A Fast Algorithm for Computing the Deceptive Degree of an Objective Function

LI Yun-qiang

Electronic Technique Institute,
Zhengzhou Information Engineering University,
450004 Zhengzhou, China

Abstract. In this paper we present a fast algorithm for computing the deceptive degree of an objective function. We discuss theoretical foundations of the fast algorithm and how to get the polynomial representation of a function quickly under the condition of that the function value of every input is known. We prove a fast decision theorem of whether a monomial has deception about a variable in it, which makes computing the deceptive degree of a function easier. In the final, we describes the fast algorithm and analyses its complexity.

1 Introduction

Since Goldberg introduced the notion of *deception* in genetic algorithms (GAs), deception has come to be widely regarded as a central feature in the design of problems that are difficult for GAs. Though a number of different definitions of deception as well as types of deception have been proposed in the GA literature(e.g., see [2,3,4,5,6]), the relationship between a deceptive problem and a difficult problem is still not easy to be explained. Some counterexamples in [7,8,9,10] show that deception is neither necessary nor sufficient to make a problem difficult for a GA. For there is no generally accepted definition of deception, how to define deception is still a topic that deservers scrutiny.

In [1] we presented a novel quantitative measure metric for the "degree of deception" of a problem. We presented a new definition for the deceptive degree of a function. We investigated the relationship between the best solution and the monomial coefficients of a function, and we gave theorems and experiments that showed the usefulness of the new definition. For it is a complex work to computing the deceptive degree of a function according to the new definition in [1], we have to consider whether there is a fast algorithm for computing the deceptive degree of a function.

In this paper we will present a fast algorithm for computing the deceptive degree of a function which was defined in [1]. The remainder of the paper is organized as follows: Section 2 reviews the new definition and the main results in [1]. The next two sections show theoretical foundations of the fast algorithm. Section 3 discusses how to get the polynomial representation of a function quickly under the condition of that the function value of every input is known. Section 4 proves a decision theorem of whether a monomial has deception about a variable in it, which makes computing the deceptive degree of a function easier. Section 5 describes the fast algorithm and analyses the complexity of the algorithm. Section 6 summarizes the paper.

2 The new definition and the main results in [1]

In this section we will review the new definition for the deceptive degree of a function in [1]. Section 2.1 introduces the notation to simplify the expression in this paper. Section 2.2 gives the definition and properties of a critical value which plays an important role in the new definition of deception. Section 2.3 describes the new definition which was presented in [1]. Some results in [1] which may be useful for the fast algorithm is given in lemma form.

2.1 Notation

Without loss of generality, we consider $\Omega^n = \{0,1\}^n$ as the search space for GAs and a pseudo-boolean function f : $\{0,1\}^n \to R$ as a fitness function. The goal is to find a best string (or solution) which is correlative to the maximum of the function f. Because the new definition of deception was presented according to the relation between the best string and monomial coefficients of a given function, we must consider the function in polynomial form. Every pseudo-boolean function can be expressed in polynomial form. By convention, f will denote a pseudo-boolean function in polynomial form:

$$f(x) = a_0 + a_1 x_1 + \cdots + a_n x_n + \cdots a_{i_1 \cdots i_k} x_{i_1} \cdots x_{i_k} + \cdots + a_{12\cdots n} x_1 x_2 \cdots x_n$$

where $a_{i_1 \cdots i_k}$ is called the coefficient of the monomial $x_{i_1} \cdots x_{i_k}$, k is called the degree of the monomial $x_{i_1} \cdots x_{i_k}$, and the maximum of the degrees of all monomials is called the polynomial degree of *f*. $f_-(i_1, \cdots, i_k)$ = $f - a_{i_1 i_2 \cdots i_k} x_{i_1} x_{i_2} \cdots x_{i_k}$ will denote *f* without the monomial $x_{i_1} \cdots x_{i_k}$. *K* will always represent a subset of $\{1,2,\cdots,n\}$, and $f[(i_k, \theta_{i_k}); i_k \in K]$ will denote *f* where $x_{i_k} = \theta_{i_k}$ for all $i_k \in K$ and $\theta_{i_k} \in \{0,1\}$. For example, let $f(x) = x_1 + x_2 + x_3 - x_1 x_2 x_3$, then $f_-(1,2,3) = x_1 + x_2 + x_3$, $f[(1,1),(2,0)] = 1 + 0 + x_3 - 1 \bullet 0 \bullet x_3 = 1 + x_3$.

A schema has been widely studied in the field of GAs, and is also an important concept in the new definition of deception. A schema corresponds to a subset of the search space $\Omega^n = \{0,1\}^n$, or more precisely a hyper-plane of Ω^n. An additional symbol "*" representing a wild card ("0" or "1") is used to represent a schema. For example, if n=4,the strings 0101 and 1101 are the two elements of the schema S=*101. Non-* positions in a schema are called defining positions. Defining positions and their corresponding values can be used to get another representation of a schema. For example, S[(2,1),(3,0),(4,1)] is also used to represent the schema S=*101. The latter representation of a schema is often used in this paper.

2.2 Definition and properties of a critical value

A critical value plays an important role in the new definition of deception. Now we gives the concept and properties of a critical value of a monomial with respect to a variable.

Definition 1[1]. *For a given function f ,* λ *is called a critical value of the monomial* $x_{i_1} \cdots x_{i_k}$ *with respect to the variable* x_{i_1} *if for arbitrary* ε *>0, all the best strings of* $f_-(i_1, \cdots, i_k)$ + $(\lambda + \varepsilon)\, x_{i_1} \cdots x_{i_k}$ *are included in the schema* $S[(i_1,1)]$ *and one of the best strings of* $f_-(i_1, \cdots, i_k)$ +$(\lambda - \varepsilon)\, x_{i_1} \cdots x_{i_k}$ *is included in the schema* $S[(i_1,0)]$*, which means that all the maximum of* $f_-(i_1, \cdots, i_k)$ + $(\lambda + \varepsilon)\, x_{i_1} \cdots x_{i_k}$ *satisfy* $x_{i_1} = 1$ *and one of the maximum of* $f_-(i_1, \cdots, i_k)$ +$(\lambda - \varepsilon)\, x_{i_1} \cdots x_{i_k}$ *satisfies* $x_{i_1} = 0$.

Remark. If there exists a critical value of the monomial $x_{i_1} \cdots x_{i_k}$ with respect to the variable x_{i_1} in f , then the critical value is unique. If multiple critical values of the same monomial with respect to all variables exist, if exist , they are equal. If one of the best strings of f is included in the schema $S[(i_1,0)]$, then there exists a critical value of each monomial $x_{i_1} \cdots x_{i_k}$ with respect to the variable x_{i_1}. In other words, If there doesn't exist a critical value of the monomial $x_{i_1} \cdots x_{i_k}$ with respect to the variable x_{i_1} , then all the best strings of f must be included in the schema S[(i$_1$,1)].

Lemma 1[1]. *If a critical value of the monomial* $x_{i_1} \cdots x_{i_k}$ *with respect to the variable* x_{i_1} *of f is 0, then*

$$\max\{f_-(i_1, \cdots, i_k)[(i_1,0)]\} = \max\{f_-(i_1, \cdots, i_k)[(i_1,1)]\}$$

If there doesn't exist a critical value of the monomial $x_{i_1} \cdots x_{i_k}$ with respect to the variable x_{i_1} in f, we say the critical value is $-\infty$ where $-\infty$ denotes a negative infinite number. The following result can be proved.

Lemma 2[1]. *If the coefficient of the monomial* $x_{i_1} \cdots x_{i_k}$ *is greater than critical values of the monomial* $x_{i_1} \cdots x_{i_k}$ *with respect to all variables, then the best string must be included in* $S[(i_1,1),(i_2,1),\cdots,(i_k,1)]$*. If the coefficient of the monomial* $x_{i_1} \cdots x_{i_k}$ *is less than one of critical values of the monomial* $x_{i_1} \cdots x_{i_k}$ *with respect to all variables, then the best string mustn't be included in* $S[(i_1,1),(i_2,1),\cdots,(i_k,1)]$.

2.3 The new definition of deception

In the following, we discussed whether a monomial has deception about a variable according to the relation between a monomial coefficient and a critical value.

Definition 2[1]. *If a variable in a monomial satisfies one of the following three conditions, we say that the monomial has deception about the variable.*

1. *If a critical value of the monomial with respect to the variable is 0;*
2. *If a critical value of the monomial with respect to the variable is positive and less than the monomial coefficient;*
3. *If a critical value of the monomial with respect to the variable is negative and greater than the monomial coefficient.*

Obviously these three conditions contradict each other and at most one can be satisfied.

Lemma 3[1]. *Let f have only one best string, then the monomial* $x_{i_1}\cdots x_{i_k}$ *doesn't have deception about every variable in it if and only if the best string of* $f_{-}(i_1,\cdots,i_k)$ *is the same as the best string of f.*

Remark: Lemma 2 and 3 are help to understand the new definition of deception. For a monomial $x_{i_1}\cdots x_{i_k}$, different bits between the best string of *f* and the best string of $f_{-}(i_1,\cdots,i_k)$ can be used to depict the GA difficulty. If it is easy for a GA to find the best string of *f*, it may be not easy for the GA to find the best string of $f_{-}(i_1,\cdots,i_k)$ only because of these different bits, and vice visa. In fact, for a monomial $x_{i_1}\cdots x_{i_k}$, if the best string of *f* is different from the best string of $f_{-}(i_1,\cdots,i_k)$, one of the two best strings must be included in the schema $S[(i_1,1),(i_2,1),\cdots,(i_k,1)]$. When we assume that the best string of *f* is included in the schema $S[(i_1,1),(i_2,1),\cdots,(i_k,1)]$, then the number of variables for which the monomial has deception can reflect the difficulty of a GA evolving to the schema $S[(i_1,1),(i_2,1),\cdots,(i_k,1)]$. Furthermore, if the best string of *f* is $11\cdots1$, we can reflect the evaluation of the GA difficulty of *f* by the number of variables about which the monomial has deception.

For simplicity, we only discussed the deceptive degree of a function that has only one best string.

Definition 3[1]. *Let the best string of a function be* $11\cdots1$*, for a monomial whose coefficient isn't equal to 0 , the number of variables about which the monomial has deception is called the deceptive degree of the monomial. The maximum of deceptive degrees of monomials of the function is called the deceptive degree of the function.*

Let $\oplus$ denote the binary XOR operation on bits and the bitwise XOR operation on strings.When the best string of a function isn't $11\cdots1$, the deceptive degree of the function can be computed as follows.

Definition 4[1]. *Let the best string of f be* $11\cdots1$*, and*

$$g(x_{i_1},x_{i_2},\cdots,x_{i_n}) = f(x_{i_1}\oplus 1,x_{i_2},\cdots,x_{i_n}),$$

then we define the deceptive degree of g to be equal to the deceptive degree of f.

From Definition 3 and 4, the deceptive degrees of every function which has only one best string can be computed, the deceptive degrees of functions range from 1 to n.

Lemma 4[1]. *For an arbitrary constant* $C\in\{0,1\}^n$, *the deceptive degree of* $f(x\oplus C)$ *is equal to the deceptive degree of f* (*x*).

Lemma 5[1]. *The deceptive degree of a function isn't greater than the polynomial degree of the function.*

3 How to get the polynomial representation of a function quickly

In order to get the fast algorithm for computing the deceptive degree of a function, we must resolve two problems which are theoretical foundations of the fast algorithm, one is how to get the polynomial representation of a pseudo-boolean function quickly, another is how to computing the deceptive degree of a monomial quickly. In the following, we will discuss how to resolve the first problem.

3.1 A theory for getting the polynomial representation

If we know the polynomial representation of a pseudo-boolean function, we can get the value of the function for every string (input) easily. But if we know the value of a pseudo-boolean function for every string (input), how to get the polynomial representation of the function quickly is worth to consider.

Theoretically we can get the polynomial representation of a function as follows. First, a pseudo-boolean function *f*: $\{0,1\}^n \to R$, should be written in the small item presentation:

$$f(x)=\sum_{c=0}^{2^n-1} f(c)x_1^{c_1}x_2^{c_2}\cdots x_n^{c_n} \qquad (1)$$

where

$$c=(c_1,\cdots,c_n)\in\{0,1\}^n,\qquad x=(x_1,\cdots,x_n)\in\{0,1\}^n,$$

$$x_1^{c_1}x_2^{c_2}\cdots x_n^{c_n}=\begin{cases}1, & if\ (x_1,x_2,\cdots,x_n)=(c_1,c_2,\cdots,c_n)\\ 0, & if\ (x_1,x_2,\cdots,x_n)\neq(c_1,c_2,\cdots,c_n)\end{cases}.$$

Then , let

$$x_i^1=x_i,\ \ x_i^0=1-x_i,\qquad i=1,2,\cdots,n.$$

For every $i\ (i=1,\cdots,n)$, We substitute x_i for x_i^1 and $1-x_i$ for x_i in (1). Final, through the normal polynomial operations over real filed , we can get the polynomial representation of *f* in this form:

$$f(x)=a_0+a_1x_1+\cdots+a_nx_n+\cdots a_{i_1\cdots i_k}x_{i_1}\cdots x_{i_k}+\cdots+a_{12\cdots n}x_1x_2\cdots x_n$$

Now we discuss how to get the polynomial representation of a function by a computer program. First, we need code all coefficients in the polynomial representation of *f*. The code rule is as follows: The code number of every coefficient is a n-length binary number. if $i\ (i=1,\cdots,n)$ is included in the lower ordinate set of a coefficient, the *i*th bit of the corresponding code number is 1. The ith bit is 0 in another case. For instance, the corresponding code number of a_0 is $(0,0,\cdots,0)=0$, the corresponding code number of a_1 is $(1,0,\cdots,0)=2^{n-1}$, the corresponding code number of a_{12} is $(1,1,\cdots,0)=2^{n-1}+2^{n-2}$, the corresponding code number of $a_{12\cdots n}$ is $(1,1,\cdots,1)=2^{n-1}-1$. Obviously ,every coefficient has and only has a code number, 2^n coefficients are corresponding to 2^n code numbers.

In the following theorem, we will describe how a small item $f(c)x_1^{c_1}x_2^{c_2}\cdots x_n^{c_n}$ of *f* has influence on polynomial coefficients of *f* .

Theorem 1. Let array A[k] represent polynomial coefficients of *f* , for every $c\in\{0,1\}^n$,

$$K_c=\{k=(k_1,k_2,\cdots,k_n)\mid c=(c_1,c_2,\cdots,c_n)\in\{0,1\}^n,k_i\in\{c_i,1\},i=1,2,\cdots,n\}$$

If $k\notin K_c$, the small item $f(c)x_1^{c_1}x_2^{c_2}\cdots x_n^{c_n}$ of *f* has no influence on A[k]. If $k\in K_c$, the small item $f(c)x_1^{c_1}x_2^{c_2}\cdots x_n^{c_n}$ of *f* makes A[k] increase $(-1)^{w(k)-w(c)}f(c)$. Where $w(k),w(c)$ are the number of 1 in $(k_1,k_2,\cdots,k_n)$, $(c_1,c_2,\cdots,c_n)$ respectively.

Proof. Obviously, if $f(c)=0$, the small item $f(c)x_1^{c_1}x_2^{c_2}\cdots x_n^{c_n}$ of *f* has no influence on A[k] for every $k\in\{0,1\}^n$. Now we consider only the case of $f(c)\neq 0$. For $x_i^1=x_i,\ x_i^0=1-x_i,(i=1,2,\cdots,n)$,then if $c_i=1$,all monomials gotten from $f(c)x_1^{c_1}x_2^{c_2}\cdots x_n^{c_n}$ must include x_i; if $c_i\neq 1$,only half of monomials gotten from

$f(c)x_1^{c_1}x_2^{c_2}\cdots x_n^{c_n}$ include x_i. So if $c_i = 1$, $f(c)x_1^{c_1}x_2^{c_2}\cdots x_n^{c_n}$ has influence on A[k] where $k_i = 1$; if $c_i = 0$, $f(c)x_1^{c_1}x_2^{c_2}\cdots x_n^{c_n}$ has influence on A[k] where $k_i = 0$ or $k_i = 1$. For all $i\ (i = 1,2,\cdots.n)$,we can get that $f(c)x_1^{c_1}x_2^{c_2}\cdots x_n^{c_n}$ has influence on A[k] where $k \in K_c$ and $f(c)x_1^{c_1}x_2^{c_2}\cdots x_n^{c_n}$ has no influence on A[k] where $k \notin K_c$.

$f(c)x_1^{c_1}x_2^{c_2}\cdots x_n^{c_n}$ has influence on A[k] where $k \in K_c$ and make A[k] change. The corresponding monomial of A[k], which is gotten form $f(c)x_1^{c_1}x_2^{c_2}\cdots x_n^{c_n}$, is the product of a $f(c)$, $n - w(k)$ ones, $w(k) - w(c)$ minus ones and $w(k)$ variables. So $f(c)x_1^{c_1}x_2^{c_2}\cdots x_n^{c_n}$ makes A[k] increase $(-1)^{w(k)-w(c)} f(c)$.

3.2 A procedure for getting the polynomial representation

Now we can give a procedure for getting the polynomial representation of a function under the condition of that the function value of every input is known. The concrete steps are as follows: First , let array A[k] represent polynomial coefficients of *f*, the range of k is from 0 to 2^n-1, set A[k]=0 for every k. Then ,for every $c \in \{0,1\}^n$, if $f(c) \neq 0$, we make A[k] increase $(-1)^{w(k)-w(c)} f(c)$ for every $k \in K_c$. In the final, the value of A[k] is the value of the corresponding monomial coefficient .

Procedure 1 describes the concrete steps in a similar computer program language.

Procedure 1
```
begin
    for k←0 to $2^n$-1 do A[k]←0;
    for c←0 to $2^n$-1 do
      if( $f(c) \neq 0$ )
      begin
        for $k_1$←$c_1$ to 1 do
        for $k_2$←$c_2$ to 1 do
             ⋮
        for $k_n$←$c_n$ to 1 do
        A[k]←A[k] + $(-)^{w(k)-w(c)} f(c)$
      end
end
```

4 How to compute the deceptive degree of a monomial quickly

If we want to compute the deceptive degree of a function, we must compute the deceptive degree of every monomial of the function according to Definition 3 and 4. then we must compute the corresponding critical values according Definition 1 and 2, which is a complex work. Now we will present a new method to compute the deceptive degree of a monomial.

4.1 A new method to compute the deceptive degree of a monomial

In the following, we will present a decision theorem of whether a monomial has deception about a variable in it. Furthermore, we can get the deceptive degree of a monomial by some simple comparisons.

Theorem 2. *Let f have only one best string which is* $11\cdots1$ *, and for every* $i\ (i = 1,2,\cdots,n)$ *,let* $partbest[i] = \max\{f(x) \mid x \in S[(i,0)]\}$. *If* $a_{i_1\cdots i_k} < 0$, *then the monomial* $x_{i_1}\cdots x_{i_k}$ *doesn't have deception about every variable in it. If* $a_{i_1\cdots i_k} > 0$ *and* $partbest[i_1] < \max\{f(x)\} - a_{i_1 i_2\cdots i_k}$, *then the monomial* $x_{i_1}\cdots x_{i_k}$

doesn't have deception about the variable x_{i_1}. *If* $a_{i_1\cdots i_k} > 0$ *and* $partbest[i_1] \ge \max\{f(x)\} - a_{i_1 i_2\cdots i_k}$, *then the monomial* $x_{i_1}\cdots x_{i_k}$ *has deception about the variable* x_{i_1}.

Proof. If $a_{i_1\cdots i_k} < 0$, suppose the monomial $x_{i_1}\cdots x_{i_k}$ has deception about the variable x_{i_1}. If a critical value of the monomial $x_{i_1}\cdots x_{i_k}$ with respect to the variable x_{i_1} is 0, according to Definition 1, the best string of *f* is included in the schema $S[(i_1,0)]$,which contradicts the condition of that the best string of *f* is 11⋯1. *If* a critical value of the monomial $x_{i_1}\cdots x_{i_k}$ with respect to the variable x_{i_1} is negative and greater than the monomial $a_{i_1\cdots i_k}$, according to Definition 1, the best string of *f* is also included in the schema $S[(i_1,0)]$,which also contradicts the condition of that the best string of *f* is 11⋯1. So If $a_{i_1\cdots i_k} < 0$, the monomial $x_{i_1}\cdots x_{i_k}$ doesn't have deception about every variable in it.

If $a_{i_1\cdots i_k} > 0$ and $partbest[i_1] < \max\{f(x)\} - a_{i_1 i_2\cdots i_k}$,suppose the monomial $x_{i_1}\cdots x_{i_k}$ has deception about the variable x_{i_1}. According to Definition 2, a critical value of the monomial $x_{i_1}\cdots x_{i_k}$ with respect to the variable x_{i_1} is 0 or positive and less than the coefficient $a_{i_1\cdots i_k}$. If a critical value of the monomial $x_{i_1}\cdots x_{i_k}$ with respect to the variable x_{i_1} is 0, Lemma 1 shows that:

$$\max\{f_-(i_1,\cdots,i_k)[(i_1,0)]\} = \max\{f_-(i_1,\cdots,i_k)[(i_1,1)]\},$$

We can get

$$\begin{aligned}&\max\{f(x)\} - a_{i_1 i_2\cdots i_k}\\ &\le \max\{\max\{f_-(i_1,\cdots,i_k)[(i_1,0)]\},\max\{f_-(i_1,\cdots,i_k)[(i_1,1)]\}\}\\ &= \max\{f_-(i_1,\cdots,i_k)[(i_1,0)]\}\\ &= partbest[i_1]\end{aligned}$$

which contradicts the condition of that $partbest[i_1] < \max\{f(x)\} - a_{i_1 i_2\cdots i_k}$. If a critical value of the monomial $x_{i_1}\cdots x_{i_k}$ with respect to the variable x_{i_1} is positive and less than the coefficient $a_{i_1\cdots i_k}$, according to Definition 1, we can get

$$\max\{f_-(i_1,\cdots,i_k)[(i_1,0)]\} \ge \max\{f_-(i_1,\cdots,i_k)[(i_1,1)]\}$$

Then

$$\begin{aligned}&\max\{f(x)\} - a_{i_1 i_2\cdots i_k}\\ &\le \max\{\max\{f_-(i_1,\cdots,i_k)[(i_1,0)]\},\max\{f_-(i_1,\cdots,i_k)[(i_1,1)]\}\}\\ &\le \max\{f_-(i_1,\cdots,i_k)[(i_1,0)]\}\\ &= partbest[i_1]\end{aligned}$$

which contradicts the condition of that $partbest[i_1] < \max\{f(x)\} - a_{i_1 i_2\cdots i_k}$. So If $a_{i_1\cdots i_k} > 0$ *and* $partbest[i_1] < \max\{f(x)\} - a_{i_1 i_2\cdots i_k}$, then the monomial $x_{i_1}\cdots x_{i_k}$ doesn't have deception about the variable x_{i_1}.

If $a_{i_1\cdots i_k} > 0$ *and* $partbest[i_1] = \max\{f(x)\} - a_{i_1 i_2\cdots i_k}$, we will prove that a critical value of the monomial $x_{i_1}\cdots x_{i_k}$ with respect to the variable x_{i_1} is 0. For arbitrary ε , $0 < \varepsilon < a_{i_1\cdots i_k}$, we get

$$\begin{aligned}&\max\{f_-(i_1,\cdots,i_k) + \varepsilon\, x_{i_1}\cdots x_{i_k}\}\\ &= \max\{f_-(i_1,\cdots,i_k) + a_{i_1\cdots i_k}\, x_{i_1}\cdots x_{i_k} + (\varepsilon - a_{i_1\cdots i_k})\, x_{i_1}\cdots x_{i_k}\}\end{aligned}$$

$$\geq \max\{f(x)\} - \max\{(a_{i_1\cdots i_k} - \varepsilon)\, x_{i_1}\cdots x_{i_k}\}$$
$$= \max\{f(x)\} - (a_{i_1\cdots i_k} - \varepsilon)$$
$$= partbest[i_1] + \varepsilon$$
$$= \max\{f_-(i_1,\cdots,i_k)[(i_1,0)]\} + \varepsilon$$

Then the best strings of $f_-(i_1,\cdots,i_k) + \varepsilon\, x_{i_1}\cdots x_{i_k}$ are included in the schema $S[(i_1,1)]$. On the other hand, since the best string of is $11\cdots1$,we get

$$\max\{f_-(i_1,\cdots,i_k)[(i_1,0)]\}$$
$$= partbest[i_1]$$
$$= \max\{f(x)\} - a_{i_1 i_2\cdots i_k}$$
$$= \max\{f_-(i_1,\cdots,i_k)[(i_1,1)]\}$$
$$\geq \max\{f_-(i_1,\cdots,i_k)[(i_1,1)]\} - \varepsilon\, x_{i_1}\cdots x_{i_k}$$

Then one of the best strings of $f_-(i_1,\cdots,i_k) - \varepsilon\, x_{i_1}\cdots x_{i_k}$ is included in the schema $S[(i_1,0)]$. According to Definition 1and 2, we get that a critical value of the monomial $x_{i_1}\cdots x_{i_k}$ with respect to the variable x_{i_1} is 0 and the monomial $x_{i_1}\cdots x_{i_k}$ has deception about the variable x_{i_1} .

If $a_{i_1\cdots i_k} > 0$ *and* $partbest[i_1] > \max\{f(x)\} - a_{i_1 i_2\cdots i_k}$, we will prove that a critical value of the monomial $x_{i_1}\cdots x_{i_k}$ with respect to the variable x_{i_1} is positive and less than the coefficient $a_{i_1\cdots i_k}$. First, since the best string of is $11\cdots1$,we get

$$\max\{f_-(i_1,\cdots,i_k)[(i_1,0)]\} > \max\{f_-(i_1,\cdots,i_k)[(i_1,1)]\}$$
$$\max\{f(x)[(i_1,1)]\} > \max\{f_-(i_1,\cdots,i_k)[(i_1,0)]\}$$

Let

$$\lambda^1 = (\max\{f_-(i_1,\cdots,i_k)[(i_1,0)]\} - \max\{f_-(i_1,\cdots,i_k)[(i_1,1)]\})/2$$
$$\lambda^2 = (\max\{f(x)[(i_1,1)]\} - \max\{f_-(i_1,\cdots,i_k)[(i_1,0)]\})/2$$

Obviously ,

$$\max\{f_-(i_1,\cdots,i_k)[(i_1,1)] + (a_{i_1\cdots i_2} - \lambda^2)x_{i_1}\cdots x_{i_k}\}$$
$$> \max\{f_-(i_1,\cdots,i_k)[(i_1,0)]\}$$
$$> \max\{f_-(i_1,\cdots,i_k)[(i_1,1)] + \lambda^1 x_{i_1}\cdots x_{i_k}\}$$

Furthermore , the function $\max\{f_-(i_1,\cdots,i_k)[(i_1,0)]\}$ is a constant and the other function $\max\{f_-(i_1,\cdots,i_k)[(i_1,1)] + \lambda\, x_{i_1}\cdots x_{i_k}\}$ is an increasing function about the variable λ . So there must exist a λ^0 $(0 < \lambda^1 \leq \lambda^0 \leq a_{i_1\cdots i_k} - \lambda^2 < a_{i_1\cdots i_k})$, , for arbitrary $\varepsilon > 0$, all the best strings of $f_-(i_1,\cdots,i_k)$ + ($\lambda^0 + \varepsilon$) $x_{i_1}\cdots x_{i_k}$ are included in the schema $S[(i_1,1)]$ and one of the best strings of $f_-(i_1,\cdots,i_k)$ + $(\lambda^0 - \varepsilon)\, x_{i_1}\cdots x_{i_k}$ is included in the schema $S[(i_1,0)]$. Therefore, by Definition 1 and 2, λ^0 is a critical value of the monomial $x_{i_1}\cdots x_{i_k}$ with respect to the variable x_{i_1} and the monomial $x_{i_1}\cdots x_{i_k}$ has deception about the variable x_{i_1} .

Until now, the theorem is proved.

4.2 A procedure for computing the deceptive degree of a function

Now we can give a procedure for computing the deceptive degree of a function at the condition of that all the value of A[k] are known and the best string of f is $11\cdots1$. The concrete steps are as follows: First , let dd represents the deceptive degree of f , set dd=0. Compute $partbest[i]$ for $i = 1,2,\cdots,n$. Then, for every $k \in \{0,1\}^n$, compute the number of variables which the corresponding monomial has deception about , if the number is larger than dd, change the value of dd by the number. In the final, the value of dd is the deceptive degree of f.

Procedure 2 describes the concrete steps in a similar computer program language.

Procedure 2

```
begin
  dd=0;
  for i←1 to n do partbest[i]←0;
  for k←0 to $2^n$-1 do
    for i←1 to n do
      if (($k_i$=0) and (A[k]>partbest[i])) do  partbest[i]←A[k];

  for k←0 to $2^n$-1 do
    if (A[k]>0) do
     begin
       flag=0;
       for i←1 to n do
         if (partbest[i] ≥ A[$2^n$-1]-A[k]) do flag←flag+1;
       if(flag>dd) dd←flag;
     end
  return(dd);
end
```

Remark: By Lemma 5, we can simplify Procedure 2. we can compute deceptive degree of monomials according to the sequence of monomial degrees from big to small. If the monomial degree of a monomial is less than the current value of dd, we needn't compute the deceptive degree of the monomial for which has no influence on the deceptive degree of the function.

5 A fast algorithm for computing the deceptive degree of a function

Section 3 and 4 have prepared for a fast algorithm to compute the deceptive degree of a function which has only one best string under the condition of that the function value of every input is known. Now we describe the fast algorithm.

Algorithm 1:
Input: the function value of every input of a function $f(x)$.
Output: the deceptive degree of the function $f(x)$.
Step 1: verify whether $f(x)$ has only best string and the best string is $11\cdots1$. If $f(x)$ has more than one best string, we have to terminate the algorithm. If the best string of $f(x)$ is $C \in \{0,1\}^n$, which C isn't equal to $11\cdots1$ and $\overline{C} \oplus C = 11\cdots1$, by Definition 4 and Lemma 4, we will compute the deceptive degree of $f(x \oplus \overline{C})$ instead of the deceptive degree of $f(x)$ in the continuing part of the algorithm.
Step2: get polynomial coefficients A[k] of the corresponding function by using Procedure 1.
Step3: get the deceptive degree of the corresponding function by using Procedure2.

Since the space complexity of the algorithm is mainly for storing function values of f, polynomial coefficients A[k] and other intermediate results, the algorithm needs $O(2^n)$ storage. On the other hand, the time complexity of the algorithm can be get as follows: Since Step 1 needs $O(2^n)$ time, Step 2 needs $O(3^n)$ time and Step 2 needs $O(2^n)$ time, so the algorithm needs $O(3^n)$ time.

6 Summary

In [1], we presented a new definition for the deceptive degree of a function, but it is very complicated to compute the deceptive degree of a function according to the new definition. In this paper we present a fast algorithm for computing the deceptive degree of a function. We discuss theoretical foundations of the fast algorithm and how to get the polynomial representation of a function quickly under the condition of that the function value of every input is known. We prove a fast decision theorem of whether a monomial has deception about a variable in it, which makes computing the deceptive degree of a function easier. In the final, we describes the fast algorithm and analyses the complexity of the algorithm.

The fast algorithm is helpful to discussing the usefulness of the new definition of deception. Whether the new definition can play an important role in the connection between the GA performance and the structure of a given function deserves further consideration. The relation between the new definition and the old ones should also be taken into account for future research.

References

1. LI Yunqiang. The Deceptive Degree of the Objective Function. Foundations Of Genetic Algorithms 2005 (FOGA'05), Aizu-Wakamatsu City, Japan ,January 2005. Revised Selected Paper. Lecture Notes in Computer Science 3469:300-314. Springer Verlag, 2005.
2. A. D. Bethke.: Genetic Algorithm as Function Optimizations. Doctoral Dissertation, University of Michigan. Dissertation abstracts International 41(9), 3503B. University Microfilms No.8106101. (1981) .
3. Goldberg, D. E.: Simple genetic algorithms and the minimal deceptive problem. In L. Davis(Ed.), Genetic algorithms and simulated annealing (pp.74-88). London: Pitman. 1987.
4. Y. Davidor.: Epistasis variance: a viewpoint on GA-hardness. In G.J.E. Rawlins, editor, Foundations of Genetic Algorithms, pages 23-35. Morgan Kaufmann, 1991.
5. Liepins, G. E. and Vose, M. D.: Representational Issues in Genetic Optimization. Journal of Experimental and Theoretical Artificial Intelligence. 1990(2):101-115.
6. Whitley, L D.: Fundamental Principles of Deception in Genetic Search. In G. Rawlins (ED.). Foundations of Genetic Algorithms. San Mateo, CA:Morgan Kaufmann(1991).
7. J. J. Grefenstette.: Deception considered harmful. in: L. Darrell Whitley(Ed.), FOGA-2, Morgan Kaufmann, 1993, pp. 75-91.
8. J. J. Grefenstette and J. E. Baker.: How Genetic Algorithms Work: A Critical Look at Implicit Parallelism. Proceedings of the Third International Conference on Genetic Algorithms, J. D. Schaffer, Editor, San Mateo, CA, Morgan Kaufmann, 1989.
9. M. Mitchell, S. Forrest, and J. H. Holland.: The Royal Road Genetic Algorithms: Fitness Landscapes and GA Performance. Proceedings of the First European Conference and
Artificial Life, Cambridge, MA, MIT Press/Branford Books, 1992.
10. Stephanie Forrest and Melanie Mitchell.: What Makes a Problem Hard for Genetic Algorithm? Some Anomalous Results in the Explanation. Machine Learning, 13, pages
285-319, 1993.
11. C.R.Reeves and C. C. Wright .: Epistasis in genetic algorithms: an experimental design perspective. In L. J. Eshelman (Ed.) (1995)Proceedings of the 6 th International Conference on Genetic Algorithms, Morgan Kaufmann, San Mateo, CA,217-224.
12. B. Leblanc and E. Lutton.: Bitwise regularity and GA-hardness. ICEC 98, May 5
Anchorage, Alaska, 1998.
13. C.chryssomalakeos and C.R.Stephens. What Basis for Genetic Dynamics? Acta.Phys.Slov. 2004.
14. C.R.Stephens. The renormalization group and the dynamics of genetic systems. Acta.Phys.Slov. 52:515-524,2003.
15. Marc Schoenauer. Evolutionary Algorithms for Parameter Optimization in theory and Practice. Lectures of 2004 International Workshop on Nature Inspired Computation and Applications. Oct.25-29,2004. Hefei,China.
16. C.R.Stephens, and A.Zamora. EC theory: A unified viewpoint. In Erick Cantu.Paz; editor, GECCO2003. pp1394-1402. Berlin, Germany,2003.