ABSTRACT
Computing unique input-output sequences (UIOs) from finite state machines (FSMs) is important for conformance testing in software engineering, where evolutionary algorithms (EAs) have been found helpful. Previously, by using a fitness function called W-fitness, (1+1)-EA was theoretically shown to be superior to random search on some FSM instances. Motivated by the observation that many plateau exits in the fitness landscape of the W-fitness function, in this paper, we propose a new fitness function called C-fitness which is able to override the plateaus through exploiting collisions among the states of FSMs. We theoretically analyze the running time of (1+1)-EA on two problem classes. Our results show that the performance of (1+1)-EA using C-fitness is generally better and never worse than that using W-fitness in our studied cases, implying the importance of exploiting problem structures.

Categories and Subject Descriptors
E.2 [Theory of Computation]: Analysis of Algorithms and Problem Complexity

General Terms
Theory

Keywords
Evolutionary algorithms, fitness function, running time, software engineering

1. INTRODUCTION
Conformance testing, which verifies whether the software system meets the specified requirements, is an important task in software engineering. Essentially, the state verification of conformance testing is to compute unique input-output sequences (UIOs) from finite state machines (FSMs); this problem is NP-hard [5] and thus tackled often by heuristic approaches. Computing UIO sequences can be reformulated as a search problem, and therefore evolutionary algorithms (EAs) have been applied [3, 1].

Definition 1 (FSM [5]) An FSM $M$ is a quintuple $M = (I, O, S, \delta, \lambda)$, where $I$ and $O$ are the sets of input and output symbols, respectively, $S$ is the state set, $\delta : S \times I \rightarrow S$ is the state transition function and $\lambda : S \times I \rightarrow O$ is the output function.

Definition 2 (UIOs [5]) For an FSM $M = (I, O, S, \delta, \lambda)$, an UIO sequence of state $s$ is an input sequence $x$ such that $\lambda(s, x) \neq \lambda(s', x)$, for any $s' \in S - \{s\}$.

When applying EAs to compute UIOs, a solution is usually represented as a sequence of length-$n$ input symbols, since all the considered FSM instance classes have the UIO sequence of length $n$. Among the several types of fitness functions which have been used to measure the goodness of the solutions [1, 3, 6], Lehre and Yao [6] used the following one; we call it W-fitness.

Definition 3 (W-fitness [6]) For computing UIO sequence of state $s$ on an FSM $M = (I, O, S, \delta, \lambda)$, the fitness of an input sequence $x$ is $f^w(x) = |\{s' \mid \lambda(s', x) \neq \lambda(s, x), s' \in S\}|$.

Empirically, EAs can outperform random search on complex and large FSMs [1, 3]. There are also theoretical studies disclosing the behaviors of EAs on computing UIOs. For example, Lehre and Yao analyzed the influence of the EA configurations on the running time for computing UIOs [7], and disclosed that (1+1)-EA is superior to random search in computing UIOs [6].

2. MAIN RESULTS
First, we propose the C-fitness function by exploiting collisions among the states for computing UIOs.

Definition 4 (Collision) Given an FSM $M = (I, O, S, \delta, \lambda)$, an input sequence $x$ has a collision between states $s$ and $s'$ if $\lambda(s, x) = \lambda(s', x) \land \delta(s, x) = \delta(s', x)$. For any $s$ and $x$, let $\hat{x}$ be the prefix of $x$ with length $i$. The collision time $\phi(s, x) = \max\{i | 0 \leq i \leq |x|, \hat{x}, \hat{x} \text{ has no collision between } s \text{ and other states}\}$.

Definition 5 (C-fitness) For computing UIO sequence of state $s$ on an FSM $M = (I, O, S, \delta, \lambda)$, the fitness of an input sequence $x$ is $f^c(x) = f^w(x) + \phi(s, x)$. In contrast to the W-fitness function, the C-fitness function incorporates the collision time, and therefore contains more collision information. The UIO sequences of state $s$ have the maximal C-fitness value $(n - 1 + |x|)$.

Next, we define two FSM instance classes and compare the running time of (1+1)-EA using C-fitness and W-fitness, respectively. Denote $LO(*)$ for the length of the longest-common-prefix and $H(*)$ for the Hamming distance between the sequence $*$ and the UIO sequence. Let $n$ be the length of the input sequence, $m$ be the cardinality of the input symbol set $I$.

Definition 6 (With-Collision FSM Instance Class) Assuming only one UIO sequence exists for a state, FSM in this class satisfies that, for some state $s$, $\phi(s, x) = LO(x)$.

That is, the input sequence $x$ has a collision as soon as its symbol is different from the corresponding symbol of the UIO sequence.
Theorem 1 In the with-collision FSM instance class, if \( W\)-fitness assigns the same value to all solutions except the UIO sequence, the expected running time of (1+1)-EA on \( C\)-fitness is \( \Theta(mn^2) \) while that on \( W\)-fitness is \( \Theta(n^2) \).

Theorem 2 In the with-collision FSM instance class, if it holds for \( W\)-fitness that \( f^w(x) > f^w(x') \) if \( LO(x) > LO(x') \), and \( f^w(x) = f^w(x') \) if \( LO(x) = LO(x') \), then the expected running time of (1+1)-EA on \( C\)-fitness and \( W\)-fitness are both \( \Theta(n^2) \).

Theorem 3 In the with-collision FSM instance class, if it holds for \( C\)-fitness that \( f^c(x) > f^c(x') \) if \( H(x) > H(x') \) or \( H(x) = H(x') \land LO(x) > LO(x') \), then the expected running time of (1+1)-EA on \( C\)-fitness and \( W\)-fitness are both \( \Theta((m-1)n^2) \).

The above theorems are proved by considering the structural similarity between the fitness (\( W\)-fitness or \( C\)-fitness) functions and the well-studied Trap, LeadingOnes and Needle functions. The analysis of (1+1)-EA on these three functions by [2] is used here to analyze (1+1)-EA on the above problems. We give three example FSM instances which satisfy Theorems 1 to 3, in Figures 1 to 3, respectively.

We conjecture that, in the with-collision FSM instance class, if it holds for \( W\)-fitness that \( f^w(x) < f^w(x') \) if \( H(x) < H(x') \), \( C\)-fitness can decrease the difficulty of \( W\)-fitness for (1+1)-EA, or they are with the same hardness. Figure 4 presents an example FSM instance. It can be proved that \( W\)-fitness on the SPC FSM instance leads to \( \Theta(n^3) \) expected running time, as \( W\)-fitness resembles the Trap function [2] here; while \( C\)-fitness leads to \( O(n^2) \) expected running time, as \( C\)-fitness resembles the SPC function [4] here.

Proposition 1 The expected running time of (1+1)-EA on the \( C\)-fitness function for SPC FSM instance is upper-bounded by \( O(n^2) \).

Proof. The \( C\)-fitness function for SPC FSM instance is

\[
f^c(x) = \begin{cases} 
3n - 1, & \text{if } x = 1^{n-1}0 \\
n, & \text{if } x = 1^n \\
n + 1, & \text{if } x = 1^i0^{n-i}, 0 \leq i < n - 1 \\
1 + \sum_{i=1}^{n-1} (1 - x_i) + \sum_{j=1}^{n-1} x_j, & \text{otherwise.}
\end{cases}
\]

(1+1)-EA reaches the path \( 0^n, 10^{n-1}, \ldots, 1^{n-1}0 \) within \( O(n \log n) \) steps. After that, it needs \( O(n^2) \) expected running time to reach the optimal solution \( 1^{n-1}0 \) [4].

Definition 7 (The Without-Collision FSM Instance Class) FSM in this class satisfies that \( \forall x, o(s, x) = n \) when computing the UIO sequence for some state \( s \).

That is, no collision occurs between state \( s \) and other states for any input sequence. Figures 5 and 6 give two example FSM instances of this class.

Theorem 4 In the without-collision FSM instance class, (1+1)-EA on the \( C\)-fitness function behaves as same as \( W\)-fitness function.

3. CONCLUSION

In this paper, by exploiting collisions among states, we propose the \( C\)-fitness function which is able to override the plateaus of the \( W\)-fitness function and thus makes problems swift from hard to easy for (1+1)-EA. Our results show that \( C\)-fitness function is generally better and never worse than \( W\)-fitness function; this suggests that an adequate problem formulation is important and acceleration can be obtained by exploiting problem structures. Detailed proofs will be presented in a longer version.

4. REFERENCES


