Analysis of a Gray-Box Operator for Vertex Cover

Samuel Baguley
Hasso-Plattner-Institute, Potsdam
University
Potsdam, Germany
Samuel.Baguley@hpi.de

Tobias Friedrich
Hasso-Plattner-Institute, Potsdam
University
Potsdam, Germany
Tobias.Friedrich@hpi.de

Timo Kötzing
Hasso-Plattner-Institute, Potsdam
University
Potsdam, Germany
Timo.Koetzing@hpi.de

Xiaoyue Li
Hasso-Plattner-Institute, Potsdam
University
Potsdam, Germany
Xiaoyue.Li@hpi.de

Marcus Pappik
Hasso-Plattner-Institute, Potsdam
University
Potsdam, Germany
Marcus.Pappik@hpi.de

Ziena Zeif
Hasso-Plattner-Institute, Potsdam
University
Potsdam, Germany
Ziena.Zeif@hpi.de

ABSTRACT
Combinatorial optimization problems are a prominent application area of evolutionary algorithms, where the (1+1) EA is one of the most investigated. We extend this algorithm by introducing some problem knowledge with a specialized mutation operator which works under the assumption that the number of 1s of a solution is critical, as frequently happens in combinatorial optimization. This slight modification increases the chance to correct wrongly placed bits while preserving the simplicity and problem independence of the (1+1) EA.

As an application of our algorithm we examine the vertex cover problem on certain instances, where we show that it leads to asymptotically better runtimes and even finds with higher probability optimal solutions in comparison with the usual (1+1) EA. Precisely, we compare the performance of both algorithms on paths and on complete bipartite graphs of size \( n \). Regarding the path we prove that, for a particular initial configuration, the (1+1) EA takes in \( \Theta(n^2) \) iterations while the modification reduces this to \( \Theta(n^3) \), and present experimental evidence that such a configuration is reached. Concerning the complete bipartite graph our modification finds the optimum in polynomial time with probability \( 1 - 1/\Omega(n^3) \) for every positive constant \( \xi < 1 \), which improves the known probability of \( 1 - 1/poly(n) \) for the (1+1) EA.

CCS CONCEPTS
- ♦ Computer systems organization → Embedded systems; Redundancy; Robotics; ♦ Networks → Network reliability.

KEYWORDS
evolutionary algorithm, runtime analysis, vertex cover

ACM Reference Format:
Samuel Baguley, Tobias Friedrich, Timo Kötzing, Marcus Pappik, and Ziena Zeif. 2022. Analysis of a Gray-Box Operator for Vertex Cover. In Genetic and Evolutionary Computation Conference (GECCO ’22), July 9–13, 2022, Boston, MA, USA. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3512290.3528848

1 INTRODUCTION
Evolutionary algorithms (EAs) are a diverse and highly versatile class of randomized search heuristics, and their effectiveness at solving combinatorial optimization problems has been recognized and widely studied since the nineties [18, Chap. 14]. Typically, EAs are implemented with naive mutation behavior, and adapt to a particular problem only via an associated fitness function. This simplicity has lead to success in many applications, but leaves room for improvement on specific problems.

The minimum vertex cover problem is an ideal target for evolutionary algorithms because it is NP-complete [9], and appears in a broad range of practical applications. Already in 1994 Bäck and Khuri [1] defined the canonical fitness function for EAs on vertex cover and provided experimental evidence that on some graph instances EAs can produce better approximate solutions than bespoke solvers. In the years that followed runtime analysis of EAs tended to be heuristic and experimental, but recently rigorous theoretical analyses have gained prominence. The various approaches to this include proving runtime bounds for approximate solutions [7, 15], analyzing performance on best- or worst-case graph instances [13, 14], and implementing genetic mechanisms like crossover [4]. It is also common for new research to define variations of well-studied EAs, like Oliveto et al.’s (1+1) EA with an additional ‘diversity maintenance mechanism’ [14], or a fitness function with multiple objectives [3, 10].

In the application of evolutionary algorithms to real world problems, the full black-box view of the fitness function (ignoring all additional information about its inner workings) is at most the initial prototype; significant improvements can be made by applying problem knowledge for designing tailored variation operators [16]. The resulting view is called gray-box optimization, and research tries to understand what general structural properties can be usefully employed in evolutionary computation [19]. We are interested in combinatorial problems where a certain subset has to be selected; typically, the number of elements in the chosen set is
rather important. For example, a minimum spanning tree contains exactly \( n - 1 \) edges, the vertex cover problem tries to minimize the number of vertices picked, and the independent set problem maximizes the selected vertices. While [11] argued that variation operators should be unbiased with respect to whether a 0 or a 1 encodes a specific property of a solution, the gray-box view of combinatorial problems wants the 1 to mean ‘in’ and 0 to mean ‘out’, and then design operators that can make use of this meaning. From the very general work of Rowe and Vose [17] we can see that only for problems which have symmetrical meaning in 1 and 0 can the operators be assumed unbiased.

In this paper we augment the (1+1) EA by allowing a second kind of mutation called a ‘balanced flip’, in which a bit is chosen uniformly at random, and then swapped with the bit of a neighboring vertex of opposite value, if one exists. At every iteration the augmented algorithm chooses between a balanced flip and a regular step of the (1+1) EA with equal probability. The ‘balanced flip’ behavior is already exhibited by the (1+1) EA on nearly-optimal bit-strings, the difference being that it takes an order of magnitude longer to perform each flip. In particular, the balanced flip operator could provide benefits in problem-specific and instance-specific cases. Thus the evolutionary algorithm presented here refines the (1+1) EA by incorporating its emergent behavior on inputs close to the optimum directly into the mutation operator; this is somewhat in the spirit of Giel and Wegener’s ‘local (1+1) EA’ [5]. The maximum matching problem considered by those authors is dual to vertex cover on bipartite graphs, and similar techniques appear in the proofs here, though differing in some technical details. The intention behind the Balanced (1+1) EA is not to compete with state-of-the-art solvers for vertex cover, but rather to show that an additional simple mutation in the (1+1) EA can yield reductions in runtime while preserving the generality which is the strength of EAs. This framework is called ‘gray-box’ because the additional mutation is motivated by specific problem knowledge. In recent years there has been research into gray-box operators, for example in [8, 12], but it remains a relatively unexplored area.

In Section 3 we compare the runtime of the (1+1) EA solving vertex cover on paths to that of the Balanced (1+1) EA. Our analysis of the (1+1) EA uses techniques similar to those in [5] to obtain an expected runtime of \( O(n^4) \), and we then apply those same techniques to the Balanced (1+1) EA to prove an expected runtime of \( O(n^4) \). We also give a partial proof of \( \Omega(n^4) \) expected runtime for the (1+1) EA, assuming that the algorithm at some point produces a bit-string with a long consecutive sequence of incorrectly assigned bits. We provide experimental evidence to motivate this condition, and delay a full proof to future work.

In Section 4 we consider the complete bipartite graph \( K_{n,n} \), which is a common instance class of interest when it comes to evolutionary algorithms. Under the assumption that the ratio \( R/L \) is at least 2, the (1+1) EA is known to have exponential runtime with at least polynomial probability [3, Theorem 5]. In contrast, the Balanced (1+1) EA has far better performance on this graph instance, with exponentially small probability of not reaching the optimum in polynomial time.

2 PRELIMINARIES

In this paper we demonstrate that incorporating even small amounts of problem knowledge can significantly reduce the expected runtime of the (1+1) EA solving vertex cover. We provide a new evolutionary algorithm that we call Balanced (1+1) EA (Algorithm 2) that we compare with the classical (1+1) EA (Algorithm 1). Proofs that are omitted from the article can be found in the full version [2].

We introduce briefly the graph terminology used in this paper. A graph will be denoted \( G = (V, E) \), and for any subgraph \( H \) of \( G \), we write \( V(H) \) and \( E(H) \) for vertices and edges of \( H \) respectively. We denote an edge \( e = \{u, v\} \in E \) by \( uv \) and the neighborhood of a vertex \( v \in V \) in \( G \) by \( N(v) = \{u \in V \mid uv \in E\} \). Lastly, we define the length of a path \( P = (V, E) \) to be number of vertices \( |V| \).

A vertex cover for a given graph \( G = (V, E) \) is a subset of vertices \( V' \subseteq V \) which satisfies \( V' \cap e \neq \emptyset \) for all \( e \in E \). A solution to the vertex cover problem is a vertex cover \( V' \) which has minimal cardinality \( |V'| \), in the sense that no cover of smaller cardinality exists.

Depending on the context, we shall use \( X \) to denote either the stochastic process induced by one of the algorithms or an arbitrary state of that process \( X \in \{0, 1\}^n \). We make sure that the meaning of \( X \) is always clarified. For \( t \in \mathbb{N}, X_t \) denotes the configuration after \( t \) iterations, and \( X_0 \) corresponds to the initial configuration of the process \( X \). Both algorithms studied in this work have search space \( \{0, 1\}^n \) of bit strings of length \( n \), where \( n = |V| \). If \( V = (v_1, \ldots, v_n) \) is some enumeration of the vertices of \( G \), then a bit string \( X = (x_1, \ldots, x_n) \in \{0, 1\}^n \) corresponds to a subset of \( V_X \subseteq V \) via the mapping \( x_i \in V_X \iff x_i = 1 \). We call a bit string \( X \) feasible if \( V_X \) is a vertex cover for \( G \), and infeasible otherwise. If \( V_X \) is a minimal vertex cover then \( X \) is called optimal. Because the relationship between \( X \) and \( V_X \) is one-to-one, we will abuse notation and use \( X \) to represent the bit string or subset of vertices interchangeably. Letting \( u(X) := |\{e \in E \mid V_X \cap e = \emptyset\}| \) be the number of uncovered edges in \( X \), we have that \( X \) is feasible if and only if \( u(X) = 0 \). Further, we define \( |X|_0 - |X|_1 \) to be the number of ones in \( X \), \( |X|_0 = n - |X|_1 \) to be the number of zeroes, and \( OPT = |X_{opt}|_1 \) to be the number of ones in the minimal vertex cover.

The fitness of \( X \) is given by the integer \( f(X) \), where

\[
f: \{0, 1\}^n \to \mathbb{N} : X \mapsto |X|_1 + (n + 1)u(X)
\]

defines the fitness function. The multiplicative factor \( n + 1 \) ensures that the (1+1) EA and Balanced (1+1) EA prioritize covering any uncovered edge over reducing the size of \( X \). This fitness function (and slight variations of it) has been standard for evolutionary algorithms solving the vertex cover problem since [1]. With this in hand, the (1+1) EA can be defined.

Because the (1+1) EA has positive probability of sampling any point of \( \{0, 1\}^n \) in step 3, regardless of the current state \( X \), it almost surely produces a solution to the vertex cover problem in finite time. A fundamental property of interest, and the main focus of this article, is the expected time to find this solution. If the optimum cover is unique then a naïve estimate based on this behavior yields an extremely poor expected runtime of \( n^6 \) for the (1+1) EA. Worst-case analysis, for example in [3, Theorem 5], suggests that no better can be expected for general \( G \).

We now introduce a version of the (1+1) EA which chooses with equal probability between two sampling behaviors, one of which is
Algorithm 1: (1+1) EA

1. Choose $X \in \{0, 1\}^n$ (if unspecified, uniformly at random)
2. While stopping criterion not met do
3. \[ Y \leftarrow \text{flip each bit of } X \text{ independently with probability } \frac{1}{n} \]
4. \[ \text{if } f(Y) \leq f(X) \text{ then} \]
5. \[ X \leftarrow Y \]
6. End

Algorithm 2: Balanced (1+1) EA

1. Choose $X \in \{0, 1\}^n$ (if unspecified, uniformly at random)
2. While stopping criterion not met do
3. \[ p \leftarrow \text{pick a number u.a.r. in the interval } (0, 1) \]
4. \[ \text{if } p \leq \frac{1}{2} \text{ then} \]
5. \[ Y \leftarrow \text{flip each bit of } X \text{ independently with probability } \frac{1}{n} \]
6. End
7. Else
8. \[ v \leftarrow \text{pick a vertex u.a.r from } V(G) \]
9. \[ N_v \leftarrow \{v' \in N(v) \mid x_0 \neq x_{v'}\} \]
10. \[ \text{if } N_v \neq \emptyset \text{ then} \]
11. \[ u \leftarrow \text{pick a vertex u.a.r from } N_v \]
12. \[ Y \leftarrow \text{flip } x_u \text{ and } x_v \]
13. End
14. Else
15. Go to line 2
16. End
17. If $f(Y) \leq f(X)$ then
18. \[ X \leftarrow Y \]
19. End

2.1 Feasible bit strings

In order to analyze the expected runtimes of the (1+1) EA and Balanced (1+1) EA, we first use a multiplicative drift theorem to bound the time it takes for $X$ to become feasible.

Lemma 2.1. Let $G = (V, E)$ be a graph with $n = |V|$ vertices, and let $T$ be the first time that the (1+1) EA samples a feasible string. Then $E[T] \leq en(\ln(n)+1/2)$ and for $k > 0, P[T > 2ekn(\ln(n) + 1/2)] \leq 2^{-k}$.

Further, if $S$ is the first time that the Balanced (1+1) EA samples a feasible string, then $E[S] \leq 2en(\ln(n) + 1/2)$ and for $k > 0, P[S > 4ekn(\ln(n) + 1/2)] \leq 2^{-k}$.

We frequently make use of Lemma 2.1 throughout different parts of this paper. Note that a desirable property of the update criteria of both Algorithms 1 and 2 is that, once a feasible solution is found, the algorithms never leave the space of feasible solutions again.

3 VERTEX COVER ON PATHS

Although the worst-case expected runtime of the (1+1) EA is known to be exponential in $n$, [3, Theorem 5], far better runtime bounds can be obtained on simple graph instances. In this section we take $G$ to be a path of length $|V| = n$, with the $i$-th bit of $X$ corresponding to the $i$-th vertex of $G$, numbering sequentially along the path. By considering path instances, we can precisely analyse the runtime of the (1+1) EA, and then transfer that analysis to the Balanced (1+1) EA, showing that in expectation it performs better by a linear factor. It is reasonable to expect that the Balanced (1+1) EA also improves on the runtime of the (1+1) EA on other graph instances, but rigorous analysis of those will require more general tools than those we develop below.

In order to achieve the lower bound for either algorithm we require additional two assumptions, the first of which is defined in more detail in Section 3.2.1, and supported by experimental evidence in Fig. 1.

(A) With probability $\geq 1/2$, there exists a $t \geq 0$ such that $X_t$ is feasible with one more selected vertex than the optimum and contains a connected subpath of vertices which are not in their optimal state of size $\Theta(n)$.

(B) $|V| = n$ is odd.

Instead of (A), we could assume that $X$ has an initial distribution which is supported by the subset of $\{0, 1\}^n$ containing only feasible bit strings. Assumption (A) simply ensures that $X$ hits this subset with probability $1/2$, and since $X$ is a strong Markov process, restarting at this hitting time preserves the lower bound.

Assumption (B) is needed for particular arguments in Section 3.2.1, but seems unlikely to be necessary.

The two main theorems of this section are as follows.

Theorem 3.1. Let $P = (V, E)$ be a path of length $n = |V|$. In expectation, the (1+1) EA samples an optimal solution in $O(n^2)$ iterations, and in $\Omega(n^3)$ iterations if assumptions (A) and (B) hold true.

Theorem 3.2. Let $P = (V, E)$ be a path of length $n = |V|$. In expectation, the Balanced (1+1) EA samples an optimal solution in $\tilde{O}(n^3)$ iterations, and in $\Omega(n^4)$ iterations if assumptions (A) and (B) hold true.

3.1 Upper Bound Running Time

By Lemma 2.1, we know that the (1+1) EA finds a feasible solution in expectation in $O(n \log(n))$ iterations, given any starting distribution, including in particular the uniform distribution. Once a feasible solution is found, all states in following iterations are feasible as well. Hence our analysis can focus on the expected runtime given a feasible initial state. If $n$ is odd, there exists a unique optimal vertex cover of size $OPT = (n - 1)/2$, and if $n$ is even, there are multiple optimal covers of size $OPT = n/2$. We define the level of $X$ at time $t$ to be $|X_t| - OPT$, to parametrize the deviation of $X$ from the optimum. Our strategy is to compute the expected number of
iterations that \( X \) spends at any level \( \ell \), and then sum over all levels to obtain an upper bound for total runtime. For this purpose, we define \( Y_\ell \) to be the total number of iterations spent at level \( \ell \geq 0 \), i.e. \( Y_\ell = \sum_{t=0}^\ell 1[|X_t| = \ell + \text{OPT}] \), and let \( Y = \sum_{\ell=0}^{\lceil n/2 \rceil} Y_\ell \) be the first hitting time of level \( \ell = 0 \).

The main work of this section is in proving the following bounds on the expected time spent at level \( \ell \) - which may be zero - by the \((1+1)\) EA and Balanced \((1+1)\) EA respectively.

**Lemma 3.3.** The expected number of iterations that the \((1+1)\) EA spends at level \( \ell \) is of order \( O(n^3/\ell^2) \).

**Lemma 3.4.** The expected number of iterations that the Balanced \((1+1)\) EA spends at level \( \ell \) is of order \( O(n^3/\ell^2) \).

With these results in hand, we prove the following upper bound on the number of expected iterations before the \((1+1)\) EA samples an optimal solution for the first time.

**Theorem 3.5.** Let \( P = (V, E) \) be a path of length \( n = |V| \). Then the expected number of iterations that the \((1+1)\) EA needs to sample an optimal solution is of order \( O(n^5) \), and the expected number of iterations that the Balanced \((1+1)\) EA needs to sample an optimal solution is of order \( O(n^3) \).

### 3.1.1 Random Walk Coupling

The proofs of Lemmas 3.3 and 3.4 make use of a coupling argument, given below, between \( X \) and an independent random walk on a path of length \( O(n) \). This coupling is similar to one defined by Giel and Wegener [5], who studied a gray-box modification of the \((1+1)\) EA solving the maximum matching problem on paths, which due to König’s theorem is equivalent to vertex cover on all bipartite graphs. However, the restarting arguments appearing here are novel.

To take advantage of the coupling we first prove the following Lemma; an alternative proof of this result using drift theory can be found in Göbel et al. [6, Theorem 13].

**Lemma 3.6.** Let \( q \in (0, 1/2] \), and let \( Z \) be the symmetric Markov chain on \( \{0, 1, \ldots, d\} \) with transitions \( P(r,s) = P[Z_{t+1} = r | Z_t = s] \) defined by

\[
P(s,s-1) = \begin{cases} q & \text{if } 1 \leq s \leq d-1, \\ 2q & \text{if } s = d, \\ 0 & \text{if } s = 0, 
\end{cases}
\]

and \( P(s,s) = 1 - P(s,s-1) - P(s,s+1) \). This is the symmetric random walk with reflecting barrier at \( d \) and absorbing barrier at \( 0 \), with an additional probability to stay put at any vertex. Suppose \( Z_0 = d \) and \( T = \min\{t \geq 0 : Z_t = 0\} \) is the first time that \( Z \) hits 0. Then \( E[T] \leq d^2/2q \).

The unique optimal solution on the odd-length path - which will be the terminal value of \( X \) - has alternate vertices selected, starting with the second and ending with the \((n-1)\)-th. Similarly, both optima of the even-length path are similarly alternating, apart from a possible pair of adjacent ones. Let us suppose that \( X \) is at level \( \ell \) and is feasible. We shall begin by supposing that \( X \) is irreducible, and bound the time it takes to become reducible. Since \( X \) is feasible, no two neighbouring vertices are unselected, and since \( X \) is irreducible, there is no subpath of consecutive ones of length more than 2. An example for \( n = 11 \) is given below.

\[
\begin{array}{cccccccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \\
0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\
\end{array}
\]

Here \( X \) is at level \( \ell = 1 \), although in this example most nodes (from 3 to 9 inclusive) are not in their optimal state. Now we can define a dual process \( \tilde{X} \) on the path of length \( 2 + (n-1)/2 \), which is coupled to \( X \), in the following way: for \( i \in \{1, \ldots, (n-1)/2\} \),

\[
\tilde{X}(i) = X(2i) \cdot 1 \text{ [at least one of } X(2i-1) \text{ and } X(2i+1) \text{ is a } 1] = X(2i)(X(2i-1) + X(2i+1) - X(2i-1)X(2i+1)),
\]

and \( \tilde{X}(0) = X(1), \tilde{X}(n+1/2) = X(n) \). Then the dual state to the one above is

\[
\begin{array}{cccccccc}
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

If \( X \) is the \((1+1)\) EA, then conditional on \( X \) flipping at most two bits at a time - since \( X \) is feasible and irreducible, it cannot flip only one bit in a step - the ‘particles’ at positions 1 and 5 move independently like symmetric random walks (with reflective boundaries) until the time that they are adjacent, at which point \( X \) is reducible. If \( X \) is the Balanced \((1+1)\) EA, then the same random walk behavior is observed, but with differing transition probabilities. In this example there are only two such particles, since \( X \) is at level \( \ell = 1 \), but in general there will be \( 2\ell \). As long as \( X \) is not optimal, we define the minimum distance (measured by number of zero nodes) between any two particles in \( X \) to be the longest subpath of consecutive zeroes (corresponding to subpaths of alternating bits in \( X \)). This is itself a stochastic process on \( \{0, \ldots, d\} \), where \( d \leq 2\ell \). For ease of notation we denote this process by \( H \).

### 3.1.2 The \((1+1)\) EA: Proof of Lemma 3.3.

Take \( X \) to be the \((1+1)\) EA, and for the moment, let us condition on the event that the algorithm only flips 2 bits at every iteration. Under this conditioning, the process \( H \) can only make steps of size one in either direction. It need not be symmetric, but decreases (that is, \( H_{t+1} = H_t - 1 \)) with probability \( \geq 2p^2(1-p)^{n-2} \), and increases with probability \( \leq 2p^2(1-p)^{n-2} \). Thus the expected number of iterations taken for \( H \) to hit zero is less than the expected hitting time \( T \) of zero by the independent process \( Z \) on \( \{0, \ldots, d\} \) satisfying the conditions of Lemma 3.6 with \( q = 2p^2(1-p)^{n-2} \). Recalling that \( p = 1/n \) and \( d \leq n/2\ell \), the following corollary is immediate.

**Corollary 3.7.** \( E[T] \leq d^2/4(1-p)^{n-2} \leq n^4/\ell^2 \).

Thus we have an upper bound on the expected time for \( X \) to become reducible, conditional on the \((1+1)\) EA flipping no more than 2 bits at every step. Now we want to remove that conditioning. To do so, we modify the transitions of \( Z \) so that it returns to position \( d \) with probability \( 4p^4 \) at every step; this is an upper bound for the probability that \( (H_{t+1} - H_t) \geq 2 \), that is, that \( H \) increases by 2 or more in a single step. Let us denote this modified version of \( Z \) by \( \tilde{Z} \). The transitions of \( \tilde{Z} \) are

\[
\tilde{Z}_{t+1} = \begin{cases} \tilde{Z}_t + 1 & \text{with probability } 2p^2(1-p)^{n-2}, \\ \tilde{Z}_t - 1 & \text{with probability } 2p^2(1-p)^{n-2}, \\ d & \text{with probability } 4p^4, \\ \tilde{Z}_t & \text{otherwise}, \end{cases}
\]
When \( \tilde{Z}_t \in \{1, \ldots, d-1\} \), and similar in the obvious way for \( \bar{Z}_t = d \). In essence, \( \tilde{Z} \) is ‘rested’ every time \( H \) increases by 2 or more, and so the expectation of the hitting time \( T = \min\{t \geq 0 : \tilde{Z}_t = 0\} \) is greater than the expected time it takes for \( H \) to hit zero - that is, for \( X \) to become reducible.

The question to answer now is how many times \( \tilde{Z} \) will have to restart; we shall show that it is constant in expectation. Each ‘run’ of \( \tilde{Z} \), starting at position \( d \) and ending when it hits zero or is restarted, is independent from the others. They are also all identically distributed, with the same law as \( T \) from Corollary 3.7. The number of restarts is therefore a geometric random variable, say \( R \), and to calculate its mean we need only calculate the probability of success, that is, of \( \tilde{Z} \) hitting zero before it is sent back to \( d \).

The ‘restart time’ of \( \tilde{Z} \) is itself a geometric random variable \( S \), independent of the position of \( \tilde{Z} \), with success probability \( 4p^2 \). Therefore, for any \( r \in \mathbb{N}^+ \),

\[
\mathbb{P}[T < S] = \mathbb{P}[T \leq r < S] = \mathbb{P}[T \leq r] (1 - 4p^2)^r.
\]

We saw in Corollary 3.7 that \( \mathbb{E}[T] \leq n^4/t^2 \), and it follows from Markov’s inequality that \( \mathbb{P}[T \leq 2\mathbb{E}[T]] \geq 1/2 \). Therefore, taking \( r = 2n^4/t^2 \) for all \( t \geq 1 \), we have that

\[
\mathbb{P}[T < S] = \frac{1 - 4n^{-4}2n^4/t^2}{2} \rightarrow c \in (0, 1) \quad \text{as } n \to \infty.
\]

Then Wald’s equation yields that

\[
\mathbb{E}[\tilde{T}] \leq \mathbb{E}[T] \mathbb{E}[R] = \frac{n^4}{t^2 \mathbb{P}[T \geq S]} = \frac{n^4}{(1-c)t^2}.
\]

This gives an upper bound for the expected number of iterations it takes for \( H \) to hit zero - that is, \( X \) to become reducible - given that \( X \) started in level \( t \).

It remains to note that when \( X \) is in a reducible state, it has a positive probability of taking a step to an irreducible state but remaining at level \( t \). However, the probability of this occurring before \( X \) is reduced is bounded above by a constant,\(^4\) and so an argument similar to above using Wald’s equation and restarting gives that \( \mathbb{E}[Y_t] \leq kn^4/t^2 \) for some constant \( k > 1 \); this proves Lemma 3.3.

### 3.1.3 The \((1+1)\) EA: Proof of Lemma 3.4.

The same arguments as for the \((1+1)\) EA hold for the Balanced \((1+1)\) EA, with the difference that \( H \) decreases with larger probability. More precisely, with probability \( 1/2 \) the algorithm chooses a bit uniformly at random, and attempts to swap it with a neighbour of a different value. There are 4 balanced flips out of at most \( 2n \) which cause \( H \) to decrease; conditioning the algorithm on flipping no more than 2 bits in a single step, this leads to a decrease probability of

\[
q \geq \frac{1}{2} \left( \frac{4}{2n} + 4p^2(1-p)(n-2) \right).
\]

which again via Lemma 3.6 leads to a runtime bound of \( d^2/2q \leq n^3/t^2 \). The same restarting arguments as above then yield the result of Lemma 3.4.

\(^4\)This requires two simultaneous bit flips, and roughly has probability \( t/n^2 \), compared to \( 1/n \) for simply reducing. Markov’s inequality gives the constant.

### 3.2 Lower Bound Running Time

In this section, we prove the lower bound parts of Theorems 3.1 and 3.2. That is, given a suitably chosen starting configuration, the \((1+1)\) EA and Balanced \((1+1)\) EA require at least \( \Omega(n^3) \) iterations to find the minimum vertex cover on a path of odd length \( n \). Again we note that such instances \( G \) only have one minimal vertex cover, namely the set of vertices at even positions, which simplifies our analysis.

The technical core of our lower bound proof is the following general lemma. It essentially lower bounds the hitting time of a process that is dominated by a symmetric random walk. There are two main difference between Lemma 3.8 and most bounds that can be found in the literature. Firstly, the presented bound does not only hold in expectation but at least with constant probability and, secondly, the description of the process allows for arbitrarily large jumps, as long as their probability is bounded.

**Lemma 3.8.** Let \( d \in \mathbb{N} \) with \( d \geq 4 \ln(10) \) and let \( Z_t \in \mathbb{N} \) be a stochastic process such that, for some \( p \in (0, 1/2) \) and \( q \in [0, 1) \) with \( q + 2p \leq 1 \), it holds that

1. \( \forall s \in \mathbb{N} \cap [0, d-1] : \mathbb{P}[Z_{t+1} = s + 1 | Z_t = s] \leq p 
2. \( \forall s \in \mathbb{N} \cap [1, d-1] : \mathbb{P}[Z_{t+1} = s + 1 | Z_t = s] \leq p \)
3. \( \forall s \in \mathbb{N} \cap [0, d] : \mathbb{P}[Z_{t+1} - Z_t > 1 | Z_t = s] \leq q \)

Then, for \( Z_0 = 0 \) and \( T = \inf\{t \in \mathbb{N} : Z_t \geq d\} \), it holds that

\[
\mathbb{P}\left[T \geq \min\left\{ 1, \frac{d^2(1 - q)}{4\ln(10)p} \right\} \right] \geq \frac{16}{25}.
\]

Lemma 3.8 is proven in multiple steps. In the first step, the time until the first jump happens is bounded, which is responsible for the \( 1/25 \) term. Then, a new symmetric process is constructed that dominates the original process when conditioning on no jump to happen for a sufficiently large number of time steps. Lastly, the hitting time of this new process is bounded by combining Azuma’s inequality with a concentration result for sums of independent geometric random variables, which leads to the term \( d^2(1-q)/4\ln(10)p \) in the bound.

We start by introducing some additional notation and terminology that will come in handy for stating and proving the main result of the section. Let \( B \) be a path of odd length \( n \) with vertices \( v_1, \ldots, v_n \) and let \( X \in \{0, 1\}^n \) be a bit string \( x_1, \ldots, x_n \), representing a solution candidate for the vertex cover problem.

Similarly to Section 3.1.1, a major role is played in these proofs by connected subpaths of \( X \) in which all vertices are not in their optimal state, but they appear in a different form, and so we introduce some new notation. Let \( B(X) \subseteq [n]^2 \) denote the set of all tuples \((i, j)\) with \( i \leq j \) that corresponds to endpoints of subpath \( v_i, \ldots, v_j \subseteq V \) such that all vertices \( v_k \) for \( i \leq k \leq j \) are not in their optimal state. That is, each \( x_k \) for \( i \leq k \leq j \) has value zero if and only if \( k \) is even. We shall call such subpaths ‘bad paths’.

Then \( X \) represents a minimum vertex cover if and only if \( X \) is feasible and \( B(X) = \emptyset \). Assumption (A) is equivalent to assuming that for initial configuration \( X_0 \), \( B(X_0) \) contains exactly on tuple \((i, j)\) with \( j-i \in \Theta(n) \).

The main results of this section are the following two statements.
Theorem 3.9. Let $P = (V, E)$ be a path of odd length $n = |V|$. Assume a feasible initial configuration $X_0$ with $b(X_0) = \{(i, j)\}$ such that $j - i \in \Theta(n)$. With constant positive probability the (1+1) EA requires at least $\Omega(n^3)$ iterations to find the optimal solution. Consequently, the expected number of iterations is in $\Omega(n^3)$.

Theorem 3.10. Let $P = (V, E)$ be a path of odd length $n = |V|$. Assume a feasible initial configuration $X_0$ with $b(X_0) = \{(i, j)\}$ such that $j - i \in \Theta(n)$. With constant positive probability the Balanced (1+1) EA requires at least $\Omega(n^3)$ iterations to find the optimal solution. Consequently, the expected number of iterations is in $\Omega(n^3)$.

3.2.1 The (1+1) EA: Lower Bound Proof. As long as $b(X_t) \neq \emptyset$, let $l(X_t)$ and $r(X_t)$ denote the left and right endpoint of the bad path in $P$ with respect to $X_t$ (i.e., $B(X_t) = \{(l(X_t), r(X_t))\}$). Let $m = l(X_0) + (r(X_0) - l(X_0))/2$ and set $l(X_t) = r(X_t) = m$ for all $t \in \mathbb{N}$ with $b(X_t) = \emptyset$. Define $b(X_t) = r(X_t) - l(X_t)$ and observe that $T = \inf\{t \in \mathbb{N} \mid b(X_t) \leq 0\}$ is a lower bound on the required number of iterations to reach the optimum.

We proceed by showing that $T \in \Omega(n^3)$ with positive probability. To this end, define two new stopping times $T_1 = \inf\{t \in \mathbb{N} \mid l(X_{t+1}) > 2 \mid X_t\} \leq 2/n^4$ and $T_2 = \inf\{t \in \mathbb{N} \mid r(X_{t+1}) = r(X_t) - 2 \mid X_t\}$ whenever $l(X_t) \geq 3$ and analogously $P[|l(X_{t+1}) - l(X_t)| > 2 \mid X_t] \leq 2/n^4$ and $P[|r(X_{t+1}) - r(X_t)| = 2 \mid X_t]$ whenever $r(X_t) \leq n - 2$.

Using the characterization of the transition probabilities given in Lemma 3.11 we use Lemma 3.8 to prove Theorem 3.9.

Proof of Theorem 3.9. Let $l(X_t)$, $r(X_t)$, $b(X_t)$, $m$, $T$, $T_1$ and $T_2$ be defined as above. As discussed earlier, $T$ is a lower bound on the number of iterations that the (1+1) EA requires to get to the optimum. Furthermore, it holds that $T \geq \min\{T_1, T_2\}$. Thus, for every $t \in \mathbb{N}$, it holds that

$$P[T \geq t] \geq 1 - P[T_1 < t \text{ or } T_2 < t] \geq 1 - P[T_1 < t] - P[T_2 < t].$$

We proceed by using Lemma 3.8 to prove that, if $r(X_0) - l(X_0) \in \Theta(n)$, then there is some $\tau \in \Theta(n^3)$ such that $P[T \geq \tau] \geq \frac{1}{4n^2}$ and $P[T \geq \tau] \geq \frac{1}{2n^2}$.

To this end, consider the process $Z_t = \frac{l(X_{t+1}) - l(X_t)}{\log_{2}l(X_{t+1})}$ and set $d = \frac{l(X_{t+1}) - l(X_t)}{4}$. Note that $d \in \Theta(n)$, which implies $d \geq 4\ln(10)$ for $n$ sufficiently large. By Lemma 3.11, we know that $Z_t$ satisfies the requirements of Lemma 3.8 for $p = \frac{1}{n}$ and $q = \frac{2}{n}$. Thus, for $\tau = \min\{\frac{3}{n^4}, \frac{d/n^4}{8\ln(10)} (1 - \frac{1}{2})\} \in \Theta(n^3)$ and $T_{d} = \inf\{t \in \mathbb{N} \mid Z_t \geq d\}$, Lemma 3.8 yields $P[T_{d} \geq \tau] \geq \frac{10}{27n^2}$. Now, note that for our choice of $Z_t$ and $d$, $l(X_t) \geq m - 4$ implies $Z_t \geq d$. Therefore, we have $P[T \geq \tau] \geq \frac{1}{2n^2}$.

For $r(X_t)$, we can argue analogously that $P[T \geq \tau] \geq \frac{10}{27n^2}$ for the same $d \in \Theta(n)$ and $\tau \in \Theta(n^3)$ as above and using the process $Z_t = \frac{r(X_{t+1}) - r(X_t)}{2} \cdot I_{r(X_t) \leq r(X_{t+1})}$. Thus, we obtain $P[T \geq \tau] \geq \frac{7}{25n^2}$, which proves the first part of the statement. For the second part, note that by $T \geq 0$ and the law of total expectation $E[T] \geq \tau \geq \frac{7}{25} \in \Theta(n^3)$. By monotonicity of the expectation, this carries over to the expected number of iterations.

3.2.2 The Balanced (1+1) EA: Lower Bound Proof. The proof of Theorem 3.10 works analogously to that of Theorem 3.9. The only substantial difference is in the transition probabilities in Lemma 3.11. In particular, the Balanced (1+1) EA satisfies

$$P[l(X_{t+1}) = l(X_t) + 2 \mid X_t] = 1/2(1/n^2 + 1/n),$$

and

$$P[r(X_{t+1}) = r(X_t) - 2 \mid X_t] = 1/2(1/n^2 + 1/n).$$

Clearly $1/2(1/n^2 + 1/n) \leq 1/n$, which gives a lower bound of $\Omega(n^3)$ for the (1+1) EA.

3.2.3 Experimental Evidence for Long Bad Paths. So far we have argued that given a feasible initial configuration with exactly one bad path of length in $\Theta(n)$, the (1+1) EA requires with constant probability $\Omega(n^3)$ iterations to reach the optimum. Analogously, we have shown that the Balanced (1+1) EA requires $\Omega(n^3)$ steps.

We are now going to present experimental evidence that, starting from a uniformly chosen configuration, the (1+1) EA and the Balanced (1+1) EA both reach a state $X \in \{0, 1\}^n$ with $f(X) = 0$ with high probability. Once such a state is reached, our experiments suggest that the bad path of $X$ has linear length with at least constant probability.

We did experiments on paths of odd length $n \in \{51 + 10 \cdot k \mid k \in \mathbb{N} \cap [0, 15]\}$ with 100 runs for each $n$ and each algorithm. In every iteration, the algorithm starts with a uniformly random initial configuration. Once a state $X$ with $f(X) = 0$ is reached, we record the length of the bad path in $X$, divided by $n$. Since $X$ has only one vertex more than the optimal cover, there will be only one bad path, and all other vertices will have their optimal value.

In the case that the algorithm never reaches such a state (i.e. it jumps directly from fitness level $t > 1$ to the optimum), the length of the bad path is set to zero, which is a worst-case. Interestingly, this behaviour wasn’t observed in even a single simulation, which suggests that such a jump is difficult for either algorithm to perform.

The results are given in Fig. 1. Each box indicates span from the first to the third quartile (i.e. the inner 50%) of the data) and the whiskers indicate minimum and maximum values that were observed. The result for both algorithms look very alike. Apart from minor deviations, most of the iterations resulted in similar relative lengths of the bad paths, independent of $n$. The medians seem to fluctuate around a value of $1/3$ and never drop below $1/5$, which behavior is consistent throughout all tested values of $n$.

Our experiments suggest that, with constant probability, both algorithms enter the last fitness level with a bad path of linear length. Therefore, when starting from a uniformly random initial configuration, with constant probability a state is reached to which
A common instance class of interest when it comes to evolutionary algorithms for the vertex cover problem are complete bipartite graphs. In this section, we investigate basic properties of the Balanced (1+1) EA in this setting. Specifically, for \( L, R \in \mathbb{N} \setminus \{0\} \), let \( K_{L,R} = (V, E) \) be a graph, such that \( V \) can be partitioned into sets \( V_L \) and \( V_R \) with \( |V_L| = L \) and \( |V_R| = R \), and \( E = \{\{v, u\} | v \in V_L \text{ and } u \in V_R\} \). We call \( V_L \) the left partition and \( V_R \) the right partition. A bit string \( X \in \{0,1\}^{L+R} \) represents a feasible solution for the vertex cover problem on \( K_{L,R} \) if and only if \( V_L \subseteq X \) or \( V_R \subseteq X \).

Now assume \( R/L = c \) for some \( c > 1 \), and note that in this case \( X = V_L \) is the unique optimum of the vertex cover problem on \( K_{L,R} \). On the other hand, the state \( X = V_R \) represents a local optimum with respect to the fitness function \( f \). Once this local optimum is reached, \( X \) must flip at least \( 2L \) bits in one step to escape. This causes the (1+1) EA to need an exponential number of iterations in expectation, and local search heuristics like Random Local Search (RLS) to fail completely. The probability of reaching the optimal solution in polynomial time is usually closely related to the probability of never getting too close to that local optimum, which makes complete bipartite graphs an interesting instance class to study.

In this section, we specifically consider the following setting. We assume the ratio \( c := R/L > 0 \) to be fixed and investigate the probability for finding the optimum in polynomial time asymptotically in \( L \). Our main result is that the probability of not finding the optimum in polynomial time decays exponentially in \( L \) as long as \( c > 2 \). As Table 1 shows, this is quite different from the RLS, where the probability is constant for any fixed \( c \), and the normal (1+1) EA, where it only decays polynomially in \( L \). Note all asymptotic behavior here is stated in terms of \( L \). This can be easily translated to asymptotics in terms of the absolute number of vertices \( n = L + R = (c + 1)L \). In particular, as long as the ratio \( c \) is considered a fixed constant, this only changes linear factors, leading to the results that are presented in the abstract.

**4 VERTEX COVER ON COMPLETE BIPARTITE GRAPHS**

Theorems 3.9 and 3.10 apply. This suggests that, even when starting from a uniform random initialization, the expected number of iterations to find the optimum are in \( \Omega(n^4) \) for the (1+1) EA and in \( \Omega(n^3) \) for the Balanced (1+1) EA, due to the appearance of such linear bad paths.

The main technical result that we use to prove this bound on the number of iterations can informally be stated as follows. As long as \( V_R \) is more than twice as big as \( V_L \), the probability of selecting at least \( R - L \) vertices in \( V_R \) before selecting all vertices in \( V_L \) decays exponentially in the size of \( L \).

**Lemma 4.1**. Let \( K_{L,R} \) be a complete bipartite graph with vertex partitions \( V_L \) of size \( L \) and \( V_R \) of size \( R \), and set \( c := R/L \). Let \( X_t \in \{0,1\}^{L+R} \) be the sequence of states of the Balanced (1+1) EA with fitness function \( f \) on \( K_{L,R} \) and let \( X_0 \) be chosen uniformly at random. Define \( T_L := \inf\{t \in \mathbb{N} | |V_L \cap X_t| = L \} \) and let \( T_R = \inf\{t \in \mathbb{N} | |V_R \cap X_t| \geq R - L \} \). If \( c > 2 \), then, for all positive constants \( \xi < 1 \) and \( L \) sufficiently large, it holds that \( P\{T_L \geq T_R^\xi\} \leq 2^{-\Omega(L^5)} \).

The following probabilistic bound on the number of iterations for finding the minimum vertex cover on \( K_{L,R} \) can be derived from Lemma 4.1.

**Theorem 4.2**. Let \( K_{L,R} \) be a complete bipartite graph with vertex partitions \( V_L \) of size \( L \) and \( V_R \) of size \( R \), and set \( c := R/L \). If \( c > 2 \) then, for all positive constants \( \xi < 1 \), it holds that the Balanced (1+1) EA with fitness function \( f \), starting from a uniformly random initial configuration, requires at most \( O((c+1)L^2 \log((c+1)L)) \) iterations to find the minimum vertex cover of \( K_{L,R} \) with probability at least \( 1 - 2^{-\Omega(L^5)} \).

**Proof**. We prove this statement in two steps. At first, we argue that, with sufficiently high probability, we reach a state where all vertices in \( V_L \) are selected and more than \( L \) vertices from \( V_R \) are not selected in at most \( O((c+1)L^2 \log((c+1)L)) \) iterations. In the second step, we argue that, once such a state is reached, we go to

![Figure 1: Relative length of the bad path at last fitness level for (1+1) EA and Balanced (1+1) EA](image-url)
the optimum in at most $O((c + 1)L^2 \log((c + 1)L))$ iterations with high probability.

For the first part, let $X_t$ denote the states of the Balanced $(1+1)$ EA and let $T_L$ and $T_R'$ be defined as in Lemma 4.1. Formally, we want to lower bound $P[T_L \leq \tau]$ and $T_L < T_R'$ for some suitably chosen $\tau \in O((c + 1)L^2 \log((c + 1)L))$. Let $\tau = \inf \{t \in \mathbb{N} \mid X_t \text{ is feasible} \}$ and observe that, if $T_L \leq T_R'$, then $T = T_L$. Thus, we have

$$P[T_L \leq \tau \land T_L < T_R'] = P[T \leq \tau \lor T_L \geq T_R'] .$$

Using De Morgan’s laws and union bound, we get

$$P[T \leq \tau \land T_L < T_R'] = 1 - P[T > \tau] - P[T_L \geq T_R'] .$$

By Lemma 4.1, we have $P[T_L \geq T_R'] \leq 2^{-\Omega(L^2)}$. Setting $\tau = 4eL(L + R)(\ln(L + R) + 1/2)$, Lemma 2.1 yields $P[T \geq \tau] \leq 2^{-L}$. Thus, we get that the Balanced $(1+1)$ EA reaches a state with all vertices in $V_L$ selected and more than $L$ vertices in $V_R$ are not selected in at most $\tau \in O((c + 1)L^2 \log((c + 1)L))$ iterations.

Note, now assume we start in a state $X_0$ with $V_L \subseteq X_0$ and $|X_0 \cap V_R| < R - L$. Note that adding vertices from $V_R$ requires removing at least as many vertices from $V_L$. However, to remain feasible, removing any vertex from $V_R$, requires adding all vertices from $V_R$ that are not selected yet. As there are at least $L + 1$ such vertices and at most $L$ vertices in $V_L$ that could be removed, the Balanced $(1+1)$ EA can never add a vertex from $V_R$ or remove a vertex from $V_L$ in such a state. Thus, all we need to bound is the required time to remove all vertices in $X_0 \cap V_R$ to also bound the time for reaching the optimum. First, observe that for each fixed vertex in $X_0 \cap V_R$, the probability of not being removed after $\tau'$ steps is at most

$$e^{-\frac{\tau'}{e L^2}} = e^{-\frac{\tau'}{e L^2}} .$$

This, using union bound, the probability for not reaching the optimum after $\tau'$ steps is at most

$$|X_0 \cap V_R| e^{-\frac{\tau'}{e L^2}} < (R - L)e^{-\frac{\tau'}{e L^2}} = (c - 2)Le^{-\frac{\tau'}{e L^2}} .$$

Choosing $\tau' = (c+1)L \log((c - 2)L + eL) + eL \log((c + 1)L)$, this probability is upper bounded by $e^{-L}$. Observe that, if the first phase needs $\tau$ iterations and the second phase needs $\tau'$ iterations, then the total number of iterations for finding the optimum is $\tau + \tau' \in O((c + 1)L^2 \log((c + 1)L))$. The probability that at least one of both phases fails is at most $2^{-\Omega(L^2)} + e^{-L} \leq 2^{-\Omega(L^2)}$, which concludes the proof.

We proceed by proving Lemma 4.1. To simplify notation, we define the processes $L_t = V_L \cap X_t$ and $R_t = V_R \cap X_t$. A central step in our proof of Lemma 4.1 is to argue that $T_R'$ is super polynomial with high probability. Once this is done, it remains to show that, with sufficiently high probability, all vertices in $V_L$ are selected beforehand.

To accomplish the first part, we are going to apply a negative drift theorem to the slightly modified process $S_r \in \{0, 1\}^R$, which is defined based on $X_t$ as follows:

- Initially, $S_0 = R_0$.
- Whenever $X_t$ attempts to flip bits for some set of vertices $A \subseteq V, S_t$ flips all bits in $(V_R \cap A) \setminus S_t$ (in set notation, this is $S_{t+1} = S_t \cup (V_R \cap A)$).

Whenever $X_t$ attempts a balanced flip with initial vertex $v \in S_t$, the corresponding bit in $S_t$ is set to 0 (i.e., $S_{t+1} = S_t \setminus \{v\}$).

The following relationship between $R_t$ and $S_t$ will come in handy throughout our analysis.

**Lemma 4.3.** Let $T_L, T_R'$ and $S_t$ be defined as above. Then $R_t \leq S_t$ for all $t \leq \min(T_L, T_R')$.

We are aiming to use the stopping time $T_S = \inf \{t \in \mathbb{N} \mid |S_t| \geq R - L \}$ as a lower bound for $T_R'$. The following lemma justifies this.

**Lemma 4.4.** For $T_R'$ and $T_S$ as above it holds that $T_R' \geq T_S$.

Based on Lemma 4.4, the following statement is essentially derived by applying a negative drift argument to $|S_t|$.

**Lemma 4.5.** Let $T_R'$ be defined as above. If $c > 2$, then, for every positive constant $\xi < 1$ and all $L \geq (\max \{2, 4/(c - 2)\})^{1/\xi}$, there is a constant $\alpha > 0$ and a function $g(L) \in \Omega(L^2)$ such that

$$P[T_R' \leq g\alpha^L] \leq 2^{-g^L} ,$$

given the initial configuration $X_0$ is chosen uniformly at random.

Having Lemma 4.5 at hand, we are prepared to prove Lemma 4.1.

**Proof of Lemma 4.1.** By Lemma 4.5 we know that for $L$ sufficiently large and $c > 2$ there is a constant $\alpha > 0$ and a function $g(L) \in \Omega(L^2)$ such that $P[T_R' \leq g\alpha^L] \leq 2^{-g^L}$. We know that $P[T_R' < T_R' \geq \tau \land T_R' > \tau] \geq \tau$ holds for every $\tau \in \mathbb{N}$. Let $T = \inf \{t \in \mathbb{N} \mid X_t \text{ is feasible} \}$ and observe that, if $T > \tau$, then $T_L \leq \tau$ and only if $T \leq \tau$. Consequently, we have $P[T_L \leq \tau \land T_R' > \tau] = P[T \leq \tau \land T_R' > \tau]$. Moreover, using union bound, we get

$$P[T \leq \tau \land T_R' > \tau] = 1 - P[T > \tau \lor T_R' \leq \tau] = 1 - P[T > \tau] - P[T_R' \leq \tau] .$$

By choosing $\tau = 4e(c + 1)L^2 \log((c + 1)L) + 1/2$, Lemma 2.1 yields $P[T \geq \tau] \leq 2^{-L}$. Moreover, for $L$ sufficiently large, we have $\tau \leq 2\alpha^L$. Thus, $P[T_R' \leq \tau] \leq 2^{-\alpha^L}$ for a function $g(L) \in \Omega(L^2)$. Consequently, $P[T_L \leq T_R' \geq \tau] \geq 1 - 2^{-\Omega(L^2)}$ and $P[T_L \geq T_R' \geq \tau] \leq 2^{-\Omega(L^2)}$, which concludes the proof.

Besides Theorem 4.2, a variety of other properties that might be of independent interest can be derived from Lemma 4.1. For example, the following corollary shows that the probability of $X$ filling the larger side before all vertices in the smaller partition are selected, and the probability to ever select the entire larger side at all, both decay exponentially in $L$ as well.

**Corollary 4.6.** Consider the setting of Lemma 4.1 and let $T_R = \inf \{t \in \mathbb{N} \mid V_R \subseteq X_t\}$. If $c > 2$, then, for all positive constants $\xi < 1$ and $L$ sufficiently large, it holds that

1. $P[T_L \geq T_R] \leq 2^{-\Omega(L^2)}$
2. $P[T_R < \infty] \leq 2^{-\Omega(L^2)}$.

**Acknowledgments**

This research was partially funded by the HPI Research School on Data Science and Engineering.
REFERENCES

[1] Thomas Bäck and Sami Khuri. 1994. An evolutionary heuristic for the maximum independent set problem. Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence (1994), 331–335 vol.2.

[2] Samuel Baguley, Tobias Friedrich, Timo Kötzing, Xiaoyue Li, Marcus Pappik, and Ziena Zeil. 2022. Analysis of a Gray-Box Operator for Vertex Cover. arXiv preprint arXiv:2203.08790 (2022).

[3] Tobias Friedrich, Jun He, Nils Hebbinghaus, Frank Neumann, and Carsten Witt. 2010. Approximating covering problems by randomized search heuristics using multi-objective models. Evolutionary Computation 18, 4 (2010), 617–633.

[4] Tobias Friedrich, Timo Kötzing, Martin S. Krejca, Samadhi Nallaperuma, Frank Neumann, and Martin Schirneck. 2016. Fast Building Block Assembly by Majority Vote Crossover. In Proceedings of the Genetic and Evolutionary Computation Conference 2016 (GECCO ’16). Association for Computing Machinery, 661–668.

[5] Oliver Giel and Ingo Wegener. 2003. Evolutionary Algorithms and the Maximum Matching Problem. In STACS 2003 Springer Berlin Heidelberg, 415–426.

[6] Andreas Göbel, Timo Kötzing, and Martin S. Krejca. 2018. Intuitive Analyses via Drift Theory. CoRR abs/1806.01919 (2018). arXiv:1806.01919

[7] Thomas Jansen, Pietro S Oliveto, and Christine Zarges. 2013. Approximating vertex cover using edge-based representations. In Proceedings of the twelfth workshop on Foundations of genetic algorithms XII. 87–96.

[8] T. Jansen and Dirk Sudholt. 2010. Analysis of an Asymmetric Mutation Operator. Evolutionary Computation 18 (2010), 1–26.

[9] Richard M. Karp. 1972. Reducibility Among Combinatorial Problems. Complexity of Computer Computations (1972), 85–103.

[10] Stefan Kratsch and Frank Neumann. 2013. Fixed-parameter evolutionary algorithms and the vertex cover problem. Algorithmica 65, 4 (2013), 754–771.

[11] Per Kristian Lehre and Carsten Witt. 2012. Black-Box Search by Unbiased Variaton. Algorithmica 64, 4 (2012), 623–642.

[12] Frank Neumann and Ingo Wegener. 2004. Randomized Local Search, Evolutionary Algorithms, and the Minimum Spanning Tree Problem. In Genetic and Evolutionary Computation – GECCO 2004, Kalyanmoy Deb (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 713–724.

[13] Pietro S Oliveto, Jun He, and Xin Yao. 2007. Evolutionary algorithms and the vertex cover problem. In 2007 IEEE Congress on Evolutionary Computation. IEEE, 1870–1877.

[14] Pietro S Oliveto, Jun He, and Xin Yao. 2008. Analysis of population-based evolutionary algorithms for the vertex cover problem. In 2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence). IEEE, 1563–1570.

[15] Pietro S Oliveto, Jun He, and Xin Yao. 2009. Analysis of the (1+1)-EA for finding approximate solutions to vertex cover problems. IEEE Transactions on Evolutionary Computation 13, 5 (2009), 1006–1029.

[16] Jannik Peters, Daniel Stephan, Isabel Amon, Hans Gazendawicz, Julius Lischesid, Lennart Salabaria, Jonas Umland, Felix Werner, Martin S. Krejca, Ralf Rothenberger, Timo Kötzing, and Tobias Friedrich. 2019. Mixed Integer Programming versus Evolutionary Computation for Optimizing a Hard Real-World Staff Assignment Problem. In Proceedings of the Twenty-Ninth International Conference on Automated Planning and Scheduling, ICAPS 2018, Berkeley, CA, USA, July 11-15, 2019. J. Benton, Nir Lipovetzky, Eva Onaïndia, David E. Smith, and Siddharth Srivastava (Eds.). AAAI Press, 541–554.

[17] Jonathan E. Rowe and Michael D. Vose. 2011. Unbiased black box search algorithms. In 13th Annual Genetic and Evolutionary Computation Conference, GECCO 2011, Proceedings, Dublin, Ireland, July 12-16, 2011, Natalio Krasnogor and Pier Luca Lanzi (Eds.). ACM, 2035–2042.

[18] Ruhul Sarker, Masoud Mohammadian, and Xin Yao. 2002. Evolutionary Optimization. Kluwer Academic Publishers.

[19] L. Darrell Whitley, Francisco Chicano, and Brian W. Goldman. 2016. Gray Box Optimization for NK Landscapes (NK Landscapes and MAX-kSAT). Evol. Comput. 24, 3 (2016), 491–519.