\bar{1}$11	4	3
01010 $\bar{0}$11	2	−3	00110 $\bar{1}$11	5	3
10111 $\bar{1}$00	0	0	01111 $\bar{1}$11	3	3

As illustrated in the example, the mechanism of ILI can detect size-varied BBs without assumptions. Such an ability implies that ILI should be capable of finding all relations among these variables as long as the

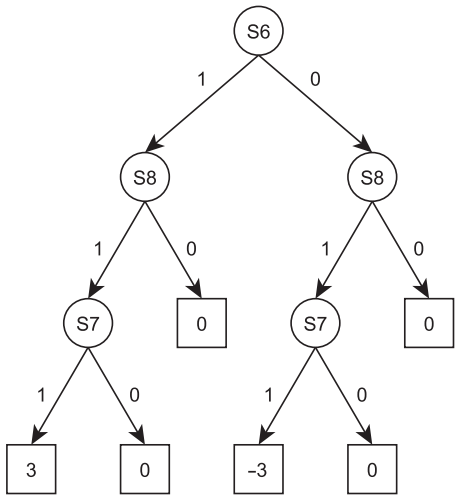

Figure 2. ID3 decision tree constructed according to Table 2.

population size is sufficiently large to provide significant statistics.

3.3. *Inductive linkage identification*

In this section, the idea demonstrated in the previous section is formalized as an algorithm, which is called ILI. The pseudo code of ILI is presented in Algorithm 2. Conceptually, ILI includes the following three main steps:

(1) Calculate fitness differences by perturbations;
(2) Construct an ID3 decision tree;
(3) Consider the tree nodes as a linkage set.

The three steps repeat until all the variables of the objective function are classified into their corresponding linkage sets.

ILI starts with 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 caused by perturbations; (3) the variables used in the tree are gathered and considered as a linkage set.

Algorithm 2: Inductive linkage identification

procedure IDENTIFYLINKAGE(f, ℓ)
  Initialise 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.
      $\mathrm{d}f^i \leftarrow$ fitness difference caused by perturbation.
    end for
    Construct an ID3 decision tree using (P, $\mathrm{d}f$).
    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 linkage sets $V_1, V_2, \ldots, V_m$
end procedure

As clearly shown in Algorithm 2, there is no parameter needed for indicating the complexity of subfunctions. That is, ILI does not rely on any assumption on the size of BBs while other existing perturbation methods usually require the maximum size of BBs to be specified. This property distinguishes ILI from other existing methods. The only factor effecting the correctness of ILI is whether or not the solution strings in the population can provide sufficient information for the decision tree construction.

From our previous studies (Chuang and Chen 2007, 2008), we know that the required population size grows linearly with the problem size while the BBs size is constant. Such results indicate that ILI is more efficient than LINC, $\mathcal{O}(\ell^2) = \mathcal{O}(k^2 m^2)$ (Munetomo and Goldberg 1998), and similar to D^5, $\mathcal{O}(\ell) = \mathcal{O}(km)$ (Tsuji et al. 2006), where ℓ is the problem size, k is the size (i.e. length or order) of BBs, and m is the number of BBs. Note that the comparison focuses on the amount of required computational resource instead of the identification quality. This is because given sufficient computational resource, all these methods can successfully identify every BB. In order to gain further understandings on the flexibility and applicability of ILI, in the next section, experiments on the BBs of different sub-functions as well as lengths are conducted and discussed.

4. Experiments and results

Experiments and results of ILI on binary and non-overlapped ADFs will be presented in this section. These experiments are designed to gain a better understanding of the behaviour of ILI on problems of different sub-functions compositions, including size-varied, size-mixed BBs and different sub-functions.

The required population size reflects the behaviour of ILI. Therefore, our experiments are designed to

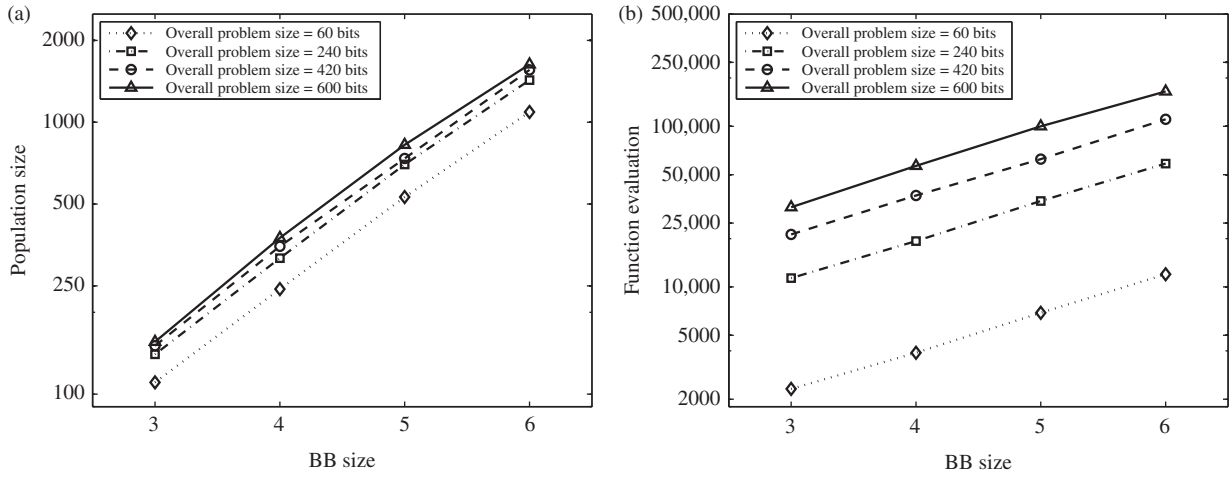

Figure 3. Requirements on different BB sizes; (a) Population size and (b) Function evaluation.

obtain the minimal population sizes required for different problem configurations. For a given problem, first a population size assuring successful trials of linkage identification, which means correctly identifying all the BBs within the problem for 30 consecutive and independent runs, is obtained by doubling the population size from 2500 until the first successful trial is archived. Once the upper bound of population sizes P_U is found, the required population size is determined in a bisection manner: the population size $P = (P_L + P_U)/2$ will be configured for ILI, where $P_L = 1$ for the first iteration. If ILI can succeed with this population size P, then P will be regarded sufficiently large for the problem. The next iteration will perform on the range $[P_L, P]$. Otherwise, the range $[P, P_U]$ will be used. This procedure repeats until the range is smaller than a predefined distance, which is 2 in this study, and the last tested population size is considered as the minimal requirement for the current problem.

4.1. *Different BB sizes*

This section describes the experiment on problems of identical overall sizes but with different-sized sub-functions. From our experimental results with different configurations of the BB size k and the number of BBs m, we group those results with the overall problem sizes and arrange them with the BB size k. Thus, the results of the same problem size with different k can be examined.

Figure 3(a) and (b) shows the experimental results where the overall problem sizes are 60 bits, 240 bits, 420 bits and 600 bits with a log-scaled y-axis. The straight lines indicate that for identical overall problem sizes, the requirements of both the population size and the function evaluation grow exponentially.

With the exponential regression of the experimental results, an estimation of $y = C \times 2^{a \times k}$ can be obtained, where a is a constant around 0.8 and C varies with different problem sizes. Earlier studies by Munetomo and Goldberg (1998) and Heckendorn and Wright (2004), respectively, suggested an empirical and a theoretical upper bounds of function evaluations, which are both in the form of $2^k \ell^j \log(\cdot)$ for problems of ℓ bits, composed of order-k BBs and each BB sharing j bits with others. Reviewing our empirical results with the upper bounds, ILI shows the same computational complexity of the exponential growth with k when overall sizes remain constant, such an observation is consistent with the upper bounds reported in the literature. However, the regression gives 0.8 as the base of exponent and thus indicates a practically better efficiency compared to the suggested upper bound when the complexity of sub-problem increases.

4.2. *Mixed BB sizes*

One of the key features of ILI is unsupervised. In this section, we inspect this feature by conducting experiments on the problems consisting of non-overlapping BBs of order-k_1 and order-k_2 trap functions as

$$trap_{k_1+k_2}(\cdot) = \sum_{i=1}^{m} \big(trap_{k_1}(\cdot) + trap_{k_2}(\cdot) \big), \qquad (7)$$

where m is the number of $trap_{k_1}$ and $trap_{k_2}$. By designing the experiments in this way, the empirical results can be easily compared with those from problems consisting of identical sub-problem complexities in the following manner: for each problem size obtained from the experiment of $trap_{k_1+k_2}(\cdot)$, two

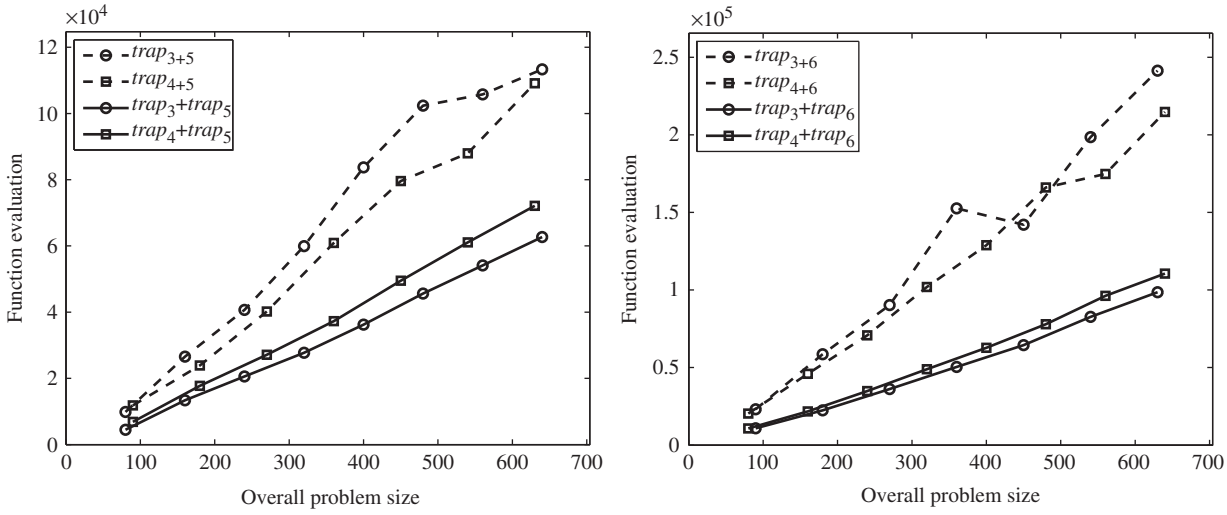

Figure 4. Problems with mixed BB sizes. The solid lines represent the actual experimental results while the dashed lines are the summed up calculations from Section 4.1.

results of the same amount of $trap_{k_1}$ and $trap_{k_2}$ from experiments in Section 4.1 are summed up to get the same problem size and total number of BBs, interpolation is utilised when there are no results of such configurations. These calculated numbers are denoted as $trap_{k_1} + trap_{k_2}$ in Figure 4 with the experimental results $trap_{k_1+k_2}$.

First, these results show that ILI is capable of detecting BBs of different sizes within one problem without any extra information regarding the complexity of sub-problems. Second, comparing with calculated data, it can be seen that although ILI requires more function evaluations for the problems composed of mixed BB sizes, the growth rate is still linear or very close to linear. The observation indicates that identifying size-varied BBs within a problem poses no particular difficulties for ILI. Such a property of robustness makes ILI more practical when being applied to real world problems where information regarding the sub-problem complexity is usually unavailable and no guideline exists to make appropriate assumptions.

4.3. *BBs of various elementary functions*

Despite of using $trap_k$ functions as the sub-function to construct BBs, the capability of ILI to handle BBs formed by other functions shown in Figure 5 is examined in this section. These elementary functions are used to compose the objective function according to the ADF model, and the complexity of order 4 is used in this section.

Figure 6 shows the experimental results. The required population sizes and function evaluations of $trap_4$, $nith_4$, $tmmp_4$ and $valley_4$ are plotted together, and the standard deviation of the results for $trap_4$ is also shown in the figures. Because the population and function evaluation requirements of these problems are similar, the behaviour of ILI should also be similar for problems constructed by mixing sub-problems of the same complexity. Moreover, the applicability of ILI on a wide range of problems is also confirmed. ILI is capable of detecting the interactions among variables as long as a sufficiently large population is employed to provide significant statistics.

5. Summary and conclusions

In this article, we examined ILI on several different configurations of BBs in order to gain better understandings. We focused on the mixed sizes of BBs and the elementary functions of different types. These series of experiments verified the efficiency of ILI on the population requirement growth, the robustness of ILI on mixed sizes of BBs, and the applicability of ILI on BBs formed with various elementary functions.

From the experiments of BB sizes, it is demonstrated that the required function evaluations grow exponentially with the size of BBs when the overall problem size remains constant. Such a result is consistent with the conclusions of previous studies from other researchers in the manner of Big-$\mathcal{O}$ while ILI demands less computational resource in practice. On the other hand, if computationally expensive real-world problems, such as parametric engineering design (Saridakis and Dentsoras 2009), are handled, and the optimisation framework has to be made much more efficient, techniques of the *surrogate-assisted*

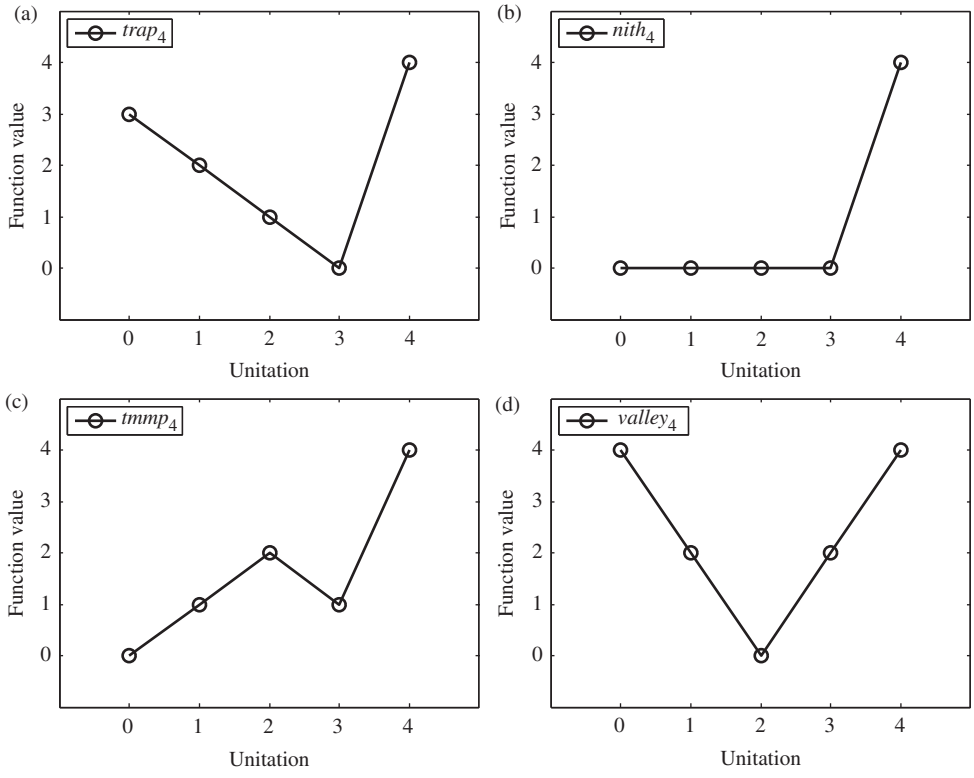

Figure 5. Elementary functions adopted in the series of experiments in Section 4.3; (a) $trap_4$, (b) $nith_4$, (c) $tmmp_4$ and (d) $valley_4$.

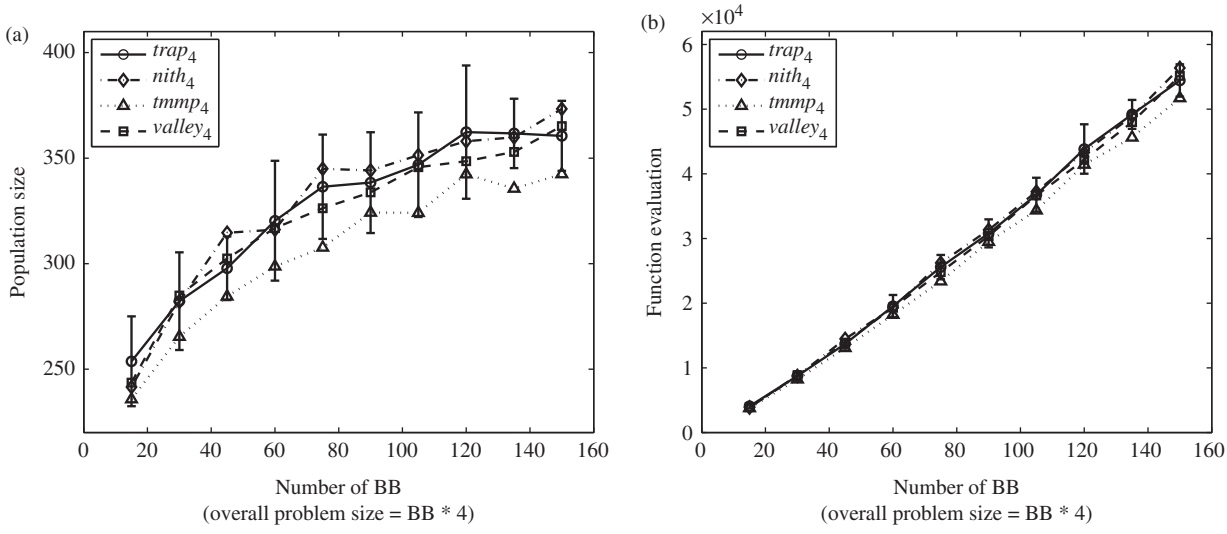

Figure 6. Experimental results on different 4-bits BB types: (a) required population sizes and (b) required function evaluations.

evolutionary algorithm (SAEA) (Sastry, Goldberg, and Pelikan 2001; Jin 2003; Lim, Jin, Ong, and Sendhoff 2010a; Lim, Ong, Setiawan, and Idris 2010b) may be adopted and utilised.

Another observation is that when ILI performs on problems composed of mixed-sized BBs, the computational complexity of ILI is still in the same order. This phenomenon indicates that detecting these more complicated problem structures poses no particular difficulty for ILI. Finally, the experimental results obtained by using four different elementary functions to construct BBs are quite similar. Thus, this series of experiments evidentially proves that ILI behaves similarly when handling sub-problem of different types.

As a consequence, we can now know that the most important factor that affects ILI's ability to identify linkage is the size of BBs. Compared with the BB size, ILI is relatively insensitive to other factors commonly studied by the related work, including the overall problem size, the number of BBs, and the type of BBs. Hence, ILI can be considered as a good linkage learning technique and can be adopted as a tool for analysing structures of target problems or a pre-processing procedure in frameworks of GAs.

Since its introduction, ILI as a linkage learning technique has been empirically proven efficient, robust and widely applicable. Research along this line includes integrating ILI into a GA framework, handling real-world applications with ILI, exploring ILI's capability of analysing problem structures and understanding the nature of linkage learning via getting deeper insights of ILI. As for the immediate future studies, the idea of 'linkage identification as decision learning' can be adapted to work with other advanced decision tree techniques. Characteristics of different decision tree algorithms might exhibit behaviour of different kinds and give us a better understanding of linkage identification. Such knowledge can be utilised to practically help the algorithmic development of GAs and theoretically reveal the working principle of evolutionary computation.

Acknowledgements

The work was supported in part by the National Science Council of Taiwan under Grant NSC 99-2221-E-009-123-MY2. The authors are grateful to the National Center for High-performance Computing for computer time and facilities.

Notes on contributors

Ying-ping Chen received his BS and MS degrees in Computer Science and Information Engineering from National Taiwan University, Taiwan, in 1995 and 1997, respectively, and PhD in 2004 from the Department of Computer Science, University of Illinois at Urbana-Champaign, Illinois, USA. He has been an Assistant Professor from 2004 to 2009 and an Associate Professor since 2009 in the Department of Computer Science, National Chiao Tung University, Taiwan. His research interests in the field of genetic and evolutionary computation include theories, working principles, particle swarm optimisation, estimation of distribution algorithms, linkage learning techniques and dimensional/facet-wise models.

Chung-Yao Chuang received his BS and MS degrees in Computer Science from National Chiao Tung University, Taiwan, in 2006 and 2008, respectively. He is currently working at Academia Sinica, Taiwan for obligated citizen service and looking forward to being included in a PhD program starting at Fall 2012. His research interests include linkage problem in evolutionary algorithms, estimation of distribution algorithms, evolutionary computation and machine learning in general.

Yuan-Wei Huang received the BS and MS degrees in Computer Science from National Chiao Tung University, Taiwan, in 2008 and 2010, respectively. His research interests include linkage learning techniques, machine learning, and artificial intelligence.

References

Baluja, S. (1994), 'Population-based Incremental Learning: A Method for Integrating Genetic Search Based Function Optimization and Competitive Learning', Technical Report, Pittsburgh, PA, USA.

Baluja, S., and Davies, S. (1997), 'Using Optimal Dependency-trees for Combinational Optimization', in *Proceedings of the Fourteenth International Conference on Machine Learning*, *ICML'97*, pp. 30–38.

Chen, Y.-p., Yu, T.-L., Sastry, K. and Goldberg, D.E. (2007), 'A Survey of Linkage Learning Techniques in Genetic and Evolutionary Algorithms', IlliGAL Report No. 2007014, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign.

Chuang, C.Y., and Chen, Y.P. (2007), 'Linkage Identification by Perturbation and Decision Tree Induction', in *Proceedings of 2007 IEEE Congress on Evolutionary Computation (CEC, 2007)*, pp. 357–363.

Chuang, C.Y., and Chen, Y.P. (2008), 'Recognizing Problem Decomposition with Inductive Linkage Identification: Population Requirement vs. Subproblem Complexity', in *Proceedings of the Joint 4th International Conference on Soft Computing and Intelligent Systems and 9th International Symposium on Advanced Intelligent Systems (SCIS & ISIS, 2007)*, pp. 670–675.

de Bonet, J., Isbell, C., and Viola, P. (1997), 'MIMIC: Finding Optima by Estimating Probability Densities', *Advances in Neural Information Processing Systems*, 9, 424–430.

de Jong, E.D., Watson, R., and Thierens, D. (2005), 'On the Complexity of Hierarchical Problem Solving', in *Proceedings of Genetic and Evolutionary Computation Conference 2005 (GECCO, 2008)*, pp. 1201–1208.

Gladwin, D., Stewart, P., and Stewart, J. (2011), 'Internal Combustion Engine Control for Series Hybrid Electric Vehicles by Parallel and Distributed Genetic Programming/Multiobjective Genetic Algorithms', *International Journal of Systems Science*, 42, 249–261.

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

Goldberg, D.E., Korb, B., and Deb, K. (1989), 'Messy Genetic Algorithms: Motivation, Analysis, and First Results', *Complex Systems*, 3, 493–530.

Harik, G.R. (1997), 'Learning Gene Linkage to Efficiently Solve Problems of Bounded Difficulty Using Genetic Algorithms', Ph.D. Dissertation, University of Michigan, Ann Arbor, MI, USA.

Harik, G. (1999), 'Linkage Learning via Probabilistic Modeling in the ECGA, IlliGAL Report No. 99010, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign.

Harik, G.R., Lobo, F.G., and Goldberg, D.E. (1999), 'The Compact Genetic Algorithm', *IEEE Transactions on Evolutionary Computation*, 3, 287–297.

Heckendorn, R.B., and Wright, A.H. (2004), 'Efficient Linkage Discovery by Limited Probing', *Evolutionary Computation*, 12, 517–545.

Holland, J.H. (1992), *Adaptation in Natural and Artificial Systems*, Cambridge, MA, USA: MIT Press.

Huang, Y.W., and Chen, Y.-p. (2009), 'On the Detection of General Problem Structures by Using Inductive Linkage Identification', in *Proceedings of ACM SIGEVO Genetic and Evolutionary Computation Conference 2009 (GECCO, 2009)*, pp. 1853–1854.

Huang, Y.W., and Chen, Y.-p. (2010), 'Detecting General Problem Structures with Inductive Linkage Identification', in *Proceedings of the 2010 Conference on Technologies and Applications of Artificial Intelligence*, TAAI, pp. 508–515.

Jin, Y. (2003), 'A Comprehensive Survey of Fitness Approximation in Evolutionary Computation', *Soft Computing*, 9, 3–12.

Kargupta, H. (1995), *SEARCH,* 'Polynomial Complexity, and the Fast Messy Genetic Algorithm', Technical Report, University of Illinois.

Kargupta, H. (1996), 'The Gene Expression Messy Genetic Algorithm', in *Proceedings of IEEE International Conference on Evolutionary Computation*, pp. 814–819.

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

Li, S., and Li, Z.Z. (2010), 'Spare Parts Allocation by Improved Genetic Algorithm and Monte Carlo Simulation', *International Journal of Systems Science*, First published on: 01 March 2010 (iFirst). DOI: 10.1080/00207720802556252.

Lim, D., Jin, Y., Ong, Y.S.O., and Sendhoff, B. (2010a), 'Generalizing Surrogate-assisted Evolutionary Computation', *IEEE Transactions on Evolutionary Computation*, 14, 329–355.

Lim, D., Ong, Y.S., Setiawan, R., and Idris, M. (2010b), 'Classifier-assisted Constrained Evolutionary Optimization for Automated Geometry Selection of Orthodontic Retraction Spring', in *Proceedings of 2010 IEEE Congress on Evolutionary Computation (CEC, 2010)*, pp. 1449–1456.

Mühlenbein, H., and Höns, R. (2005), 'The Estimation of Distributions and the Minimum Relative Entropy Principle', *Evolutionary Computation*, 13, 1–27.

Mühlenbein, H., and Mahnig, T. (1999), 'FDA – A Scalable Evolutionary Algorithm for the Optimization of Additively Decomposed Functions', *Evolutionary Computation*, 7, 353–376.

Mühlenbein, H., and Paaß, G. (1996), '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)*, pp. 178–187.

Munetomo, M., and Goldberg, D.E. (1998), 'Identifying Linkage by Nonlinearity Check', IlliGAL Report No. 98012, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign.

Munetomo, M., and Goldberg, D.E. (1999), 'Identifying Linkage Groups by Nonlinearity/Non-monotonicity Detection', in *Proceedings of Genetic and Evolutionary Computation Conference 1999 (GECCO-99)*, pp. 433–440.

Pal, N.R., Nandi, S., and Kundu, M.K. (1998), 'Self-crossover – A New Genetic Operator and Its Application to Feature Selection', *International Journal of Systems Science*, 29, 207–212.

Pelikan, M., Goldberg, D.E., and Cantú-Paz, E. (1999), 'BOA: The Bayesian Optimization Algorithm', in *Proceedings of Genetic and Evolutionary Computation Conference 1999 (GECCO-99)*, pp. 525–532.

Pelikan, M., Goldberg, D.E., and Lobo, F.G. (2002), 'A Survey of Optimization by Building and Using Probabilistic Models', *Computational Optimization and Applications*, 21, 5–20.

Pelikan, M., and Mühlenbein, H. (1999), 'The Bivariate Marginal Distribution Algorithm', *Advances in Soft Computing – Engineering Design and Manufacturing*, 521–535.

Quinlan, J.R. (1986), 'Induction of Decision Trees', *Machine Learning*, 1, 81–106.

Saridakis, K., and Dentsoras, A. (2009), 'Integration of Genetic Optimisation and Neuro-fuzzy Approximation in Parametric Engineering Design', *International Journal of Systems Science*, 40, 131–145.

Sastry, K., Goldberg, D.E., and Pelikan, M. (2001), 'Don't Evaluate, Inherit', in *Proceedings of Genetic and Evolutionary Computation Conference 2001 (GECCO-2001)*, pp. 551–558.

Stonedahl, F., Rand, W., and Wilensky, U. (2008), 'CrossNet: A Framework for Crossover with Network-based Chromosomal Representations', in *Proceedings of ACM SIGEVO Genetic and Evolutionary Computation Conference 2008 (GECCO, 2008)*, pp. 1057–1064.

Ting, C.K., Zeng, W.M., and Lin, T.C. (2010), 'Linkage Discovery Through Data Mining', *IEEE Computational Intelligence Magazine*, 5, 10–13.

Tsuji, M., Munetomo, M., and Akama, K. (2006), 'Linkage Identification by Fitness Difference Clustering', *Evolutionary Computation*, 14, 383–409.

Wang, K. (2009), 'Application of Genetic Algorithms to Robot Kinematics Calibration', *International Journal of Systems Science*, 40, 147–153.

Zhou, S., Heckendorn, R.B., and Sun, Z. (2008), 'Detecting the Epistatic Structure of Generalized Embedded Landscape', *Genetic Programming and Evolvable Machines*, 9, 125–155.

Zhou, S., Sun, Z., and Heckendorn, R.B. (2007), 'Extended Probe Method for Linkage Discovery Over High-cardinality Alphabets', in *Proceedings of ACM SIGEVO Genetic and Evolutionary Computation Conference 2007 (GECCO-2007)*, pp. 1484–1491.