On the Easiest and Hardest Fitness Functions

Jun He, *Member, IEEE,* Tianshi Chen, and Xin Yao, *Fellow, IEEE*

Abstract—The hardness of fitness functions is an important research topic in the field of evolutionary computation. In theory, this paper can help with understanding the ability of evolutionary algorithms (EAs). In practice, this paper may provide a guideline to the design of benchmarks. The aim of this paper is to answer the following research questions. Given a fitness function class, which functions are the easiest with respect to an EA? Which are the hardest? How are these functions constructed? This paper provides theoretical answers to these questions. The easiest and hardest fitness functions are constructed for an elitist (1 + 1) EA to maximize a class of fitness functions with the same optima. It is demonstrated that the unimodal functions are the easiest and deceptive functions are the hardest in terms of the time-based fitness landscape. This paper also reveals that in a fitness function class, the easiest function to one algorithm may become the hardest to another algorithm, and vice versa.

Index Terms—Algorithm analysis, benchmark design, evolutionary algorithm, fitness landscape, problem difficulty.

I. Introduction

WHICH fitness functions are easy for an evolutionary algorithm (EA) and which are not? This is an important research topic in the field of evolutionary computation. In theory, the study of the hardness of fitness functions can help understanding the ability of EAs. In practice, this paper may provide a guideline to the design of benchmarks. Answers to the above questions vary as the scope of fitness functions changes from all possible functions to a single function.

The first scenario is to consider all possible fitness functions. In this case, No Free Lunch theorems [1], [2] have answered the question. The theorems claim that the performance of any two EAs are equivalent in terms of average performance.

The second scenario is to consider a class of fitness functions with the same features, such as unimodal functions versus multimodal functions, or deceptive functions versus nondeceptive functions. However, a multimodal function may be easy to solve [3]. A unimodal function may be difficult for certain EAs but easy for others [4]. A nondeceptive function may be difficult to an EA [5], and a deceptive function may be

easy [6]. Few features are available to distinguish whether a function class is easy or hard for an EA.

The third scenario is to consider a single fitness function. A popular approach is to develop a statistic measure to predict the hardness of a fitness function, such as fitness-distance correlation [7], fitness variance [8], and epistasis variance [9]. Unfortunately, it is intractable to design a measure that can predict the hardness of a function efficiently [10], [11].

Different from the above three scenarios, an alternative scenario is considered in this paper. Given an EA and a class of fitness functions with the same optima, which function is the hardest within the class? Which is the easiest? And how do we construct them? Here, the easiest function is referred to a function on which the runtime of the EA is the shortest; and the hardest is a function on which the runtime of the EA is the longest. Both are compared with other functions in the same class. These questions have never rigorously been answered before.

Our research aims at understanding the hardest and easiest fitness functions within a function class, and helping design benchmarks. The set of benchmarks usually include several typical fitness functions, for example, easy, hard and "averagely hard" functions in the class. An EA has the best performance on the easiest function, and the worst performance on the hardest function. We will focus on these two extreme cases in this paper.

The paper is organized as follows. Section II describes related work. Section III defines the easiest and hardest fitness functions, and establishes criteria of determining whether a function is the easiest or the hardest. Section IV constructs the easiest and hardest functions. Section V discusses the mutual transformation between the easiest and hardest functions. Section VI concludes the paper.

II. Related Work

The hardness of fitness functions (or called problem difficulty) has been studied over two decades. Normally a fitness function is said to be easy to an EA if the runtime is polynomial on the function or hard if the runtime is exponential. How to characterize which fitness functions are easy or hard was thought to be a major challenge [10].

One approach is to link features of a fitness landscape to the hardness of fitness functions. Several features have been investigated, for example, isolation, deception and multimodality, ruggedness, and neutrality. A fitness landscape with isolation is hard for EAs, but other characteristics may not be related too much to the hardness of fitness functions [10]. A fully nondeceptive function may be difficult for an EA [5] but some deceptive functions can be solved easily by an EA [6].

Some multimodal functions may be easy to solve [3], but the unimodal function like the "long path" problem [3] could be difficult for certain EAs [4]. Few features are universally useful to distinguish between hard and easy fitness functions.

Another approach is to predict the hardness of a fitness function through a statistic measure. Many measures are proposed, for example, fitness-distance correlation [7], correlation length and operator correlation [12], fitness variance [8], and epistasis variance [9]. Nevertheless, to compute the exact value of such measures usually is exponential in the problem size due to the fact that the search space is exponentially large [10], [11], [13]. Inherent flaws also exist in the common hardness measures such as epistasis variance, fitness-distance correlation, and epistasis correlation [14].

An alternative theoretical approach is based on fitness levels. Hard fitness functions are classified into two types: "wide gap" problems and "long path" problems [15], [16]. For the "wide gap" type, the EA is trapped at a fitness level, because there is a wide gap between that fitness level and higher fitness levels. For the "long-path" type, the EA has to take a long path to reach an optimum. The behavior of EAs on these two problems are different [17], [18].

The research in the current paper is totally different from previous work. The hardest and easiest functions are compared with other fitness functions within the same function class. The hardest function are not relevant to exponential runtime and the easiest fitness functions are not relevant to polynomial runtime. For some function class, an EA only needs polynomial time on the hardest function. For some other function class, an EA may take exponential time on the easiest function.

Our paper is also different from No Free Lunch theorems [1], [2], which state that any two EAs are equivalent when their performance is averaged across all possible fitness functions. We do not intend to investigate the easiest and hardest functions among all possible fitness functions, instead only within a class of fitness functions with the same optima.

III. EASIEST AND HARDEST FITNESS FUNCTIONS

In this section, we define the easiest and hardest fitness functions in a function class and establish the criteria to determine whether a function is the easiest or hardest.

A. Definition of Easiest and Hardest Fitness Functions

Consider the problem of maximizing a class of fitness functions with the same optima. An instance of the problem is to maximize a fitness function $f(x)$

$$\max\{f(x); x \in S\} \tag{1}$$

where S is a finite set. The optimal set is denoted by S_{opt} and the nonoptimal set by S_{non}. Without loss of generality, the function $f(x)$ takes $L + 1$ finite values $f_0 > f_1 > \cdots > f_L$ (called fitness levels). Corresponding to fitness levels, the set S is decomposed into $L + 1$ subsets

$$S_l := \{x \mid f(x) = f_l\}, l = 0, 1, \ldots, L.$$

For simplicity of analysis, we only investigate strictly elitist $(1 + 1)$ EAs. Using strictly elitist selection, the parent is

---

Algorithm 1 $(1 + 1)$ Elitist Evolutionary Algorithm

1: **input**: fitness function $f(x)$;
2: generate a solution at random and denote it by ϕ_0;
3: generation counter $t \leftarrow 0$;
4: **while** the maximum value of $f(x)$ is not found **do**
5:     child $\phi_{t.m} \leftarrow$ is mutated from parent ϕ_t;
6:     **if** $f(\phi_{t.m}) > f(\phi_t)$ **then**
7:        next generation parent $\phi_{t+1} \leftarrow \phi_{t.m}$;
8:     **else**
9:        next generation parent $\phi_{t+1} \leftarrow \phi_t$;
10:     **end if**
11:     $t \leftarrow t + 1$;
12: **end while**
13: **output**: the maximal value of $f(x)$.

---

replaced by the child only when the child is fitter. Therefore, the best found solution is always preserved. In the EAs, mutation is independent of the fitness function. Both mutation and selection operators are time invariant (i.e., static). The procedure of such an elitist $(1 + 1)$ EA is described in Algorithm 1.

Let $G(x)$ denote the expected number of generations for an EA to find an optimal solution for the first time when starting at x (called expected hitting time). In $(1 + 1)$ EAs, $G(x)$ also represents the expected number of fitness evaluations (called expected runtime). In this paper, we restrict our discussion to those EAs whose expected runtime is finite (convergent).

Definition 1: Given an EA for maximizing a class of fitness functions with the same optima, a function $f(x)$ in the class is said to be the easiest to the EA if starting from any initial point, the runtime of the EA for maximizing $f(x)$ is no more than the runtime for maximizing any fitness function $g(x)$ in the class when starting from the same initial point. A function $f(x)$ in the class is said to be the hardest to the EA if starting from any initial point, the runtime of the EA for maximizing $f(x)$ is no less than the runtime for maximizing any fitness function $g(x)$ in the class when starting from the same initial point.

The definition of the easiest and hardest functions is based on a point-by-point comparison of the runtime of the EA on two fitness functions. It is irrelevant to polynomial or exponential runtime. The easiest and hardest functions are not unique. This will be demonstrated in Section IV-C.

B. Criterion for Determining Easiest Function

Before we establish the criterion, we apply drift analysis to the random sequence $\{\phi_t, t = 0, 1, \cdots\}$ and draw several preliminary results. Notice that each generation of the $(1 + 1)$ EA consists of two steps: mutation and selection

$$\phi_t \overset{\text{mutation}}{\longrightarrow} \phi_{t.m} \text{ with } \phi_t \overset{\text{selection}}{\longrightarrow} \phi_{t+1}.$$

The mutation operator is a transition from ϕ_t to $\phi_{t.m}$, whose transition probabilities are represented by

$$P^{[m]}(x, y) := P(\phi_{t.m} = y \mid \phi_t = x), x, y \in S. \tag{2}$$

Here, ϕ is a random variable and x its value.

The selection operator is another transition from ϕ_t and $\phi_{t.m}$ to ϕ_{t+1}, whose transition probabilities are represented by

$$P^{[s]}(x, y; z) := P(\phi_{t+1} = z \mid \phi_t = x, \phi_{t.m} = y), x, y, z \in S.$$

The tth generation is a transition from ϕ_t to ϕ_{t+1}, whose transition probabilities are represented by

$$P(x, z) := P(\phi_{t+1} = z \mid \phi_t = x). \qquad (3)$$

In drift analysis, a function $d(x)$ is called a drift function if it is nonnegative at any point and equals to 0 at any optimum. Given a drift function $d(x)$, drift represents the progress rate of moving toward the optima per generation. Drift at point x is defined by

$$\Delta(x) := \sum_{y \in S} P(x, y)(d(x) - d(y)).$$

Define positive drift $\Delta^+(x)$ and negative drift $\Delta^-(x)$ as

$$\Delta^+(x) = \sum_{y: d(x) > d(y)} P(x, y)(d(x) - d(y))$$

$$\Delta^-(x) = \sum_{y: d(x) < d(y)} P(x, y)(d(x) - d(y)).$$

Then the drift $\Delta(x) = \Delta^+(x) + \Delta^-(x)$.

Using drift analysis [19], we obtain the following preliminary results.

Lemma 1: [19, Lemma 1] If the drift satisfies that $\Delta(x) \geq 1$ for any nonoptimal point x, then the expected runtime satisfies that $G(x) \leq d(x)$ for any point x.

Lemma 2: [19, Lemma 2] If the drift satisfies that $\Delta(x) \leq 1$ for any nonoptimal point x, then the expected runtime satisfies that $G(x) \geq d(x)$ for any point x.

Lemma 3: [19, Lemma 3] Let the drift function $d(x) = G(x)$, then the drift satisfies $\Delta(x) = 1$ for any nonoptimal point x.

Furthermore, the runtime of an elitist (1 + 1) EA can be explicitly expressed in transition probabilities.

Lemma 4: [16, Theorem 4] For any elitist (1 + 1) EA, its expected runtime is given by

$$G(x) = \begin{cases} 0, & x = S_0. \\ \dfrac{1 + \sum_{k=0}^{l-1} \sum_{y \in S_k} P(x,y) G(y)}{\sum_{k=0}^{l-1} \sum_{y \in S_k} P(x,y)}, & x \in S_l, l > 0. \end{cases}$$

Using the above lemmas, we establish a criterion of determining whether a fitness function is the easiest to an elitist (1 + 1) EA.

Theorem 1: Given an elitist (1 + 1) EA, and a class of fitness functions with the same optima, let $G_f(x)$ denote the runtime of the (1 + 1) EA for maximizing $f(x)$. If the following monotonically decreasing condition holds.

1) For any two points x and y such that $G_f(x) < G_f(y)$, it has $f(x) > f(y)$.

Then $f(x)$ is the easiest in the fitness function class.

Proof: Let $g(x)$ be a fitness function in the function class. $\{\phi_t, t = 1, 2, \cdots\}$ denotes the sequence for maximizing $f(x)$, and $\{\psi_t, t = 1, 2, \cdots\}$ the sequence for maximizing $g(x)$. Let $G_g(x)$ denote the runtime of the (1 + 1) EA for maximizing $g(x)$.

Since our objective is to show the expected runtime on $f(x)$ is no more than the runtime on any other function, we take the runtime on $f(x)$ as the drift function: $d(x) = G_f(x)$. This plays a crucial role in our analysis.

For the sequence $\{\phi_t\}$, denote the drift at point x by $\Delta_\phi(x)$. For the sequence $\{\psi_t\}$, denote the drift at point x by $\Delta_\psi(x)$. The subscripts ϕ and ψ are used to distinguish between the two sequences $\{\phi_t\}$ and $\{\psi_t\}$.

Notice that $d(x) = G_f(x)$, then we apply Lemma 3 and get that for any nonoptimal point x, drift

$$\Delta_\phi(x) = 1. \qquad (4)$$

The rest of proof is based on the idea: first, we prove the drift $\Delta_\psi(x) \leq 1$ for the sequence $\{\psi_t\}$, and then draw the derived conclusion using Lemma 2.

1) First we compare the negative drift of the two sequences. In the case of negative drift, we consider two points x and y such that $d(x) < d(y)$ (i.e., $G_f(x) < G_f(y)$). According to the monotonically decreasing condition, $f(x) > f(y)$.

For the sequence $\{\phi_t\}$, y is never accepted due to elitist section, which leads to $P_\phi(x, y) = 0$. Thus, for the sequence $\{\phi_t\}$, there is no negative drift

$$\Delta_\phi^-(x) = 0.$$

For the sequence $\{\psi_t\}$, there exist two cases: 1) $g(x) < g(y)$ and 2) $g(x) \geq g(y)$. In the case of $g(x) < g(y)$, y will be accepted, which implies $P_\psi(x, y) \geq 0$. Thus, there exists negative drift for the sequence $\{\psi_t\}$

$$\Delta_\psi^-(x) \leq 0.$$

Comparing the negative drift of these two sequences, we get

$$\Delta_\psi^-(x) \leq \Delta_\phi^-(x). \qquad (5)$$

2) Secondly, we compare the positive drift of the two sequences. In the case of positive drift, we consider two points x and y such that $d(x) > d(y)$. If y is not an optimum, then according to the monotonically decreasing condition, $f(x) < f(y)$. If y is an optimum, then naturally $f(x) < f(y)$.

For the sequence $\{\phi_t\}$, if such a y has been mutated from x, then y is always accepted due to elitist selection. Thus

$$P_\phi(x, y) = P^{[m]}(x, y).$$

For the sequence $\{\psi_t\}$, there exist two cases: 1) $g(x) < g(y)$ or 2) $g(x) \geq g(y)$. In the case of $g(x) < g(y)$, according to elitist section, y is always accepted. Thus

$$P_\psi(x, y) = P^{[m]}(x, y).$$

In the case of $g(x) \geq g(y)$, according to elitist section, y will not be accepted. The transition probability $P_\psi(x, y) = 0$. Then we get that $P_\phi(x, y) \geq P_\psi(x, y)$. Hence

$$\sum_{y: d(x) > d(y)} P_\psi(x, y)(d(x) - d(y))$$

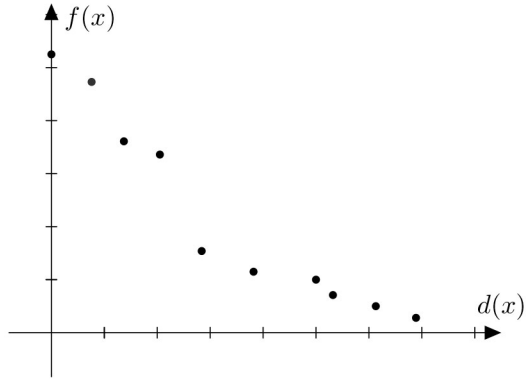$$\leq \sum_{y: d(x) > d(y)} P_\phi(x, y)(d(x) - d(y)).$$

Fig. 1.   Unimodal time-based fitness landscape. The x-axis is the runtime: $d(x) = G(x)$. The y-axis is the fitness function. The origin represents the optimum.

So the positive drift of the two sequences satisfies

$$\Delta_\psi^+(x) \le \Delta_\phi^+(x). \tag{6}$$

Merging (5) and (6) and using (4), we know that the total drift of the two sequences satisfies

$$\Delta_\psi(x) \le \Delta_\phi(x) = 1.$$

Applying Lemma 2, we see the expected runtime on $g(x)$ satisfies

$$G_g(x) \ge d(x) = G_f(x)$$

then we finish the proof.   ∎

Now we give an intuitive explanation of the above theorem. The monotonically decreasing condition means the function is unimodal in terms of the time-based fitness landscape and Theorem 1 asserts that a unimodal function is always the easiest. In the following we explain this in detail.

In a time-based fitness landscape, runtime $G(x)$ is regarded as the distance $d(x)$ between a point x and the optimum. It is completely different from a neighborhood-based distance such as the Hamming distance. Time is seldom used as a distance measure in evolutionary computation but popular in our real life. Taking runtime as the distance, we visualize the monotonically decreasing condition.

1) For any two points x and y such that $d(x) < d(y)$, it has $f(x) > f(y)$.

Using a time-based fitness landscape (see Fig. 1), where the x-axis is the runtime and the y-axis is the fitness, and the origin represents the set of optima with $d(x) = 0$.

The landscape is unimodal: the function $f(x)$ has exactly one optimum. In contrast, any unimodal function defined in the 2-D time-based fitness landscape will satisfy the monotonically decreasing condition. The unimodal property implies that no negative drift exists in an elitist EA. Thus, the EA always moves toward the optimum. This makes the unimodal time-based fitness landscapes the easiest to the EA.

The theorem only states that a unimodal time-based fitness landscape is the easiest. Nevertheless, this assertion could not be established if using a neighborhood-based distance such as the Hamming distance. A unimodal function in the context

of a neighborhood-based fitness landscape is not always the easiest.

C. Criterion of Determining Hardest Function

In a similar way, we establish a criterion of determining whether a fitness function is the hardest to an elitist (1 + 1) EA. It is similar to Theorem 1. The monotonically decreasing condition is replaced by the monotonically increasing condition.

Theorem 2: Given an elitist (1 + 1) EA, and a class of fitness function with the same optima, let $G_f(x)$ denote the expected runtime for maximizing $f(x)$. If the following monotonically increasing condition holds.

1) For any two nonoptimal points x and y such that $G_f(x) < G_f(y)$, it has $f(x) < f(y)$.

Then $f(x)$ is the hardest in the class.

Proof: The proof is similar to that of Theorem 1 but with several changes.

Let $g(x)$ be a fitness function in the function class. $\{\phi_t, t = 1, 2, \cdots\}$ denotes the sequence for maximizing $f(x)$, and $\{\psi_t, t = 1, 2, \cdots\}$ the sequence for maximizing $g(x)$. $G_g(x)$ denotes the runtime of the (1 + 1) EA for maximizing $g(x)$. We take the runtime on $f(x)$ as the drift function: $d(x) = G_f(x)$.

For the sequence $\{\phi_t\}$, notice that $d(x) = G_f(x)$, then we apply Lemma 3 and get for any nonoptimal point x

$$\Delta_\phi(x) = 1. \tag{7}$$

First we compare the negative drift of the two sequences. We consider two nonoptimal points x and y such that $d(x) < d(y)$ (i.e., $G_f(x) < G_f(y)$). According to the monotonically increasing condition, $f(x) < f(y)$.

For the sequence $\{\phi_t\}$, if such a y has been mutated from x, then y is always accepted due to elitist selection. Thus

$$P_\phi(x, y) = P^{[m]}(x, y).$$

For the sequence $\{\psi_t\}$, there exist two cases: 1) $g(x) < g(y)$ and 2) or $g(x) \ge g(y)$. In the case of $g(x) < g(y)$, if such a y has been mutated from x, then y is always accepted due to elitist selection. Thus

$$P_\psi(x, y) = P^{[m]}(x, y).$$

In the case of $g(x) \ge g(y)$, according to elitist section, y will not be accepted. The probability $P_\psi(x, y)$ equals to

$$P_\psi(x, y) = 0.$$

Then we get that $P_\phi(x, y) \ge P_\psi(x, y)$. Hence

$$\sum_{y:d(x)<d(y)} P_\psi(x, y)(d(x) - d(y))$$
$$\ge \sum_{y:d(x)<d(y)} P_\phi(x, y)(d(x) - d(y)). \tag{8}$$

Equivalently the negative drift of the two sequences satisfies

$$\Delta_\psi^-(x) \ge \Delta_\phi^-(x). \tag{9}$$

Second, we compare the positive drift of the two sequences. We consider two points x and y such that $d(x) > d(y)$, where y could be either an optimum or not.

First consider y an optimum. For the sequence $\{\phi_t\}$, if such a y has been mutated from x, then y is always accepted due to elitist selection. Thus

$$P_\phi(x, y) = P^{[m]}(x, y).$$

Similarly for the sequence $\{\psi_t\}$, y is always accepted due to elitist selection. Thus

$$P_\psi(x, y) = P^{[m]}(x, y).$$

Then we get

$$P_\psi(x, y) = P_\phi(x, y). \tag{10}$$

Then consider y not an optimum. According to the monotonically increasing condition, $f(x) > f(y)$ if y is not an optimum.

For the sequence $\{\phi_t\}$, y is never accepted due to elitist section, which leads to

$$P_\phi(x, y) = 0.$$

For the sequence $\{\psi_t\}$, even if $f(x) > f(y)$, it is still possible that $g(x) < g(y)$. So y may be accepted. This means

$$P_\psi(x, y) \geq 0.$$

Thus, we have

$$P_\psi(x, y) \geq P_\phi(x, y). \tag{11}$$

Combining (10) and (11), we have for any y

$$P_\psi(x, y) \geq P_\phi(x, y).$$

Then

$$\sum_{y:d(x)>d(y)} P_\psi(x, y)(d(x) - d(y))$$
$$\geq \sum_{y:d(x)>d(y)} P_\phi(x, y)(d(x) - d(y)). \tag{12}$$

Equivalently the positive drift of the two sequences satisfies

$$\Delta_\psi^+(x) \geq \Delta_\phi^+(x). \tag{13}$$

Merging (9) and (13) and using (7), we draw that the total drift of the two sequences satisfies

$$\Delta_\psi(x) \geq \Delta_\phi(x) = 1.$$

It follows from Lemma 1 that for any nonoptimal point x

$$G_g(x) \leq d(x) = G_f(x)$$

then we finish the proof. ∎

An intuitive explanation of the above theorem is that the monotonically increasing condition means the function is deceptive and Theorem 2 states a deceptive function is always the hardest. Let's demonstrate this using the time-based fitness landscape. Still taking runtime $G(x)$ as the distance $d(x)$ to the optima, we visualize the monotonically increasing condition.

1) For any two nonoptimal points x and y such that $d(x) < d(y)$, it has $f(x) < f(y)$.

Using a time-based fitness landscape (see Fig. 2). The landscape is deceptive: the closer a point is to the origin, the lower
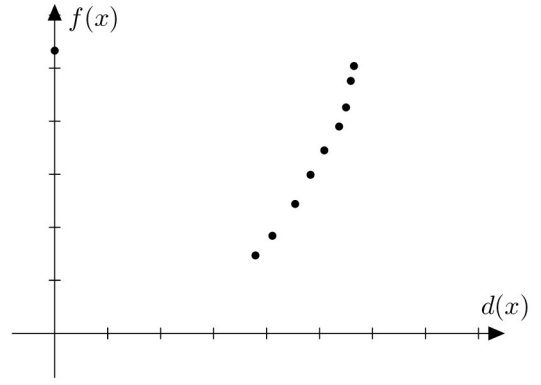

Fig. 2. Deceptive time-based fitness landscape. The x-axis is the runtime: $d(x) = G(x)$. The y-axis is the fitness function. The origin represents the optimum.

its fitness is. The deceptive time-based fitness landscape is the hardest.

When using a neighborhood-based distance, it is impossible to establish a similar result under a similar condition. A deceptive function in the context of the neighborhood-based fitness landscape is not always the hardest.

Note: The above unimodal and deceptive time-based fitness landscapes are different from the easy and hard fitness landscapes described in [20], which are classified by polynomial or exponential hitting time.

D. Case Study: 0–1 Knapsack Problem

We give two simple examples to show the application of the above theorems. The examples come from the 0–1 knapsack problem. We will not consider all instances of the 0–1 knapsack problem. Instead, we focus on an instance class.

Example 1: Consider an instance class of the 0–1 knapsack problem described as

$$\begin{aligned} \text{maximize } \quad & f(x) = \sum_{i=1}^n v_i x_i \\ \text{subject to } \quad & \sum_{i=1}^n w_i x_i \leq C \end{aligned}$$

where $v_i > 0$ is the value of item i, $w_i > 0$ its weight, and C the knapsack capacity. The value of items satisfies $v_1 > v_2 + \cdots + v_n$, and the weight of items satisfies $w_1 > w_2 + \cdots + w_n$, the knapsack capacity $C = w_1$. A solution is represented by a binary string $x = (x_1 \cdots x_n)$. The unique optimum is $(10 \cdots 0)$, denoted by x^*.

An elitist $(1 + 1)$ EA using bitwise mutation is applied to the problem.

1) EA($1/n$). Flip each bit independently with flipping probability $1/n$.

For simplicity of analysis, we adopt the simplest approach to handle the constraint: reject any infeasible solution during selection.

Let's investigate a special instance in the class: $v_1 = n$ and $v_2 = \cdots = v_n = 1$; $w_1 = n$ and $w_2 = \cdots = w_n = 1$. Notice that the global optimum is $x^* := (10 \cdots 0)$ and the local optimum is $(01 \cdots 1)$. It is a deceptive function. We can prove the monotonically increasing condition holds. We give an outline

of the proof but omit its detailed calculation. Corresponding to fitness level f_l, the subset of feasible solutions

$$S_0 = \{x^*\}, \text{ and } f_0 = n$$
$$S_l = \{x \mid h(x, x^*) = l\}, \text{ and } f_l = l - 1, \text{ for } l > 0$$

where $h(x, y)$ is the Hamming distance between x and y.

According to Lemma 4, the expected runtime of EA$(1/n)$ is given by the following recurrence relation: $G(x) = 0$ for $x \in S_0$ and

$$G(x) = \frac{1 + \sum_{k=0}^{l-1} \sum_{y \in S_k} P(x, y) G(y)}{\sum_{k=0}^{l-1} \sum_{y \in S_k} P(x, y)}, \quad x \in S_l$$

where

$$P(x, y) = P^{[m]}(x, y) = \left(1 - \frac{1}{n}\right)^{n - h(x,y)} \left(\frac{1}{n}\right)^{h(x,y)}.$$

Then the monotonically increasing condition holds

$$G(x) < G(y) \Longrightarrow f(x) < f(y), \quad x, y \in S_{\text{non}}.$$

Applying Theorem 2, we know the fitness function related to this instance is the hardest in the class.

Example 2: Consider an instance class of the 0–1 knapsack problem. The knapsack capacity C is enough large such that $C \geq w_1 + \cdots + w_n$. The unique optimum is $\vec{1} := (1 \cdots 1)$. This function class is equivalent to linear functions. We apply EA$(1/n)$ to the problem.

Let's investigate a special instance in the class: $v_1 = \cdots = v_n = 1$. Its fitness function is equivalent to the OneMax function, so that it is easy. We prove the OneMax function is the easiest through verifying the monotonically decreasing condition. We give an outline of the proof. Corresponding to fitness level f_l where $l = 0, \ldots, n$, the subset

$$S_l = \{x \mid h(x, \vec{1}) = l\}, \text{ and } f_l = n - l$$

where $h(x, y)$ is the Hamming distance between x and y.

According to Lemma 4, the expected runtime of EA$(1/n)$ is given by the following recurrence relation: $G(x) = 0$ for $x \in S_0$ and for $l > 0$

$$G(x) = \frac{1 + \sum_{k=0}^{l-1} \sum_{y \in S_k} P(x, y) G(y)}{\sum_{k=0}^{l-1} \sum_{y \in S_k} P(x, y)}, \quad x \in S_l$$

where

$$P(x, y) = P^{[m]}(x, y) = \left(1 - \frac{1}{n}\right)^{n - h(x,y)} \left(\frac{1}{n}\right)^{h(x,y)}.$$

Then the monotonically decreasing condition holds

$$G(x) < G(y) \Longrightarrow f(x) > f(y), \, x, y \in S.$$

Applying Theorem 1, we get the OneMax function is the easiest among all linear functions.

Note: The monotonically increasing condition is a sufficient condition for a fitness function being the hardest, but not necessary. The same is true to the monotonically decreasing condition of the easiest functions. The reason is trivial: consider a function class only includes one function, then the function will be both the easiest and hardest in the class, regardless of the monotonically increasing or decreasing condition.

IV. CONSTRUCTION OF EASIEST AND HARDEST FITNESS FUNCTIONS TO AN EA

In this section, we construct unimodal functions (the easiest) and deceptive functions (the hardest), respectively, to any given elitist (1 + 1) EA.

A. Construction of Easiest Fitness Functions

Given a class consisting of all fitness functions with the same optima on a finite set S, consider an elitist (1 + 1) EA for maximizing a fitness function in the class. We construct the easiest function $f(x)$ to the EA as follows.

1) Let $S_0 = S_{\text{opt}}$. For any $x \in S_0$, define $G'(x) = 0$.
2) Suppose that the subsets $S_0, \ldots, S_{l-1}$ are given and $G'(x)$ has been defined on these subsets. Let S_l be the set consisting of all points such that

$$\arg \min_{x \in S \setminus \cup_{k=0}^{l-1} S_k} \frac{1 + \sum_{k=0}^{l-1} \sum_{y \in S_k} P^{[m]}(x, y) G'(y)}{\sum_{k=0}^{l-1} \sum_{y \in S_k} P^{[m]}(x, y)}.$$

For any $x \in S_l$, define

$$G'(x) = \frac{1 + \sum_{k=0}^{l-1} \sum_{y \in S_k} P^{[m]}(x, y) G'(y)}{\sum_{k=0}^{l-1} \sum_{y \in S_k} P^{[m]}(x, y)}. \quad (14)$$

The value of $G'(x)$ is the same for any point x in the same subset.
3) Repeat the above step until any point is covered by a subset. Then there exists some integer $L > 0$ and $S = \cup_{k=0}^{L} S_k$.
4) Choose $L + 1$ numbers $f_0, \ldots, f_L$ such that $f_0 > \cdots > f_L$. Set a fitness function $f(x)$ as follows: $f(x) = f_k$, for $x \in S_k$.

The following theorem shows that the fitness function constructed above is the easiest to the EA. The proof is a direct application of the monotonically decreasing condition.

Theorem 3: $f(x)$ is the easiest function in the function class with respect to the EA.

Proof: We show that $G'(x)$ equals to the expected runtime $G(x)$.

According to Lemma 4, the expected runtime $G(x) = 0$ for $x \in S_0$ and for $l > 0$

$$G(x) = \frac{1 + \sum_{k=0}^{l-1} \sum_{y \in S_k} P(x,y) G(y)}{\sum_{k=0}^{l-1} \sum_{y \in S_k} P(x,y)}, \quad x \in S_l. \quad (15)$$

For any $x \in S_k$ and $y \in S_l$ where $k > l$, since $f(x) = f_k < f(y) = f_l$ and the EA adopts elitist selection, y is always accepted if it has been generated via mutation. Thus, the transition probability $P(x, y)$ equals $P^{[m]}(x, y)$. Equation (15) equals

$$G(x) = \frac{1 + \sum_{k=0}^{l-1} \sum_{y \in S_k} P^{[m]}(x,y) G(y)}{\sum_{k=0}^{l-1} \sum_{y \in S_k} P^{[m]}(x,y)}, \quad x \in S_l. \quad (16)$$

Comparing it with (14), $G(x)$ and $G'(x)$ are identical. We prove the monotonically decreasing condition. First we prove an inequality

$$G(x) > G(y), \quad x \in S_{l+1}, y \in S_l. \quad (17)$$

We prove it by induction. For any $x \in S_1$, $y \in S_0$, it is trivial that $G(x) > G(y) = 0$. Suppose that for any $x \in S_l$, $y \in S_{l-1}$, it holds $G(x) > G(y)$. We prove that for any $x \in S_{l+1}$, $y \in S_l$, it holds $G(x) > G(y)$.

Since $y \in S_l$, from the construction, we know that

$$G(y) = \min_{w \in S \backslash \cup_{k=0}^{l-1} S_k} \frac{1 + \sum_{k=0}^{l-1} \sum_{z \in S_k} P^{[m]}(w, z) G(z)}{\sum_{k=0}^{l-1} \sum_{z \in S_k} P^{[m]}(w, z)}.$$

Let $w = x$, then we get

$$G(y) \leq \frac{1 + \sum_{k=0}^{l-1} \sum_{x \in S_k} P^{[m]}(x, z) G(z)}{\sum_{k=0}^{l-1} \sum_{z \in S_k} P^{[m]}(x, z)}.$$

Equivalently

$$G(y) \sum_{k=0}^{l-1} \sum_{z \in S_k} P^{[m]}(x, z) \leq 1 + \sum_{k=0}^{l-1} \sum_{z \in S_k} P^{[m]}(x, z) G(z).$$

We add the term $\sum_{z \in S_l} P^{[m]}(x, z) G(z)$ to both sides. Notice that $G(z) = G(y)$ for $z \in S_l$. As to the left-hand side, we replace the factor $G(z)$ by $G(y)$ and move it outside of the summation. Then we get

$$G(y) \sum_{k=0}^{l} \sum_{z \in S_k} P^{[m]}(x, z) \leq 1 + \sum_{k=0}^{l} \sum_{z \in S_k} P^{[m]}(x, z) G(z).$$

Equivalently

$$G(y) \leq \frac{1 + \sum_{k=0}^{l} \sum_{z \in S_k} P^{[m]}(x, z) G(z)}{\sum_{k=0}^{l} \sum_{z \in S_k} P^{[m]}(x, z)}.$$

Since $x \in S_{l+1}$, it follows from (16)

$$G(x) = \frac{1 + \sum_{k=0}^{l} \sum_{z \in S_k} P^{[m]}(x, z) G(z)}{\sum_{k=0}^{l} \sum_{z \in S_k} P^{[m]}(x, z)}.$$

So we get $G(y) \leq G(x)$. The inequality is strict since x and y are in different subsets. Thus, we prove (17).

Secondly using (17), we can infer the monotonically decreasing condition easily. From (17), we draw that

$$G(x) > G(y), \quad \text{if } x \in S_l, y \in S_k \text{ with } l > k. \quad (18)$$

For any two points x and y such that $G(x) > G(y)$, let $x \in S_k$, $y \in S_l$. Then k and l must satisfy $k < l$. Then we have $f(x) = f_k < f(y) = f_l$. This proves the monotonically decreasing condition.

The conclusion is drawn from Theorem 1. ∎

The above theorem provides an approach to designing the easiest fitness functions in the function class. The idea behind the construction procedure is simple: we construct a function which is unimodal in the time-based fitness landscape and then it is the easiest. Notice that the number of the easiest

functions is infinite since the potential values of each f_l are infinite.

B. Construction of Hardest Fitness Functions

We consider an elitist $(1 + 1)$ EA and a class of fitness functions with the same optima. The hardest fitness function $f(x)$ in this class is constructed as follows.

1) Let $S_0 = S_{\text{opt}}$. For any $x \in S_0$, let $G'(x) = 0$.
2) Suppose that the subsets $S_0, \ldots, S_{l-1}$ have been produced and $G(x)$ have been defined on these subsets. Then define S_l to be the set of all points such that

$$\arg \max_{x \in S \backslash \cup_{k=0}^{l-1} S_k} \frac{1 + \sum_{k=0}^{l-1} \sum_{y \in S_k} P^{[m]}(x, y) G'(y)}{\sum_{k=0}^{l-1} \sum_{y \in S_k} P^{[m]}(x, y)}.$$

For any $x \in S_l$, set

$$G'(x) = \frac{1 + \sum_{k=0}^{l-1} \sum_{y \in S_k} P^{[m]}(x, y) G'(y)}{\sum_{k=0}^{l-1} \sum_{y \in S_k} P^{[m]}(x, y)}. \quad (19)$$

3) Repeat the above step until any point is covered by a subset. Then there exists an integer $L > 0$ such that $S = \cup_{k=0}^{L} S_k$.
4) Choose $L + 1$ number $f_0, \ldots, f_L$ such that $f_0 > \cdots > f_L > 0$. Set the fitness function to be $f(x) = f_k, x \in S_k$.

Now we prove that $f(x)$ is the hardest fitness function in the class using the monotonically increasing condition.

Theorem 4: $f(x)$ is the hardest function in the function class to the EA.

Proof: We prove that the mean runtime $G(x) = G'(x)$. The proof is similar to the first step in the proof of Theorem 3.

We prove the monotonically increasing condition. The proof is similar to the second step in the proof of Theorem 3.

The conclusion is drawn from Theorem 2. ∎

The above theorem provides an approach to designing the hardest fitness functions in the class. We construct a function which is deceptive in the time-based fitness landscape and then it is the hardest.

In the construction of the easiest and hardest functions, we do not restrict the representation of fitness functions. However, the current approach is not suitable for the fitness function class with a specific requirement, for example, all fitness functions in the class must be linear or quadratic. This research issue is left for future studies.

C. Case Study: Benchmarks in Pseudo-Boolean Optimization

So far we have introduced a general approach to constructing the easiest and hardest fitness functions. Now, we illustrate an application in pseudo-Boolean optimization: to design benchmarks within a fitness function class. According to No Free Lunch theorems, the performance of two EAs are equivalent if averaged over all possible Boolean-valued fitness functions. Therefore, we only consider a fitness class.

Example 3: Consider the class of all pseudo-Boolean functions with the same optima at $\vec{0} := (0 \cdots 0)$ and $\vec{1} := (1 \cdots 1)$

$$\max\{f(x); x \in \{0, 1\}^n\}. \quad (20)$$

We compare the performance of two $(1 + 1)$ elitist EAs on this problem using different mutation rates.

1) EA($1/n$). Flip each bit independently with flipping probability $1/n$. The mutation probability from x to y is

$$P^{[m]}(x, y) = \left(1 - \frac{1}{n}\right)^{n-h(x,y)} \left(\frac{1}{n}\right)^{h(x,y)} \quad (21)$$

where $h(x, y)$ denote the Hamming distance between x and y.

2) EA($1/2$). Flip each bit independently with flipping probability $1/2$. The mutation probability from x to y is

$$P^{[m]}(x, y) = \left(\frac{1}{2}\right)^{n}. \quad (22)$$

As to benchmark functions, their optima must be known in advance and the number of benchmarks is often between 5 to 30. Since a function class normally includes a large amount of functions, a question is which functions should be chosen as benchmarks? Naturally, we prefer typical functions in the class: easy, hard, and "averagely hard" functions. Here, we only consider how to design the easiest and hardest fitness functions.

The easiest fitness function to EA($1/n$) is constructed as follows.

1) Let $S_0 = \{\vec{0}, \vec{1}\}$. For any $x \in S_0$, define $G(x) = 0$.
2) Suppose that the subsets $S_0, \ldots, S_{l-1}$ are given and $G(x)$ has been defined on these subsets. Let S_l be the set consisting of all points such that

$$\arg \min_{x \in S \setminus \cup_{k=0}^{l-1} S_k} \frac{1 + \sum_{k=0}^{l-1} \sum_{y \in S_k} P^{[m]}(x, y)G(y)}{\sum_{k=0}^{l-1} \sum_{y \in S_k} P^{[m]}(x, y)}. \quad (23)$$

Using the mutation probability

$$P^{[m]}(x, y) = \left(1 - \frac{1}{n}\right)^{n-h(x,y)} \left(\frac{1}{n}\right)^{h(x,y)}$$

we get

$$S_l = \{x \mid \min\{h(x, \vec{0}), h(x, \vec{1})\} = l\}.$$

For any $x \in S_l$, define

$$G(x) = \frac{1 + \sum_{k=0}^{l-1} \sum_{y \in S_k} P^{[m]}(x, y)G(y)}{\sum_{k=0}^{l-1} \sum_{y \in S_k} P^{[m]}(x, y)}. \quad (24)$$

3) Repeat the above step until any point is covered by a subset. The last subset is S_L where $L := n/2$. Without loss of generality, assume n is even.
4) Choose $L+1$ numbers $f_0, \cdots, f_L$ such that $f_0 > \cdots > f_L$. Set the fitness function $f(x) = f_l$, for $x \in S_l$. Then $f(x)$ is the easiest function in the function class.

An example of the easiest function to EA($1/n$) is the Two Max function, given by

$$f(x) = n - \min\{h(x, \vec{0}), h(x, \vec{1})\}. \quad (25)$$

The runtime is calculated as follows. Let $x \in S_l$, without loss of generality, suppose it has l 0-valued bits and $n - l$ 1-valued bits (with $l \leq n - l$). The event of going from the fitness level

f_l to a higher fitness level will happen if one of 0-valued bits is flipped and other bits are kept unchanged. The probability of this event is at least

$$\binom{l}{1} \frac{1}{n} \left(1 - \frac{1}{n}\right)^{n-1} \geq \frac{l}{ne}$$

where e is Euler's constant. Thus, the runtime of going from the fitness level f_l to a higher fitness level is no more than en/l. Since the number of fitness levels is L, therefore the total runtime to reach the global optima is at most

$$\sum_{l=1}^{L} \frac{en}{l} = O(n \ln n).$$

There are infinite easiest fitness functions, including linear, quadratic, and other nonlinear functions, for example

$$f(x) = \left(n - \min\{h(x, \vec{0}), h(x, \vec{1})\}\right)^k, \quad k = 1, 2, \cdots \quad (26)$$

The runtime of the EA on all easiest fitness functions is the same no matter whether they are linear or not.

It is worth noting that the Two Max function is unimodal in the time-based fitness landscape. But using the Hamming distance, the function is two-modal due to two optima at $\vec{0}$ and $\vec{1}$.

The hardest fitness function to EA($1/n$) is constructed as follows.

1) Let $S_0 = \{\vec{0}, \vec{1}\}$. For any $x \in S_0$, let $G(x) = 0$.
2) Suppose that the subsets $S_0, \ldots, S_{l-1}$ have been produced and $G(x)$ have been defined on these subsets. Then define S_l to be the set of all points such that

$$\arg \max_{x \in S \setminus \cup_{k=0}^{l-1} S_k} \frac{1 + \sum_{k=0}^{l-1} \sum_{y \in S_k} P^{[m]}(x, y)G(y)}{\sum_{k=0}^{l-1} \sum_{y \in S_k} P^{[m]}(x, y)}.$$

Using the mutation probability

$$P^{[m]}(x, y) = \left(1 - \frac{1}{n}\right)^{n-h(x,y)} \left(\frac{1}{n}\right)^{h(x,y)}$$

we get (let $L := n/2$ and assume $n/2$ is an integer)

$$S_l = \{x \mid \min\{h(x, \vec{0}), h(x, \vec{1})\} = L - l + 1\}.$$

For any $x \in S_l$, set

$$G(x) = \frac{1 + \sum_{k=0}^{l-1} \sum_{y \in S_k} P^{[m]}(x, y)G(y)}{\sum_{k=0}^{l-1} \sum_{y \in S_k} P^{[m]}(x, y)}. \quad (27)$$

3) Repeat the above step until any point is covered by a subset. The last subset is S_L.
4) Choose $L+1$ numbers $f_0, \cdots, f_L$ such that $f_0 > \cdots > f_L$. Set the fitness function $f(x) = f_l$, for $x \in S_l$.

An example of the hardest function to EA($1/n$) is a Fully Deceptive function

$$f(x) = \begin{cases} n + 1, & \text{if } x = \vec{0}, \vec{1}; \\ \min\{h(x, \vec{0}), h(x, \vec{1})\}, & \text{otherwise.} \end{cases} \quad (28)$$

Consider a point $x \in S_1$ where x consists of exact $n/2$ zero-valued bits and x is the farthest from $\vec{0}$ and $\vec{1}$. Now we calculate the runtime $G(x)$. Since the Hamming distance between x and

TABLE I
THREE BENCHMARKS AND RUNTIME COMPARISON OF TWO EAS

name	function	time-based fitness landscape	EA($\frac{1}{n}$)	EA($\frac{1}{2}$)
Two Max	$f(x) = n - \min\{h(x, \vec{0}), h(x, \vec{1})\}$	unimodal	$O(n \ln n)$	$\Theta(2^n)$
Fully Deceptive	$f(x) = \begin{cases} n + 1, & \text{if } x = \vec{0}, \vec{1}; \\ \min\{h(x, \vec{0}), h(x, \vec{1})\}, & \text{otherwise.} \end{cases}$	deceptive	$\Theta(n^{n/2})$	$\Theta(2^n)$
Two Needles in Haystack	$f(x) = \begin{cases} 1, & \text{if } x = \vec{0}, \vec{1}; \\ 0, & \text{otherwise.} \end{cases}$	isolation	$n^{\Theta(n)} \times \Theta(1)$	$\Theta(2^n)$

the optima $\vec{0}$ and $\vec{1}$ is $n/2$, so the transition probability of going from x to the two optima is between

$$\left(1 - \frac{1}{n}\right)^{n/2} \left(\frac{1}{n}\right)^{n/2} \text{ and } 2\left(1 - \frac{1}{n}\right)^{n/2} \left(\frac{1}{n}\right)^{n/2}$$

and the runtime is $\Theta(n^{n/2})$.

There are infinite hardest fitness functions, for example, for $k = 1, 2, \cdots$

$$f(x) = \begin{cases} (n+1)^k, & \text{if } x = \vec{0}, \vec{1}; \\ \left(\min\{h(x, \vec{0}), h(x, \vec{1})\}\right)^k, & \text{otherwise.} \end{cases} \quad (29)$$

We can construct the easiest and hardest fitness functions to EA(1/2) in the same way. The easiest fitness function to EA(1/2) is constructed as follows.

1) Let S_0 be the set of optima $\vec{0}$ and $\vec{1}$.
2) Let S_1 be the set consisting of all points such that

$$\arg \min_{x \in S \setminus S_0} \frac{1}{\sum_{y \in S_0} P^{[m]}(x, y)}.$$

Using the mutation probability

$$P^{[m]}(x, y) = 2^{-n}$$

we get $S_1 = \{x \mid x \neq \vec{0}, \vec{1}\}$.
3) Choose 2 numbers f_0, f_1 such that $f_0 > f_1$. Set the fitness function $f(x) = f_l$, for $x \in S_l$. Then $f(x)$ is the easiest function in the function class.

An example of the easiest function to EA(1/2) is the Two Needles in the Haystack function

$$f(x) = \begin{cases} 1, \text{ if } x = \vec{0}, \vec{1}; \\ 0, \text{ otherwise.} \end{cases} \quad (30)$$

We calculate the runtime $G(x)$ for $x \in S_1$ as follows. The transition probability of going from x to the two optima is between $(1/2)^n$ and $2 \times (1/2)^n$. Then the runtime is $\Theta(2^n)$.

The hardest fitness function to EA(1/2) is constructed as follows.

1) Let $S_0 = \{\vec{0}, \vec{1}\}$.
2) Let S_1 be the set of all points such that

$$\arg \max_{x \in S \setminus S_0} \frac{1}{\sum_{y \in S_0} P^{[m]}(x, y)}.$$

Using the mutation probability

$$P^{[m]}(x, y) = 2^{-n}$$

we get $S_1 = \{x \mid x \neq \vec{0}, \vec{1}\}$.
3) Choose two numbers f_0, f_1 such that $f_0 > f_1$. Set the fitness function $f(x) = f_l$, for $x \in S_l$. Then the above function $f(x)$ is the hardest to EA(1/2).

An example of the hardest function to EA(1/2) is the Two Needles in the Haystack function, the same as the easiest function. The runtime is $\Theta(2^n)$. Since the runtime of EA(1/2) on both the easiest and hardest functions is $\Theta(2^n)$. Then we know for any function in the class, its runtime is $\Theta(2^n)$.

We have constructed three benchmark functions: Two Max, Fully Deceptive, and Two Needles in the Haystack. They are described in Table I. The three functions represent three typical fitness landscapes: unimodal, deceptive, and isolation. Using the benchmarks, we can make a fair comparison of the performance of EA(1/n) and EA(1/2). Table I lists the results.

The runtime of EA(1/n) on the Two Needles in the Haystack function is calculated as follows. Suppose the initial point x consists of $\Theta(n)$ 0-valued bits and $\Theta(n)$ 1-valued bits, then the event of going from x to the optima happens when either all 0-valued bits are flipped and other bits unchanged; or all 1-valued bits are flipped and other bits unchanged. The probability of the event is $(1/n)^{\Theta(n)} \times \Theta(1)$. Thus, the runtime is $n^{\Theta(n)} \times \Theta(1)$.

From the table, we see that EA(1/n) is better than EA(1/2) on the Two Max function, but worse on the Fully Deceptive Points and Two Needles in the Haystack functions. The comparison gives an understanding of the two EAs' ability in different fitness landscapes: unimodal, deceptive, and isolation. Each EA has its own advantage. EA(1/n) is more suitable for unimodal functions, but EA(1/2) performs better on deceptive or isolation functions.

The runtime of EA(1/n) and EA(1/2) increases exponentially fast on the Fully Deceptive and Two Needles in the Haystack functions. Thus, it will be difficult to compare the runtime of the EAs via computer experiments unless n is small.

V. MUTUAL TRANSFORMATION BETWEEN THE EASIEST AND HARDEST FITNESS FUNCTIONS

In the case study of the previous section, we observe that the easiest and hardest fitness functions vary as EAs change. In this section, we prove an interesting result: a fitness function that is the easiest to one elitist (1 + 1) EA could becomes hardest to another elitist (1 + 1) EA and vice versa.

A. Easiest May Become Hardest

Consider a class consisting of all functions with the same optima. Let $f(x)$ be the easiest to an elitist (1 + 1) EA (called the original EA). Denote its fitness levels by $f_0 > \cdots > f_L$ and define the set $S_l = \{x; f(x) = f_l\}$. We construct another elitist (1 + 1) EA (called the bad EA) and show $f(x)$ is the hardest to the bad EA.

The mutation operator in the bad EA is constructed as follows.

1) Choose $L+1$ nonnegative numbers $m_0, m_1, \cdots, m_L$ such that $m_0 = 0, m_1 > m_2 > \cdots > m_L > 0$.
2) For any $x \in S_0$ and $y \in S$, let the mutation transition probability $P^{[m]}(x, y)$ be any probability.
3) For any $x \in S_l$ (where $l = 1, \ldots, L$) and $y \in S$, set the mutation transition probability $P^{[m]}(x, y)$ such that

$$\frac{1 + \sum_{j=0}^{k-1} \sum_{y \in S_j} P^{[m]}(x, y) m_j}{\sum_{j=0}^{k-1} \sum_{y \in S_j} P^{[m]}(x, y)} < m_k, \text{ for } k < l$$

and

$$\frac{1 + \sum_{j=0}^{l-1} \sum_{y \in S_j} P^{[m]}(x, y) m_j}{\sum_{j=0}^{l-1} \sum_{y \in S_j} P^{[m]}(x, y)} = m_l. \tag{31}$$

The above mutation operator is determined by the subsets $S_1, \ldots, S_l$ rather than fitness levels.

The following theorem shows the function $f(x)$ satisfies the monotonically increasing condition and then it is the hardest to the bad EA.

Theorem 5: $f(x)$ is the hardest function to the bad EA.

Proof: We prove that the expected runtime of the bad EA $G(x) = m_l$, for $x \in S_l, l = 0, \cdots, L$.

According to Lemma 4, the expected runtime

$$G(x) = \frac{1 + \sum_{j=0}^{l-1} \sum_{y \in S_j} P(x, y) G(y)}{\sum_{j=0}^{l-1} \sum_{y \in S_j} P(x, y)}.$$

For any $x \in S_l$ and $y \in S_k$ where $l > k$. Since $f(x) = f_l < f(y) = f_k$ and the bad EA adopts elitist selection, y is always accepted if it has been generated via mutation. Thus, the transition probability $P(x, y)$ equals to $P^{[m]}(x, y)$.

The expected runtime becomes

$$G(x) = \frac{1 + \sum_{j=0}^{l-1} \sum_{y \in S_j} P^{[m]}(x, y) G(y)}{\sum_{j=0}^{l-1} \sum_{y \in S_j} P^{[m]}(x, y)}.$$

Comparing it with (31), we obtain $G(x)$ and m_l are identical. We prove the monotonically increasing condition.

Assume that $x \in S_l, y \in S_k$ for some l and k. If $G(x) < G(y)$, then it is equivalent to $m_l < m_k$. Thus, we have $k < l$ and

$$f(x) = f_l < f(x) = f_k.$$

which gives the monotonically increasing condition.

The conclusion is drawn from Theorem 2. ∎

In the construction of the mutation operator and the proof of the above theorem, we do not utilize the assumption of $f(x)$ being the easiest to the original EA. Thus, the theorem can be understood more generally: for any fitness function $f(x)$, we can construct an elitist (1 + 1) EA to which $f(x)$ is the hardest. From the theoretical viewpoint, the theorem shows the existence of a bad EA to the easiest fitness function.

B. Hardest May Become Easiest

Let $f(x)$ be the hardest fitness function to the original elitist (1 + 1) EA. We construct another elitist (1 + 1) EA (called the good EA), and show $f(x)$ becomes the easiest to the good EA.

The mutation operator in the good EA is constructed as follows.

1) Choose $L + 1$ nonnegative numbers $m_0, \cdots, m_L$ such that $m_0 = 0 < \cdots < m_L$.
2) For any $x \in S_0$ and $y \in S$, let the mutation transition probability $P^{[m]}(x, y)$ be any probability.
3) For any $x \in S_l$ (where $l = 1, \ldots, L$) and $y \in S$, set the mutation transition probability $P^{[m]}(x, y)$ such that

$$\frac{1 + \sum_{j=0}^{k-1} \sum_{y \in S_j} P^{[m]}(x, y) m_j}{\sum_{j=0}^{k-1} \sum_{y \in S_j} P^{[m]}(x, y)} > m_k, \quad \text{for } k < l$$

and

$$\frac{1 + \sum_{j=0}^{l-1} \sum_{y \in S_j} P^{[m]}(x, y) m_j}{\sum_{j=0}^{l-1} \sum_{y \in S_j} P^{[m]}(x, y)} = m_l. \tag{32}$$

The following theorem shows $f(x)$ satisfies monotonically decreasing condition and then it is the easiest to the good EA.

Theorem 6: $f(x)$ is the easiest function to the good EA.

Proof: We prove that the expected runtime of the good EA $G(x) = m_k$, for $x \in S_k, k = 0, 1, \ldots, L$. The proof is similar to the first step in the proof of Theorem 5.

We prove the monotonically decreasing condition. The proof is similar to the second step in the proof of Theorem 5.

The conclusion is drawn from Theorem 1. ∎

In the construction of the mutation operator and the proof of the above theorem, we also do not utilize the assumption of $f(x)$ being the hardest to the original EA. The theorem implies that for any fitness function $f(x)$, we can construct a good (1 + 1) EA to which $f(x)$ is the easiest.

The above theorem reveals if a fitness function is the hardest to one EA, then it is possible to design another good EA to which the function is the easiest. However, the above construction method is intractable in practice since the complexity of construction is exponential. How to design such a good EA is an ultimate goal in the study of EAs but beyond the scope of the current paper.

Theorems 5 and 6 can be viewed as a complement to No Free Lunch theorems. No Free Lunch theorems concern all potential fitness functions. The theorems claim the performance of any two EAs are equivalent if averaged over all possible functions. Theorems 5 and 6 concern the hardness of a single fitness function. The two theorems assert that a fitness function could be the easiest to one elitist (1 + 1) EA but the hardest to another EA. This implies for a single fitness function, a good EA (but also a bad EA) always exists.

VI. CONCLUSION AND FUTURE WORK

This paper presents a rigorous analysis devoted to the easiest and hardest fitness functions with respect to any given elitist (1 + 1) EA for maximizing a class of fitness functions with the same optima. Such fitness functions have been constructed step-by-step. It is demonstrated that the unimodal functions are the easiest and deceptive functions are the hardest in terms of the time-based fitness landscape. Furthermore, it reveals that the hardest (and easiest) functions may become the easiest (and hardest) with respect to another elitist (1 + 1) EA. From the theoretical viewpoint, a good EA (but also a bad EA) always exists for a single fitness function.

A potential application of the theoretical work is the design of benchmarks. Benchmarks play an essential role in the empirical comparison of EAs. In order to make a fair comparison, a good practice is to choose typical fitness functions in benchmarks, for example, several easy, hard and averagely hard fitness functions. Our work provides a theoretical guideline to the design of easy and hard functions: to choose unimodal (the easiest) and deceptive (the hardest) fitness functions with respect to EAs under comparison.

Another application is to understand the ability of EAs on a class of fitness functions with the same optima. Through the comparison of EAs on the easiest and hardest fitness functions, our work helps understand the ability of EAs in unimodal and deceptive time-based fitness landscapes. This has been shown in the second case study.

Nonelitist EAs, population-based EAs and dynamical EAs are not investigated in this paper. The extension of our work to such EAs will be the focus of future research. Another work in the future is to study how to construct the easiest and hardest fitness functions such that a special requirement, for example, all fitness functions must be linear or quadratic.

REFERENCES

[1] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 67–82, Apr. 1997.

[2] D. H. Wolpert and W. G. Macready, "Coevolutionary free lunches," *IEEE Trans. Evol. Comput.*, vol. 9, no. 6, pp. 721–735, Dec. 2005.

[3] J. Horn and D. E. Goldberg, "Genetic algorithms difficulty and the modality of fitness landscapes," in *Proc. 3rd Workshop Foundations Genetic Algorithms*, San Francisco, CA, USA, 1995, pp. 243–269.

[4] G. Rudolph, "How mutation and selection solve long path problems in polynomial expected time," *Evol. Comput.*, vol. 4, no. 2, pp. 207–211, 1996.

[5] M. D. Vose and A. H. Wright, "Stability of vertex fixed points and applications," in *Proc. 3rd Workshop Foundations Genetic Algorithms*, 1995, pp. 103–114.

[6] S. W. Wilson, "GA-easy does not imply steepest-ascent optimizable," in *Proc. 4th Int. Conf. Genetic Algorithms*, 1991, pp. 85–89.

[7] T. Jones and S. Forrest, "Fitness distance correlation as a measure of problem difficulty for genetic algorithms," in *Proc. 6th Int. Conf. Genetic Algorithms*, San Francisco, CA, USA, 1995, pp. 184–192.

[8] N. J. Radcliffe and P. D. Surry, "Fitness variance of formulae and performance prediction," in *Proc. 3rd Workshop Foundations Genetic Algorithms*, 1995, pp. 51–72.

[9] Y. Davidor, "Epistasis variance: A viewpoint on GA-hardness," in *Proc. 1st Workshop Foundations Genetic Algorithms*, 1991, pp. 23–35.

[10] B. Naudts and L. Kallel, "A comparison of predictive measure of problem difficulty in evolutionary algorithms," *IEEE Trans. Evolut. Comput.*, vol. 4, no. 1, pp. 1–15, Apr. 2000.

[11] J. He, C. Reeves, C. Witt, and X. Yao, "A note on problem difficulty measures in black-box optimization: Classification, existence and predictability," *Evol. Comput.*, vol. 15, no. 4, pp. 435–443, 2007.

[12] B. Manderick, M. K. de Weger, and P. Spiessens, "The genetic algorithms and the structure of the fitness landscape," in *Proc. 4th Int. Conf. Genetic Algorithms*, 1991, pp. 143–150.

[13] T. Jansen, "On classifications of fitness functions," in *Theoretical Aspects of Evolutionary Computing*, L. Kallel, B. Naudts, and A. Rogers, Eds. Berlin, Germany: Springer, 2001, pp. 371–386.

[14] C. Reeves, "Predictive measures for problem difficulty," in *Proc. 1999 Congr. Evol. Comput.*, vol. 1. Washington, DC, USA, pp. 736–743.

[15] J. He and X. Yao, "An analysis of evolutionary algorithms for finding approximation solutions to hard optimisation problems," in *Proc. IEEE 2003 Congr. Evol. Comput.*, pp. 2004–2010.

[16] J. He and X. Yao, "Towards an analytic framework for analysing the computation time of evolutionary algorithms," *Artif. Intell.*, vol. 145, nos. 1–2, pp. 59–97, 2003.

[17] T. Chen, J. He, G. Sun, G. Chen, and X. Yao, "A new approach for analyzing average time complexity of population-based evolutionary algorithms on unimodal problems," *IEEE Trans. Syst., Man, Cybern B., Cybern,* vol. 39, no. 5, pp. 1092–1106, Oct. 2009.

[18] T. Chen, J. He, G. Chen, and X. Yao, "Choosing selection pressure for wide-gap problems," *Theor. Comput. Sci.*, vol. 411, no. 6, pp. 926–934, 2010.

[19] J. He and X. Yao, "A study of drift analysis for estimating computation time of evolutionary algorithms," *Nat. Comput.*, vol. 3, no. 1, pp. 21–35, 2004.

[20] J. He, X. Yao, and Q. Zhang, "To understand one-dimensional continuous fitness landscapes by drift analysis," in *Proc. 2004 Congr. Evolut. Comput.*, pp. 1248–1253.

Jun He (M'06) received the B.S. and M.Sc. degrees in mathematics and the Ph.D. degree in computer science from Wuhan University, Wuhan, China, in 1989, 1992, and 1995, respectively.

He is a Senior Lecturer with Aberystwyth University, Aberystwyth, U.K. His research interests include evolutionary computation, global optimization, and network security. He has published over 80 papers in these areas.
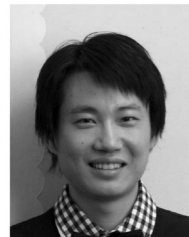
Tianshi Chen received the B.S. degree in mathematics from Special Class for the Gifted Young, University of Science and Technology of China (USTC), Hefei, China, and the Ph.D. degree in computer science from the Department of Computer Science and Technology, USTC, in 2005 and 2010, respectively.

He is an Associate Professor with State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences, Beijing. His research interests include evolutionary computation, computer architecture, and parallel computing.

Dr. Chen received the 2011 China Computer Federation Distinguished Doctoral Dissertation Award and the 2011 Chinese Academy of Sciences Distinguished Doctoral Dissertation Award for the Ph.D. work on computational complexity analysis of evolutionary algorithms. He won the ASPLOS'14 Best Paper Award for work on a hardware accelerator of artificial neural networks.

Xin Yao (M'91-SM'96-F'03) received the B.S. degree from the University of Science and Technology of China (USTC), Hefei, China, in 1982, the M.S. degree from the North China Institute of Computing Technology, Beijing, China, in 1985, and the Ph.D. degree from USTC in 1990.

He is a Chair Professor of Computer Science and the Director of the Centre of Excellence for Research in Computational Intelligence and Applications, University of Birmingham, Birmingham, U.K. He is a Distinguished Lecturer of IEEE Computational Intelligence Society (CIS). His research interests include evolutionary computation and ensemble learning. He has published over 400 refereed publications in international journals and conferences.

Prof. Yao received the 2001 IEEE Donald G. Fink Prize Paper Award, 2010 IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION Outstanding Paper Award, 2010 BT Gordon Radley Award for Best Author of Innovation (Finalist), 2011 IEEE TRANSACTIONS ON NEURAL NETWORKS Outstanding Paper Award, and many other Best Paper awards. He received the prestigious Royal Society Wolfson Research Merit Award in 2012 and the IEEE CIS Evolutionary Computation Pioneer Award in 2013. From 2003 to 2008, he was an Editor-in-Chief of IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION.