A Parameterized Runtime Analysis of Evolutionary Algorithms for MAX-2-SAT

Andrew M. Sutton Jareth Day Frank Neumann

School of Computer Science
University of Adelaide
Adelaide, SA 5005, Australia
{andrew.sutton, jareth.day, frank.neumann}@adelaide.edu.au

ABSTRACT

We investigate the MAX-2-SAT problem and study evolutionary algorithms by parameterized runtime analysis. The parameterized runtime analysis of evolutionary algorithms has been initiated recently and reveals new insights into which type of instances of NP-hard combinatorial optimization problems are hard to solve by evolutionary computing methods. We show that a variant of the (1+1) EA is a fixed-parameter evolutionary algorithm with respect to the standard parameterization for MAX-2-SAT. Furthermore, we study how the dependencies between the variables affect problem difficulty and present fixed-parameter evolutionary algorithms for the MAX-(2,3)-SAT problem where the studied parameter is the diameter of the variable graph.

Categories and Subject Descriptors

F.2 [**Theory of Computation**]: Analysis of Algorithms and Problem Complexity

General Terms

Theory, Algorithms, Performance

Keywords

Combinatorial Optimization, MAXSAT, Theory, Runtime Analysis

1. INTRODUCTION

Bio-inspired computing methods such as evolutionary algorithms [5] and ant colony optimization [3] have been widely used for problems from combinatorial optimization. The mentioned algorithms iteratively try to improve currently best solutions during the optimization process and heavily rely on the use of randomness for creating new solutions and/or selecting solutions for the next iteration. Treating evolutionary algorithms as a special type of randomized algorithms allows them to be analyzed in a rigorous way. We contribute to this line of research which has been very successful during the last 15 years. This includes results on artificial functions that rigorously point out the working principles of bio-inspired algorithms as well as results for a wide range of classical combinatorial optimization problems. A comprehensive presentation of such results can be found in [1, 11].

Recently, evolutionary algorithms have been analyzed in the framework of fixed-parameter tractability [4]. Here an additional parameter to measure the hardness of an instance is taken into account. Initial results in the field of evolutionary algorithms have been obtained for two NP-hard combinatorial optimization problems, namely the vertex cover problem [10] and the problem of computing a spanning tree with a maximal number of leaves [9]. We want to push forward this relatively young line of research as it allows one to figure out which type of instances of a given (NP-hard) combinatorial optimization problem is provably easy to solve by evolutionary algorithms. Consequently, our runtime analysis will take into account the size of the given problem denoted by n (as usual for the analysis of algorithms) and an additional parameter k which measures the difficulty of an instance.

We study the well-known MAX-SAT problem. MAX-SAT is already NP-hard if the clauses have two variables as this corresponds to the optimization of quadratic pseudo-Boolean functions. We consider MAX-2-SAT and analyze the runtime behavior of evolutionary algorithms for this problem with respect to the standard parameterization, i.e. the value of an optimal solution. Our analysis takes into account results on the fitness landscape of such instances [13] by introducing a mutation operator that obeys Grover's wave equation [8] for MAX-2-SAT which has the interesting side-effect of an existence proof of an *elementary landscape* for the MAX-2-SAT problem. To our knowledge, this result is novel. We employ this difference equation to show that the (1+1) EA is a fixed-parameter algorithm for the standard parameterization of MAX-2-SAT. This bridges for the first time the field of runtime analysis and the study of landscapes in the field of theory of evolutionary computation.

Afterwards, we turn our attention the MAX-(2,3)-SAT problem which is an NP-hard subclass of MAX-2-SAT. We consider a parameterization which takes into account the degree of dependencies between the different variables. In the variable graph of a given MAX-(2,3)-SAT problem, the nodes are given by the Boolean variables of the problem and there is an edge between any pair of variables if they are present in the same clause. The parameter that we analyze is the diameter of the variable graph. We show that the (1+1) EA is a randomized XP-algorithm for this problem and show how a more global search behavior taking into account the variable graph can lead to fixed-parameter randomized algorithms. Furthermore, we explore different more local evolutionary algorithms working on the variable graph and show that these are fixed-parameter evolutionary algorithms for MAX-(2,3)-SAT.

The content of the paper is as follows. In Section 2, we introduce the MAX-2-SAT problem and the concept of parameterized analysis. In Section 3, we show that the classical (1+1) EA is a fixed-parameter evolutionary algorithm with respect to the standard parameterization. Section 4 analyzes the MAX-(2,3)-SAT problem parameterized by the diameter of the variable graph and shows that evolutionary algorithms are fixed-parameter algorithms for this problem. Furthermore, we present additional fixed-parameter evolutionary algorithms acting more locally for the MAX-(2,3)-SAT in Section 5. We summarize our findings with some concluding remarks.

2. PRELIMINARIES

The maximum 2-satisfiability problem (MAX-2-SAT) is a classic optimization problem in which, given a Boolean formula in conjunctive normal form where each clause is of cardinality exactly two,[1] one must find an assignment to its variables that satisfies the maximum number of clauses. Despite its apparent simplicity, the MAX-2-SAT problem belongs to the class of NP-hard problems [7].

An instance of a MAX-2-SAT problem is represented as a collection of m unique clauses

$$\mathscr{C} = \{(\ell_{1,1} \vee \ell_{1,2}), (\ell_{2,1} \vee \ell_{2,2}), \ldots, (\ell_{m,1} \vee \ell_{m,2})\}$$

where each $\ell_{i,j}$ is a Boolean variable or its negation. The set of all assignments to a set of n Boolean variables is isomorphic to $\{0,1\}^n$ by interpreting each position of the string as the state of exactly one Boolean variable v_i (i.e., a 1 corresponds to $v_i = \mathit{true}$; a 0 corresponds to $v_i = \mathit{false}$). Thus, given a MAX-2-SAT instance with n variables and m clauses, we represent candidate solutions to the instance as length-n bitstrings and define the function[2] $f : \{0,1\}^n \to \{0\} \cup [m]$ where $f(x)$ counts the elements in $\mathscr{C}$ that are satisfied under the assignment corresponding to $x \in \{0,1\}^n$. The task is then reduced to the optimization of a pseudo-Boolean function f.

In this paper we rigorously analyze the runtime character of the (1+1) EA maximizing the pseudo-Boolean function f. The (1+1) EA is defined as follows.

Algorithm 1: The (1+1) EA.

```
1 Choose $x$ uniformly at random from $\{0,1\}^n$;
2 repeat forever
3     $x' \leftarrow$ mutate($x$);
4     if $f(x') \geq f(x)$ then $x \leftarrow x'$
```

[1]Some authors refer to this as MAX-E2-SAT.

[2]For a natural number n, throughout the paper we denote the set $\{1,\ldots,n\}$ as $[n]$.

The `mutate()` procedure is a randomized operator that negates some elements of the current string to produce a new string. Since the (1+1) EA relies on random decisions, we consider it as a special case of a *randomized algorithm*. We view the (1+1) EA as an infinite stochastic process $(x^{(1)}, x^{(2)}, \ldots)$ where $x^{(t)}$ denotes the string that the (1+1) EA produces in iteration t. For a given instance of MAX-2-SAT, let $f^{\max} = \max_{x \in \{0,1\}^n} f(x)$. We denote as T the lowest value of t such that $f(x^{(t)}) = f^{\max}$. T is a discrete random variable over $\mathbb{N}$, and we can measure the expected *optimization time* of the (1+1) EA as $E(T)$, the expectation of T.

Recently, the concept of *parameterized analysis* has been introduced into the theoretical analysis of evolutionary algorithms [10, 9]. Parameterized analysis allows for a more detailed understanding of the source of runtime complexity for NP-hard combinatorial optimization problems. A parameterized analysis expresses the runtime of an algorithm not only in terms of the size of the problem, but also as a function of a parameter that somehow measures the difficulty of the problem.

A *parameterized problem* (L, κ) consists of a language L over a finite alphabet Σ and a *parameterization* $\kappa : \Sigma^* \to \mathbb{N}$. (L, κ) is *fixed-parameter tractable* if there is an algorithm that decides $x \in L$ in time bounded by $g(\kappa(x)) \cdot |x|^{O(1)}$ where $g : \mathbb{N} \to \mathbb{N}$ is an arbitrary recursive function. Such an algorithm is called an *fpt-algorithm* and the problem class FPT consists of all decision problems that can be decided by an fpt-algorithm. A problem L is in the class XP if there is an algorithm that decides $x \in L$ in time bounded by $|x|^{g(\kappa(x))}$ for some computable function g.

A *Monte Carlo fpt-algorithm* for (L, κ) is a randomized fpt-algorithm such that for all $x \in \Sigma^*$ that in time bounded by $g(\kappa(x)) \cdot |x|^{O(1)}$ will accept x with probability at least 1/2 if $x \in L$ and with probability 0 if $x \notin L$. Similarly, a *Monte Carlo* XP*-algorithm* for (L, κ) is a randomized algorithm that, for all $x \in \Sigma^*$, if $x \in L$ accepts x with probability at least 1/2 in time $|x|^{g(\kappa(x))}$, otherwise it accepts with probability 0.

An evolutionary algorithm is a *fixed-parameter evolutionary algorithm* for a parameterized problem (L, κ) if its expected optimization time $E(T)$ is bounded above by $g(\kappa(x)) \cdot |x|^{O(1)}$. Similarly, an evolutionary algorithm is an XP *evolutionary algorithm* if its runtime is bounded above by $|x|^{g(\kappa(x))}$. We will later see that when the parameterization is related to the optimal solution of a combinatorial optimization problem, then a fixed-parameter evolutionary algorithm can be transformed into a Monte Carlo fpt-algorithm.

3. MAX-2-SAT

We now introduce a simple parameterized result for the (1+1) EA that connects the field of runtime analysis to the study of fitness landscapes. We devise a particular mutation operator that allows the (1+1) EA to at least make local improvements in such a way that it is guaranteed to reach a solution of a specific fitness in expected polynomial time.

3.1 Uniform-complement mutation

The standard (uniform) mutation operator in the traditional implementation of the (1+1) EA generates an offspring of $x \in \{0,1\}^n$ by flipping each bit of x with probability $1/n$. In *uniform-complement* mutation, we proceed similarly, except we also include with constant probability the *complement* of x, that is, the string obtained by flipping all bits of x.

Function `uniform-complement-mutate(`x`)`

```
1 $y \leftarrow x$;
2 choose $r \in [0,1]$ uniformly at random;
3 if $r < 1/2$ then
4     flip each bit of $y$ with probability $1/n$;
5 else
6     flip each bit of $y$ with probability 1;
7 return $y$;
```

We point out that uniform mutation can also generate the complement of x with finite probability. The distinction is that the probability of obtaining the complement of x under uniform-complement mutation is a constant, whereas it is $O(n^{-n})$ under uniform mutation. This constant probability to generate the complement is vital to our analysis.

The uniform-complement operator induces a connectivity on $\{0,1\}^n$ which obeys a strict difference equation from the theory of landscapes usually attributed to Grover [8] on MAX-2-SAT. This property will allow us to prove that the (1+1) EA discovers solutions of a certain quality in polynomial time.

Consider a MAX-2-SAT problem with n variables and m clauses. Given an element $x \in \{0,1\}^n$, we denote $N(x)$ as the set constructed from taking the union of immediate Hamming neighbors of x and the complement of x. We have the following result

Lemma 1. *For all $x \in \{0,1\}^n$,*

$$\sum_{y \in N(x)} f(y) = 3m + (n-3)f(x).$$

Proof. Denote as $c_i : \{0,1\}^n \to \{0,1\}$ the indicator function that evaluates to 1 if and only if clause i is satisfied by the assignment corresponding to x.

If the i-th clause is not satisfied by x, then there are exactly three elements $y \in N(x)$ under which it is satisfied: the two distinct Hamming neighbors that negate each variable appearing in the clause, and the element corresponding to the complement of x, which negates both variables in the clause.

If the i-th clause is satisfied by x then at least one of its literals evaluates to true under x. In the case that exactly one of its literals evaluates to true, then there is exactly one element $y \in N(x)$ in which the clause is not satisfied (corresponding to the negation of the variable involved in the true literal). If both its literals evaluate to true under x, then there is exactly one element in $N(x)$ such that the clause is not satisfied (corresponding to the complement of x). Hence we can write c_i as

$$\sum_{y \in N(x)} c_i(y) = 3(1-c_i(x)) + (|N(x)|-1)c_i(x) = 3+(n-3)c_i(x).$$

Since $f(x) = \sum_{i=1}^m c_i(x)$, we have

$$\sum_{y \in N(x)} f(y) = \sum_{i=1}^{m} \big(3 + (n-3)c_i(x)\big) = 3m + (n-3)f(x).$$

This completes the proof. □

Lemma 1 states that Grover's equation holds for MAX-2-SAT under uniform-complement mutation. Equivalently, this means that the fitness landscape induced by the uniform-complement operator and that MAX-2-SAT fitness function corresponds to an *elementary landscape* [14]. The existence of a move operator that induces an elementary landscape for the MAX-2-SAT problem was, to our knowledge, previously unknown. One immediate result we can obtain from this condition is that solutions of certain quality can be generated in polynomial time by making local improvements. This is due to the fact that, on such a landscape, local optima (with respect to N) cannot have arbitrarily poor fitness.

Lemma 2. *The (1+1) EA using uniform-complement mutation finds a solution x' where $f(x') \geq \frac{3}{4}m$ after an expected $O(mn)$ iterations.*

Proof. We first prove that after a polynomial number of mutation steps, the (1+1) EA has reached a solution x' such that for all $y \in N(x')$, $f(y) \leq f(x)$. Let x be the current state generated by the (1+1) EA . If there are no states $y \in N(x)$ with strictly improving fitness, then clearly $x' = x$.

On the other hand, suppose there exists at least one state $y \in N(x)$ with $f(y) > f(x)$. We show that in this case, the probability that mutation generates the specific improving state in $N(x)$ is $\Omega(1/n)$. We make the following case distinction.

Case 1. The complement of x is an improving state. In this case, uniform-complement mutation generates an improving state with probability at least 1/2.

Case 2. A Hamming neighbor of x is an improving state. The mutation operator generates a specific Hamming neighbor with probability

$$\frac{1}{2}\left[\frac{1}{n}\left(1-\frac{1}{n}\right)^{n-1}\right] \geq \frac{1}{2en} = \Omega(1/n).$$

It follows that the waiting time to make an improvement from such a state is at most $O(n)$. The number of improvements made by the (1+1) EA is bounded by the cardinality of the codomain. Hence the (1+1) EA can reach a state x' such that no improving moves exist in $N(x')$ in expected time bounded by $O(mn)$.

It is now straightforward to show that Lemma 1 entails a bound on the fitness of x'. In particular, consider

$$\frac{1}{|N(x')|}\sum_{y \in N(x)} f(y) \leq f(x'),$$

since otherwise the assumption that no improving states exist in $N(x')$ would be contradicted. By Lemma 1,

$$\frac{1}{|N(x')|}\big(3m + (n-3)f(x')\big) \leq f(x'),$$

and since $|N(x')| = n+1$,

$$\frac{3m}{(n+1)} + \frac{(n-3)}{(n+1)}f(x') \leq f(x').$$

It follows from simple algebraic manipulation of the above inequality that $f(x') \geq \frac{3}{4}m$ which completes the proof. □

The above result depends on the fact that, on MAX-2-SAT, the complement operation can escape certain lower-quality states that the Hamming neighborhood cannot. It is possible to have a state with no improving Hamming neighbors that *does not* satisfy the quality bound of $\frac{3}{4}m$. As an example, consider a Boolean formula constructed as follows. Let $A_{i,j}$ be the 3-set of clauses defined on two variables v_i and v_j

$$A_{i,j} = \{(\neg v_i \vee \neg v_j), (\neg v_i \vee v_j), (v_i \vee \neg v_j)\}.$$

We construct a MAX-2-SAT instance on $2q$ variables by taking the union of q 3-sets of clauses

$$A_{1,2} \cup A_{3,4} \cup \cdots \cup A_{2q-1,2q}.$$

Thus for this instance, $m = 3q$. Let $\hat{x} = (111 \cdots 1)$ be the length-$2q$ bitstring of all ones. The assignment corresponding to $\hat{x}$ satisfies the two clauses in each set $A_{i,j}$ that contain non-negated variables.

Consider a Hamming neighbor of $\hat{x}$ that is generated by flipping some bit i that, without loss of generality, corresponds to variable v_i of the instance. Since v_i only appears in clauses that are in $A_{i,j}$, it suffices to inspect the effect on $A_{i,j}$. In this case, the clause $(\neg v_i \vee \neg v_j)$ becomes satisfied, while the clause $(v_i \vee \neg v_j)$ becomes unsatisfied. The clause $(\neg v_i \vee v_j)$ remains satisfied and so the number of satisfied clauses of the instance remains the same under the Hamming neighbor. Hence no Hamming neighbor of $\hat{x}$ can be an improving state. Since $\hat{x}$ satisfies 2 clauses in each set $A_{i,j}$, we have

$$f(\hat{x}) = 2q = \frac{2}{3}m < \frac{3}{4}m.$$

In this example, however, the state generated by taking the complement of $\hat{x}$ satisfies all m clauses.

3.2 Standard parameterization

Assuming maximization, given a combinatorial optimization problem Q with maximum evaluation OPT_Q, the *standard parameterization* of Q is the parameterized problem where, given an instance of Q of size n and a parameter k, the task is to decide whether $OPT_Q \geq k$ [6]. Using the previous results, we now show that the (1+1) EA, under uniform-complement mutation, is a fixed-parameter evolutionary algorithm with respect to the standard parameterization of MAX-2-SAT.

Theorem 1. *The (1+1) EA using uniform-complement mutation solves the standard parameterization of MAX-2-SAT with n variables and m clauses in $O\left(\max\{mn, k^{2.67 \cdot k}\}\right)$ expected time.*

Furthermore, after $O(\max\{mn^2, nk^{2.67 \cdot k}\})$ iterations, the (1+1) EA using uniform-complement mutation has solved the standard parameterization of MAX-2-SAT almost surely.

Proof. Let T denote the random variable that corresponds to the first time that the (1+1) EA using uniform-complement mutation has solved the standard parameterization of MAX-2-SAT.

We make the following case distinction. First, suppose that $m \geq \frac{4}{3}k$. In this case, by Lemma 2, (1+1) EA has found a solution $f(x) \geq \frac{3}{4}m \geq k$ after an expected $O(mn)$ steps and hence $E(T) = O(mn)$. Now suppose that $m < \frac{4}{3}k$. Since there are at most $2m$ variables, the number of variables n is bounded by $\frac{8}{3}k$. The probability that the (1+1) EA transforms an arbitrary assignment into an optimal assignment is at least $(1/n)^n \geq ((8/3)k)^{-(8/3)k}$. In this case $E(T) = O(k^{2.67 \cdot k})$ which is a function that depends only on k.

For the tail bound, we need only to appeal to the Markov inequality (see, e.g., [2]) which states that, for a non-negative random variable T, $\Pr(T \geq \lambda E(T)) \leq \lambda^{-1}$ for all $\lambda \geq 1$. Suppose $m \geq \frac{4}{3}k$. Letting $c \geq 1$ be an arbitrary constant, by the Markov inequality we have

$$\Pr(T \geq cmn^2) \leq 1/cn.$$

In the case that $m < \frac{4}{3}k$,

$$\Pr(T \geq cnk^{2.67 \cdot k}) \leq 1/cn.$$

Hence the probability that the (1+1) EA has solved the standard parameterization in $O(\max\{mn^2, nk^{2.67 \cdot k}\})$ steps is $1 - o(1)$. $\square$

Note that in the case of the standard parameterization, a fixed-parameter evolutionary algorithm can be transformed into a Monte Carlo fpt-algorithm as follows. If $OPT_Q \leq k$, then the probability that the fixed-parameter evolutionary algorithm finds the optimal solution in $2E(T)$ is at least $1/2$ by the Markov inequality. On the other hand, if $OPT_Q > k$, no solution of fitness at most k exists. Therefore running the evolutionary algorithm for twice the expected runtime yields a Monte Carlo fpt-algorithm for the decision problem.

4. MAX-(2,3)-SAT

We now turn our attention to more interesting parameterizations. In particular, we introduce a variant of MAX-2-SAT that we can analyze with respect to a parameter that somehow measures the source of complexity inherent in the structure of the problem itself.

The MAX-(2,3)-SAT problem is a restricted class of MAX-2-SAT in which each variable appears in at most 3 clauses of the formula. Though slightly more restrictive, the MAX-(2,3)-SAT problem is still NP-hard [12]. Thus, assuming P $\neq$ NP, the expected runtime of an evolutionary algorithm must be superpolynomial. We now perform a parameterized analysis of the (1+1) EA on the MAX-(2,3)-SAT problem in order to express the runtime as a function that isolates the source of the exponential complexity in a parameter.

Denote as $G(V, E)$ the *variable graph* of an instance where V corresponds to the set of variables and

$$E = \{\{u, v\} \subset V : \text{ u and v appear together in a clause}\}.$$

We define the *diameter* of a (possibly disconnected) graph $G(V, E)$ as the maximum shortest-path distance in any of its connected components.

Lemma 3. *Consider a MAX-(2,3)-SAT formula in which the diameter of the variable graph is bounded by k. Then for any variable $v \in V$, the size of the connected component in $G(V, E)$ which contains v is bounded by $3 \cdot 2^k - 2$.*

Proof. Let v be an arbitrary variable. Denote as $C(v) \subseteq V$ the set of all nodes reachable on a path from v in G. Consider the breadth-first tree in G rooted at v. Since the degree of each vertex is bounded by 3 and the depth of the tree is bounded by k, there can be at most 3 children of the root, and each internal node can have at most 2 children. The number of nodes in this tree is hence bounded by

$$1 + 3 + 3 \cdot 2 + \cdots + 3 \cdot 2^{k-1} = 3 \cdot 2^k - 2 \geq |C(v)|.$$

Since the subgraph of $G(V,E)$ induced by $C(v)$ corresponds to the connected component containing v, the proof is complete. □

4.1 An XP evolutionary algorithm

Using Lemma 3, we can prove that as long as the diameter k of the variable graph is not too large, improving mutations are somehow close by in terms of Hamming distance. This results in an XP evolutionary algorithm in the parameter k, meaning that the expected runtime is bounded by a function of the form $n^{g(k)}$ where g depends only on k. We point out that the following theorem also follows from a result due to Wegener and Witt [15, Theorem 5.2] on separable quadratic functions with the slight modification that the size of the codomain of the fitness function can be restricted to $O(m)$.

Theorem 2. *Let F be a MAX-(2,3)-SAT formula where the diameter of the variable graph is bounded by k. Then the expected runtime of the (1+1) EA on F is bounded by $O(mn^{3\cdot 2^k-2})$. In other words, the (1+1) EA is an* XP *evolutionary algorithm for MAX-(2,3)-SAT.*

PROOF. The set of variables can be partitioned into sets $\mathcal{C} = \{C_1, C_2, \dots, C_q\}$ representing the connected components in G. Consider the i-th component C_i. Denote as $f_i(x)$ the number of clauses containing a variable from C_i satisfied by the assignment corresponding to x and let

$$a_i = \max_{z\in\{0,1\}^n} f_i(z).$$

Since $\mathcal{C}$ partitions the clause set, optimal value of f is $a_1 + a_2 + \cdots + a_q$.

As long as $f(x)$ is suboptimal, there exists a component C_i such that $f_i(x) < a_i$. We show that in this case there always exists an improving move within a bounded Hamming distance. Let $z \in \{0,1\}^n$ be a string such that $f_i(z) = a_i$. Denoting as $x[j]$ the j-th component of the bitstring x and v_j the corresponding Boolean variable in V, we define the bitstring x' as follows.

$$x'[j] = \begin{cases} z[j] & \text{if } v_j \in C_i; \\ x[j] & \text{otherwise.} \end{cases}$$

It follows that $f_i(x') = f_i(z) = a_i$. Moreover, $\mathcal{C}$ partitions V, so $f_j(x) = f_j(x')$ for all $j \in [q] \setminus \{i\}$. Thus we have $f(x') = f(x) - f_i(x) + a_i$ and therefore $f(x') > f(x)$.

Let d denote the number of bits that differ between x and x'. The probability that the (1+1) EA transforms x into x' under uniform mutation is $(1/n)^d(1-1/n)^{n-d}$. Invoking Lemma 3, $d \le |C_i| \le 3\cdot 2^k - 2$. Hence, as long as x is suboptimal, the probability of making an improving move is bounded below by $e^{-1}n^{-(3\cdot 2^k-2)}$. The number of steps until an improving move is found is thus distributed geometrically with expectation bounded by $O(n^{3\cdot 2^k-2})$. Finally, there are at most m suboptimal values of f, providing the claimed bound. □

4.2 A fixed-parameter evolutionary algorithm

In this section, we investigate a method to produce a fixed-parameter tractable algorithm for the MAX-(2,3)-SAT problem, yielding a runtime exponential in k but polynomial in n.

Following the format of the previous section, we consider the sets $\mathcal{C} = \{C_1, C_2, \dots, C_q\}$ representing the connected components in $G(V,E)$. We use $f_i(x)$ to represent the number of clauses in component C_i satisfied by x, and a_i to represent the optimum value of f_i. We define the component which contains the Boolean variable v_j to be $C(v_j)$.

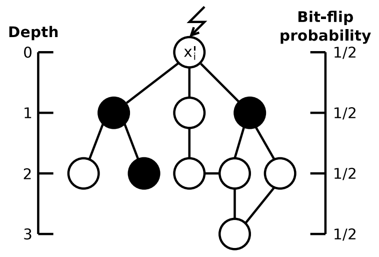

Figure 1: Probability characteristics of the Simple FPT (1+1) EA . Variable x'_i at depth 0 is the selected variable. White variables are flipped, black variables are unflipped.

Algorithm 3: The FPT (1+1) EA.

```
1 Choose $x$ uniformly at random from $\{0,1\}^n$;
2 repeat forever
3     $x' \leftarrow x$;
4     Select $i \in [n]$ uniformly at random;
5     Let $v_i$ be the Boolean variable corresponding to
      element $i$ of $x$;
6     Flip all bits of $x'$ corresponding to $v_j \in C(v_i)$ with
      probability $\frac{1}{2}$;
7     if $f(x') \ge f(x)$ then $x \leftarrow x'$
```

Algorithm 3 uses the (1+1) EA structure as defined in Section 2. The mutation used in this FPT algorithm selects a bit of the solution uniformly at random and flips all bits within the selected variable's set with probability 1/2, as demonstrated by Figure 1. If the resultant solution is at least as fit as the original solution, the new solution replaces the original solution.

Theorem 3. *Let F be a MAX-(2,3)-SAT formula in which the diameter of the variable graph is bounded by k. Then the expected runtime of the FPT (1+1) EA on F is bounded by $O(n\log n\cdot 2^{(3\cdot 2^k-2)})$.*

PROOF. We analyze the waiting time to generate a solution that satisfies all the clauses in each connected component. Given a state x, a component C_i is *solved* by x if $f_i(x) = a_i$. Otherwise, it is *unsolved* by x. A solution is optimal if and only if it solves all the connected components of $G(V,E)$. From line 5 of Algorithm 3, if a variable is selected for flipping, each variable in its connected component in $G(V,E)$ will also be flipped with probability 1/2. If the mutation process selects a variable in an unsolved component C_i, the probability of generating a state that also solves C_i (and thus increases the overall fitness) is at least

$$(1/2)^{|C_i|} \ge (1/2)^{3\cdot 2^k-2}$$

since Lemma 3 appropriately bounds $|C_i|$.

Suppose that $j > 0$ connected components are unsolved by x. The probability that mutation selects a variable that belongs to an unsolved component is j/n. Hence the probability of producing a state that also solves at least one unsolved component is

$$\frac{j}{n} \cdot 2^{-(3\cdot 2^k-2)}.$$

Since only monotonically improving moves are accepted and $G(V,E)$ has at most n connected components, the total waiting time until all components are solved is bounded above by the series

$$\sum_{j=1}^{n} \frac{n}{j} \cdot 2^{3\cdot 2^k-2} = O(n \log n \cdot 2^{3\cdot 2^k-2}),$$

yielding the claimed result. □

5. ALTERNATIVE FPT ALGORITHMS

When the connected components of the variable graph are large, flipping all the variables within a component with constant probability is somewhat disruptive and thus contradicts the philosophy of a localized mutation operator since many variables are likely to be changed at once. Indeed, when there is a single connected component, Algorithm 3 essentially resorts to a uniform random search in $\{0,1\}^n$. To address this, we introduce in this section three alternative fixed-parameter evolutionary algorithms which make less disruptive changes by attenuating the probability of changing variables that are further away in $G(V,E)$ from the variable selected for mutation.

5.1 Modified FPT (1+1) EA

The Modified FPT (1+1) EA utilizes probability attenuation in order to achieve a more natural solution that flips nodes with a lesser probability the further they are from the selected variable. This creates a more practical approach, albeit one with a greater upper bounds on the expected time to completion.

Algorithm 4: The Modified FPT (1+1) EA.

```
1  Choose $x$ uniformly at random from $\{0,1\}^n$;
2  repeat forever
3      $x' \leftarrow x$;
4      Select $i \in [n]$ uniformly at random;
5      Let $v_i$ be the Boolean variable corresponding to
       element $i$ of $x'$;
6      $S_u \leftarrow \{v_i\}$;
7      $S_v \leftarrow \emptyset$;
8      $c \leftarrow 1$;
9      while $S_u \neq \emptyset$ do
10         Flip all bits of $x'$with corresponding variables in
           $S_u$ with probability $\left(\frac{1}{2}\right)^c$;
11         $S_v \leftarrow S_v \cup S_u$;
12         $c \leftarrow c+1$;
13         $S_u \leftarrow \{v \in V \setminus S_v : \exists(\{v,v'\} \in E \wedge v' \in S_v)\}$;
14     if $f(x') \geq f(x)$ then $x \leftarrow x'$
```

The Modified FPT (1+1) EA detailed in Algorithm 4 has a decreasing probability of flipping bits the further they are from the selected bit. The mutation maintains a set of visited nodes S_v, and determines the set of unvisited adjacent nodes S_u. The bits in x' corresponding to Boolean variables in S_u are flipped with probability 1/2 raised to the power of the distance from the originally selected variable v_i, as demonstrated by Figure 2. If the modified solution is at least as fit as the original solution, the new solution replaces the original solution.

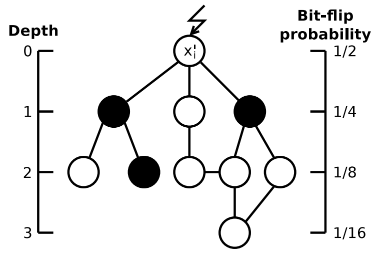

Figure 2: Probability characteristics of the Modified FPT (1+1) EA . Variable x_i' at depth 0 is the selected variable. White variables are flipped, black variables are unflipped.

Theorem 4. *Let F be a MAX-(2,3)-SAT formula in which the diameter of the variable graph is bounded by k. Then the expected runtime of the modified FPT (1+1) EA on F is bounded by* $O(n \log n \cdot 2^{(3k\cdot 2^k-2k)})$.

Proof. As the diameter of the variable graph is bounded by k, the maximum graph distance from any selected variable to any other variable within the component is k. Thus, the probability to flip each bit is bounded below by $(1/2)^k$. With this, we may determine the runtime in the same manner as in Theorem 3.

Again, we say a connected component C_i is solved by a state x if $f_i(x) = a_i$. Suppose a variable in an unsolved component C_i is selected during mutation. The probability in this case of generating a state which solves C_i is at least

$$((1/2)^k)^{|C_i|} \geq (1/2)^{(3\cdot 2^k-2)k}$$

where the bound on $|C_i|$ comes from Lemma 3.

If $j > 0$ components are unsolved by x, the probability of producing a state that solves at least one of the components is

$$\frac{j}{n} \cdot 2^{(-(3\cdot 2^k-2)k)}.$$

Since there are at most n connected components in $G(V,E)$, the waiting time until all components are solved is bounded by the series

$$\sum_{j=1}^{n} \frac{n}{j} \cdot 2^{(3\cdot 2^k-2)k} = O(n \log n \cdot 2^{3k\cdot 2^k-2k}),$$

thus providing the claimed runtime bound. □

5.2 Propagation FPT (1+1) EA

In this section, we propose two final FPT (1+1) EAs for the MAX-(2,3)-SAT problem. The algorithms presented by Algorithm 5 and 6 utilize probability propagation. Rather than flipping arbitrary nodes based on their distance from the selected variable, these algorithms flip only variables adjacent to previously flipped variables. This results in an algorithm with an easier implementation.

Algorithm 5, the Propagation FPT (1+1) EA, uses a local probability for each variable of 1/2 rather than the global

Algorithm 5: The Propagation FPT (1+1) EA.

```
Choose $x$ uniformly at random from $\{0,1\}^n$;
repeat forever
    $x' \leftarrow x$;
    Select $i \in [n]$ uniformly at random;
    Let $v_i$ be the Boolean variable corresponding to element $i$ of $x'$;
    $S_u \leftarrow \{v_i\}$;
    $S_v \leftarrow \emptyset$;
    while $S_u \neq \emptyset$ do
        $S_f \leftarrow \emptyset$;
        foreach $v \in S_u$ do
            Choose $r \in [0,1]$ uniformly at random;
            if $r < \frac{1}{2}$ then
                Flip the bit of $x'$ corresponding to $v$;
                $S_f \leftarrow S_f \cup \{v\}$;
        $S_v \leftarrow S_v \cup S_u$;
        $S_u \leftarrow \{v \in V \setminus S_v : \exists (\{v,v'\} \in E \wedge v' \in S_f)\}$;
    if $f(x') \geq f(x)$ then $x \leftarrow x'$
```

calculation from Algorithm 4. As before, the mutation maintains a set of visited nodes S_v, and determines the set of unvisited adjacent nodes S_u. In Algorithm 5, however, bits are flipped with probability 1/2, with flipped bits stored in a third set S_f. Propagation to the adjacent unvisited nodes only occurs if the current bit flips. The bits in x' corresponding to Boolean variables in S_u are thus flipped with probability 1/2 raised to the power of the distance from the originally selected variable v_i only if the previous bit was flipped. An example of this behaviour can be seen in Figure 3. If the modified solution is as least as fit as the original solution, the new solution replaces the original solution.

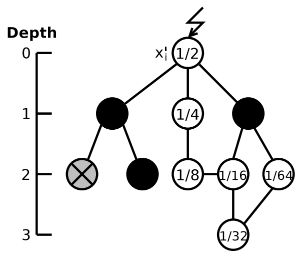

Figure 3: Probability characteristics of the Propagation FPT (1+1) EA . Variable x'_i at depth 0 is the selected variable. White variables are flipped, black variables are unflipped. The gray variable at depth 2 is unreachable and hence can not be flipped.

Theorem 5. *Let F be a MAX-(2,3)-SAT formula in which the diameter of the variable graph is bounded by k. Then the expected runtime of the propagation FPT (1+1) EA on F is bounded by $O(n \log n \cdot 2^{(3\cdot 2^k-2)^2})$.*

Proof. Suppose a variable v is selected for mutation. The variable negation process propagates out from v such that each subsequently flipped variable lies on a simple path of already-flipped variables of length d to v in $G(V,E)$. During this process, the variable is flipped only if its neighbor on the path to v was flipped, and in that case with probability 1/2. Therefore, the probability that the variable is flipped is $(1/2)^{d+1}$ for some $d > 0$. Since the length of a simple path in $G(V,E)$ is bounded above by the cardinality of its largest connected component, Lemma 3 applies and the probability the process flips a variable in the same connected component as v is at least $(1/2)^{3\cdot 2^k-2}$.

Let x be a suboptimal solution. If a variable in an unsolved component C_i is selected during mutation, the probability of generating a state which also solves C_i is at least

$$((1/2)^{3\cdot 2^k-2})^{|C_i|} \geq (1/2)^{(3\cdot 2^k-2)^2}.$$

The remainder of the proof follows the proof of Theorem 4 by imposing a bound on the probability of generating a new state x' in which at least one more component is solved than x. Again, if there are j unsolved components, the waiting time until such an improving move is generated is at most $(n/j)\cdot 2^{(3\cdot 2^k-2)^2}$. Since the number of connected components in $G(V,E)$ is at most n, summing over the waiting times yields the bound $O(n \log n \cdot 2^{(3\cdot 2^k-2)^2})$. □

This result is undesirable since it is somewhat larger than the runtime bound on Algorithm 4 given by Theorem 4. This can be solved by imposing a limit on the length of the path to the initial variable. Algorithm 6 performs this function by continuing to visit but not flipping variables along the path after an unflipped variable.

Algorithm 6: The Modified Propagation FPT (1+1) EA.

```
Choose $x$ uniformly at random from $\{0,1\}^n$;
repeat forever
    $x' \leftarrow x$;
    Select $i \in [n]$ uniformly at random;
    Let $v_i$ be the Boolean variable corresponding to element $i$ of $x'$;
    $S_u \leftarrow \{v_i\}$;
    $S_v \leftarrow \emptyset$;
    $S_d \leftarrow \emptyset$;
    while $S_u \neq \emptyset$ and $S_u \not\subset S_d$ do
        $S_f \leftarrow \emptyset$;
        foreach $v \in S_u \setminus S_d$ do
            Choose $r \in [0,1]$ uniformly at random;
            if $r < \frac{1}{2}$ then
                Flip the bit of $x'$ corresponding to $v$;
                $S_f \leftarrow S_f \cup \{v\}$;
        $S_v \leftarrow S_v \cup S_u$;
        $S_u \leftarrow \{v \in V \setminus S_v : \exists (\{v,v'\} \in E \wedge v' \in S_v)\}$;
        $S_d \leftarrow S_d \cup (S_u \setminus S_f)$;
    if $f(x') \geq f(x)$ then $x \leftarrow x'$
```

This modification is implemented simply by marking the children of unflipped nodes as disallowed, unable to flip. By maintaining another set of disallowed variables S_d, the propagation algorithm investigates the variables in a tree-like search, maintaining the k bounds for the distance from the selected variable to any other. This modification creates a much stricter behaviour as demonstrated in Figure 4. Note that the probability propagation follows a directed acyclic path from the root to strictly deeper nodes. The probability of any node being flipped is dependent on at least one of its parents being flipped. The joint distribution of any component (or indeed any set of components) C can thus be written as

$$p(C) = \prod_{v \in C} p(v|\mathrm{pa}(v)),$$

where $\mathrm{pa}(v)$ is the set of parents of v, demonstrating that the graph is a Bayesian network.

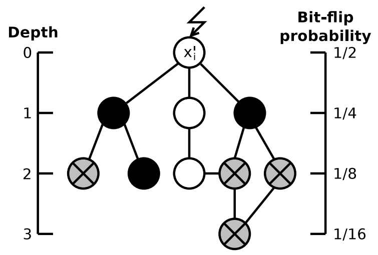

Figure 4: Probability characteristics of the Modified Propagation FPT (1+1) EA . Variable x_i' at depth 0 is the selected variable. White variables are flipped, black variables are unflipped. The gray elements at depth 2 and 3 are located after an unflipped variable and hence will also not be flipped. Note that one of the gray variables has an adjacent flipped node; to be viable for flipping, a variable must have a flipped parent at one depth shallower.

Theorem 6. *Let F be a MAX-(2,3)-SAT formula in which the diameter of the variable graph is bounded by k. Then the expected runtime of the modified propagation FPT (1+1) EA on F is bounded by $O(n \log n \cdot 2^{(3k \cdot 2^k - 2k)})$.*

Proof. Each variable within a component is flipped with probability 1/2 if and only if its parent was also flipped. As the maximum path length of the propagation is bounded by k, this provides a lower bound on the probability for each bit flip of $(1/2)^k$. Using this probability, the rest of this proof proceeds exactly as in Theorem 4. □

6. CONCLUSION

The MAX-SAT problem is one of the most famous NP-hard optimization problems. The parameterized runtime analysis of evolutionary algorithms provides new insights into the behavior of evolutionary algorithms because it allows one to measure the difficulty of instances of an NP-hard combinatorial optimization problem by taking an additional parameter into account. In this paper, we have carried out a parameterized analysis of evolutionary algorithms for the MAX-2-SAT problem. Using results from the field of elementary landscapes, we have shown that the well-known (1+1) EA is a fixed-parameter algorithm with respect to the standard parameterization. Furthermore, we have analyzed the class of MAX-(2,3)-SAT problems with respect to the diameter of the variable graph. Our results show that the classical (1+1) EA is an XP-algorithm for this problem. Furthermore, we have shown that a mutation operator based on the structure of the variable graph leads to fixed-parameter evolutionary algorithms for this problem.

7. REFERENCES

[1] A. Auger and B. Doerr, editors. *Theory of Randomized Search Heuristics: Foundations and Recent Developments.* World Scientific, 2011.

[2] B. Doerr. Analyzing randomized search heuristics: Tools from probability theory. In Auger and Doerr [1].

[3] M. Dorigo and T. Stützle. *Ant Colony Optimization.* MIT Press, 2004.

[4] R. G. Downey and M. R. Fellows. *Parameterized Complexity.* Springer, 1999.

[5] A. Eiben and J. Smith. *Introduction to Evolutionary Computing.* Springer, 2nd edition, 2007.

[6] J. Flum and M. Grohe. *Parameterized complexity theory.* Springer-Verlag, 2006.

[7] M. R. Garey, D. S. Johnson, and L. J. Stockmeyer. Some simplified NP-complete graph problems. *Theoretical Computer Science*, 1(3):237–267, 1976.

[8] L. K. Grover. Local search and the local structure of NP-complete problems. *Operations Research Letters*, 12:235–243, 1992.

[9] S. Kratsch, P. K. Lehre, F. Neumann, and P. S. Oliveto. Fixed parameter evolutionary algorithms and maximum leaf spanning trees: A matter of mutation. In R. Schaefer, C. Cotta, J. Kolodziej, and G. Rudolph, editors, *Eleventh International Conference on Parallel Problem Solving from Nature (PPSN XI)*, volume 6238 of *Lecture Notes in Computer Science*, pages 204–213. Springer, 2010.

[10] S. Kratsch and F. Neumann. Fixed-parameter evolutionary algorithms and the vertex cover problem. In *Proceedings of the Eleventh Annual Conference on Genetic and Evolutionary Computation*, GECCO '09, pages 293–300, New York, NY, USA, 2009. ACM.

[11] F. Neumann and C. Witt. *Bioinspired Computation in Combinatorial Optimization – Algorithms and Their Computational Complexity.* Springer, 2010.

[12] V. Raman, B. Ravikumar, and S. S. Rao. A simplified NP-complete MAXSAT problem. *Information Processing Letters*, 65(1):1–6, 1998.

[13] C. M. Reidys and P. F. Stadler. Combinatorial landscapes. *SIAM Review*, 44:3–54, 2002.

[14] P. F. Stadler. Toward a theory of landscapes. In R. Lopéz-Peña, R. Capovilla, R. García-Pelayo, H. Waelbroeck, and F. Zertruche, editors, *Complex Systems and Binary Networks*, pages 77–163. Springer Verlag, 1995.

[15] I. Wegener and C. Witt. On the analysis of a simple evolutionary algorithm on quadratic pseudo-Boolean functions. *Journal of Discrete Algorithms*, 3(1):61–78, 2005.