Analyses of Simple Hybrid Algorithms for the Vertex Cover Problem *

Tobias Friedrich tobias.friedrich@mpi-inf.mpg.de
Max-Planck-Institut für Informatik, Saarbrücken, Germany

Jun He jqh@aber.ac.uk
Dept. of Computer Science, University of Wales, Aberystwyth, United Kingkom

Nils Hebbinghaus nils.hebbinghaus@mpi-inf.mpg.de
Max-Planck-Institut für Informatik, Saarbrücken, Germany

Frank Neumann frank.neumann@mpi-inf.mpg.de
Max-Planck-Institut für Informatik, Saarbrücken, Germany

Carsten Witt carsten.witt@cs.uni-dortmund.de
Fakultät für Informatik, LS 2, Technische Universität Dortmund, Dortmund, Germany

Abstract

Hybrid methods are very popular for solving problems from combinatorial optimization. In contrast to this the theoretical understanding of the interplay of different optimization methods is rare. In this paper, we make a first step into the rigorous analysis of such combinations for combinatorial optimization problems. The subject of our analyses is the vertex cover problem for which several approximation algorithms have been proposed. We point out specific instances where solutions can (or cannot) be improved by the search process of a simple evolutionary algorithm in expected polynomial time.

Keywords

hybrid algorithms, evolutionary computation, combinatorial optimization, running time analysis

1 Introduction

Evolutionary Algorithms (EAs) have been widely applied to optimization problems [8, 10, 14]. Especially for combinatorial optimization they have been shown to produce good solutions. To be competitive with other methods such as approximation algorithms [24], Tabu Search [7], Branch and Bound [1], or Linear Programming [21] often not a "pure" EA is used but some knowledge is incorporated into the algorithm.

In contrast to the assumption that using hybridization in evolutionary computation often yields better results than a more general approach with problem-specific

*A preliminary version of this article appeard in *Proceedings of the 14th IEEE Congress on Evolutionary Computation (CEC 2007)*, pages 2614–2621, IEEE Press, 2007.

knowledge, there are up to now no theoretical investigations with respect to the runtime behavior of such algorithms. A lot of theoretical results giving bounds on the runtime of simple evolutionary algorithms for combinatorial optimization problems have been obtained in recent years. Among these problems, there are some of the most popular polynomially solvable problems such as sorting and shortest path [20], the computation of a maximum matching [5, 6], the Eulerian cycle problem [2, 15], and the computation of minimum spanning trees [17, 18]. In the case of NP-hard problems, results have been obtained for a scheduling problem on two machines [26] and the multi-objective minimum spanning tree problem [16]. Especially, in the case of well-known NP-hard combinatorial optimization problems often the initial solution is computed by an approximation algorithm and later on improved by a heuristic method. None of previous works on the runtime analysis of combinatorial optimization problems has considered this scenario. All investigations considered the case where the initial solution is drawn uniformly at random from the considered search space. Our aim is to make a first step to understand in which situations evolutionary algorithms are able to improve solutions obtained by specialized approximation algorithms. We do not aim to show general results on improving the approximation ratio of such a problem as this would be a statement which has to be made about all possible instances. We instead focus on describing different situations where the random search process of an evolutionary might (or might not) be useful.

The subject of our investigations is the vertex cover problem. Various EAs have been applied to this problem [3, 13]. Kehden and Neumann [12] showed in an experimental study that in the case of random graphs, solutions obtained by a well-known approximation algorithm are often far from optimal and may be improved in a small number of iterations by evolutionary algorithms so that nearly optimal solutions are obtained. The first rigorous runtime analysis on classes of instances for this problem is given in [9]. Recently, it has been shown that the well-known (1+1) EA is not able to produce a good approximation for the vertex cover problem [4]. In this work, it has been proven that even in the case of bipartite graphs the approximation ratio achievable by this algorithm in expected polynomial time is almost as bad as the trivial cover made up of all vertices of the given graph.

Such a bad approximation quality can be prevented by starting with an initial solution that has been obtained by running a good approximation algorithm. In the case of the vertex cover problem several approximation algorithms are known. The first idea for such an algorithm is to start with the empty vertex set and add in each iteration a vertex that covers the largest number of uncovered edges. It is well-known that this approach achieves an approximation ratio of $\Theta(\log n)$, where n is the number of vertices in the graph. Some simple approximation algorithms obtain an approximation ratio of 2 which is asymptotically the best known up to now. Such an approximation quality can e. g. be obtained by computing a maximal matching of the given graph and including for each edge of this matching both endpoints into the cover.

We investigate for which cases the solutions obtained by the two described approximation algorithms can be improved by an evolutionary algorithm. We investigate the (1+1) EA starting with such a solution and point out situations where such a search process has (or has not) the ability to improve a solution constructed by the approximation algorithms.

The outline of the paper is as follows. In Section 2 we introduce the vertex cover

problem and the EA that is subject to our investigations. Section 3 considers improvements achievable by the EA for the greedy approximation algorithm and Section 4 investigates the combination with the maximal matching approach. Finally, we finish with some conclusions.

2 The Vertex Cover Problem and the (1+1) EA

The vertex cover problem is one of the most studied NP-hard combinatorial optimization problems. Given an undirected graph $G = (V, E)$ where $|V| = n$ and $|E| = m$, the aim is to find a subset $V' \subseteq V$ of minimum cardinality such that for each $e \in E$, $e \cap V' \neq \emptyset$ holds.

We consider the well-known (1+1) EA (see Algorithm 1) for the vertex cover problem. The search space is $\{0,1\}^n$ and each bit x_i of a solution x corresponds to a vertex $v_i \in V$. The vertex v_i is chosen in the current solution x if $x_i = 1$ and otherwise it is unchosen.

Algorithm 1. *(1+1) EA*

1. *Choose an initial solution $x \in \{0,1\}^n$.*
2. *Repeat*
   - *Create x' by flipping each bit of x with probability $1/n$.*
   - *If $f(x') \leq f(x)$, set $x := x'$.*

Denote by $|x|_1$ and $|x|_0$ the number of ones respectively of zeros in a bitstring x. The fitness of a search point x is given by $f(x) = (u(x), |x|_1)$ where $u(x)$ denotes the number of uncovered edges of the solution x. In the case of the (1+1) EA, the function should be minimized with respect to the lexicographic order. This setting has already been examined in [4] for randomly chosen initial solutions. We examine the effect of using an initial solution that has been computed by some approximation algorithm. Here in any case all edges are covered and the (1+1) EA does not accept solutions that do not constitute a vertex cover.

Our aim is to analyze the (1+1) EA by a rigorous run-time analysis until it has produced good solutions for the vertex cover problem. The measure of interest is the number of constructed solutions until certain goals have been achieved. Often the expectation of this value is considered and called the expected time to achieve such a goal.

Most of our investigations consider the approximation ability of the proposed algorithms. The worst-case approximation ratio of an algorithm A for a given minimization problem R is defined as $\max_{I \in R} \frac{A(I)}{\mathrm{OPT}(I)}$ where $A(I)$ denotes the value obtained by A when applied to an instance I of R and $\mathrm{OPT}(I)$ denotes the value of an optimal solution for the given instance. We are mainly interested in upper and lower bounds for the number of constructed solutions until a certain approximation ratio has been achieved by the introduced algorithms.

As the (1+1) EA does not accept worsenings the approximation ratio achieved is at least as good as the approximation ratio of the algorithm to compute the initial solution.

3 Analysis of Hybrid EAs with the Greedy Method

In this section we examine how an initial solution produced by a greedy algorithm can be improved by the (1+1) EA. We show that the success of such a Hybrid EA depends on the specifics of the examined graphs. We introduce two graph classes and show that the Hybrid EA may fail on the first one to find a solution with approximation ratio $o(\log n)$ in polynomial time and finds the optimum quickly for the second class of graphs.

The greedy method is based on the following idea. In the vertex covering problem, the aim is to find a vertex cover which uses the minimum number of vertices to cover all edges. Therefore, a vertex with a larger degree is more likely to appear in the optimal vertex cover. However, this is only a heuristic knowledge and it will not lead to an optimal solution in general.

We consider the greedy method described in [19] to compute the initial solution x. In each step of the algorithm a vertex with the largest degree is added to the solution and the vertex and the edges incident to it are removed from the graph. The process is iterated until the graph becomes empty. We can state the algorithm as follows.

Algorithm 2. *Greedy Vertex Cover*

1. *Set* $x = 0^n$
2. *Repeat*
   - *Choose a vertex* v_k *having the largest degree in* G.
   - *Set* $x_k = 1$, $V := V \setminus \{v_k\}$, *and* $E := E \setminus \{e \mid e \cap v_k \neq \emptyset\}$
3. *Until* G *is empty*

The greedy approach achieves an approximation ratio of $O(\log n)$. We obtain a simple hybrid algorithm by computing the initial solution of the (1+1) EA using Algorithm 2. The resulting algorithm we will denote by *Greedy Vertex Cover (1+1) EA* ((1+1) EA$_G$). It achieves the same approximation ratio $O(\log n)$ as the greedy approach as it does not accept solutions that are worse than the initial one.

Friedrich et al. [4] have shown that the (1+1) EA starting with a solution that is chosen uniformly at random from the search space is not able to obtain a good approximation for a specific class of bipartite graphs in expected polynomial time. They have also shown that a greedy approach in form of multi-objective EA is able to produce an optimal solution for such problems quickly. The same holds for Algorithm 2. Hence, using the greedy procedure for the initial solution can make the difference between obtaining an optimal solution or achieving only a bad approximation of such a solution.

Compared with other approximation algorithms that achieve an approximation ratio of 2 for the vertex cover problem, the greedy approach behaves badly in the worst case. Therefore, the question arises whether a solution that is far from optimal can always be improved by the (1+1) EA. The following example shows that this is not always the case. The solution for Graph 1 computed by the greedy approach might be far from optimal and the (1+1) EA$_G$ is likely to achieve not even a single improvement.

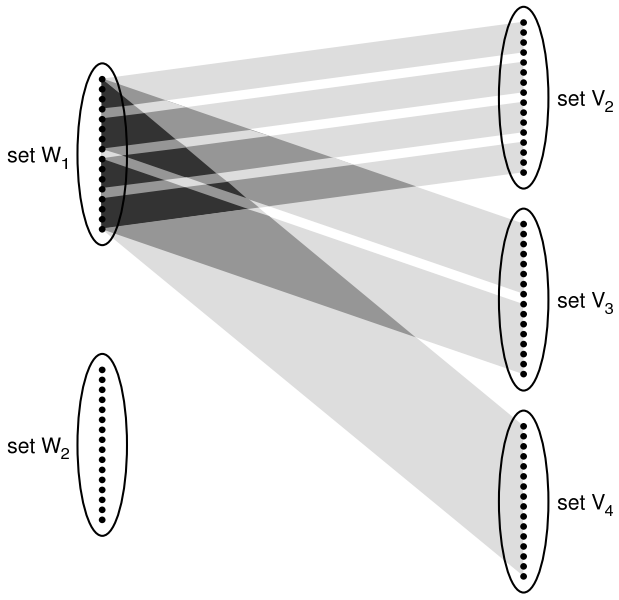

Figure 1: An illustration of Graph 1 for $k = 4$. The grey straps should indicate complete bipartite subgraphs. For better clarity, the edges incident to W_2 are omitted.

Graph 1. *Let the vertex set be* $W_1 \cup W_2 \cup \bigcup_{i=\lceil k/2 \rceil}^{k} V_i$ *with*

$$W_\ell := \{w_{\ell,j} \mid j = 1, \ldots, 2^k\} \text{ for all } \ell \in \{1, 2\}$$
$$V_i := \{v_{i,j} \mid j = 1, \ldots, 2^k\} \text{ for all } \lceil k/2 \rceil \leq i \leq k$$

and the edge set be

$$E := \{\{v_{i,j_1}, w_{\ell,j_2}\} \mid \lceil k/2 \rceil \leq i \leq k,\ \ell \in \{1, 2\},\ 1 \leq j_1, j_2 \leq 2^k,\ \lceil j_1/2^i \rceil = \lceil j_2/2^i \rceil\}.$$

The number of vertices is $n = (\lfloor k/2 \rfloor + 3)\, 2^k$ and the vertices in set W_1 and W_2 have degree $\sum_{i=\lceil k/2 \rceil}^{k} 2^i = 2^{k+1} - 2^{\lceil k/2 \rceil}$ while the vertices in V_i have degree 2^{i+1}. The optimal cover is

$$C^* := W_1 \cup W_2.$$

In the following, we show that Algorithm 2 produces a factor $\Omega(\log n)$ approximation of an optimal solution and that such a solution is hard to improve by the search procedure of the (1+1) EA.

Theorem 1. *On Graph 1 the expected time for the (1+1) EA$_G$ to obtain a solution whose approximation ratio is $o(\log n)$ is $2^{\Omega(\sqrt{n \log n})}$.*

Proof. We first show that the solution produced by Algorithm 2 is only a factor $\Omega(\log n)$ approximation and lower bound the expected time to improve such a solution afterwards.

As the vertices in sets V_k have the largest degree, the greedy Algorithm 2 first choses all vertices from V_k. After removing these nodes, the vertex degrees in W_1 and W_2 decrease to $\sum_{i=\lceil k/2 \rceil}^{k-1} 2^i = 2^k - 2^{\lceil k/2 \rceil}$ while the vertices in V_i still have degree 2^i for $i < k$. Hence, the greedy method chooses the vertices of V_{k-1} to be added next to the cover. Iterating this procedure, the greedy method obtains the cover

$$C = V_k \cup V_{k-1} \cup \ldots \cup V_{\lceil k/2 \rceil}$$

and the approximation ratio

$$\frac{|\, C \,|}{|\, C^* \,|} = \frac{(\lfloor k/2 \rfloor + 1)\, 2^k}{2\, 2^k} = \Theta(k) = \Theta(\log n).$$

Hence, Graph 1 is, up to a constant factor, a worst case example for Algorithm 2 with respect to approximability.

To gain an improvement if C is the current solution, a certain number of vertices of $W_1 \cup W_2$ have to be added while a larger number of vertices of $\bigcup_i V_i$ have to be removed. As the minimum vertex degree in Graph 1 is $\delta = 2^{\lceil k/2 \rceil + 1}$, at least 2δ arbitrary bits have to be flipped to gain an improvement. It remains to bound the probability that the (1+1) EA_G achieves such an improvement in one mutation step. For this, we use Stirling's formula, the identity $n = 2^{\log n}$, and (for sufficiently large n) $\delta \geq \sqrt{\frac{n}{\log n}}$ and $\left(\frac{2}{e}\sqrt{\frac{n}{\log n}}\right)^3 \geq n$. Hence, the probability of an improvement can be upper bounded by

$$\frac{\binom{n}{2\delta}}{n^{2\delta}} \leq \frac{1}{(2\delta)!} \leq \left(\frac{2\delta}{e}\right)^{-2\delta} \leq n^{-\frac{2}{3}\sqrt{\frac{n}{\log n}}} = 2^{-\Omega(\sqrt{n \log n})}.$$

which proves the theorem. □

Theorem 1 shows that the (1+1) EA_G may be unable to improve a bad greedy solution. We will now show that there are also graph classes on which above hybrid EA can play a positive role. On the following (bipartite) Graph 2, the (1+1) EA finds the optimal vertex cover in expected polynomial time.

Consider the following well-known graph given in [19].

Graph 2. *Let $k \in \mathbb{N}$. Let the vertex set be $B \cup \bigcup_{i=1}^{k} A_i$ with*

$$B := \{b_j \mid j = 1, \ldots, k\},$$
$$A_i := \{a_{i,j} \mid j = 1, \ldots, \lfloor k/i \rfloor\} \text{ for all } 1 \leq i \leq k$$

and the edge set be

$$E := \{\{b_j, a_{i,\lceil j/i \rceil}\} \mid 1 \leq i \leq k,\ 1 \leq j \leq i\lfloor k/i \rfloor\}.$$

Graph 2 has the following properties. Since every vertex in A_i is connected with exactly i vertices in B and no two vertices in B have a common neighbor in any A_i, all vertices in B have degree at most k and all vertices in A_i have degree i. Let us denote the total number of vertices by n. Then $n = \Theta(k \log k)$. Due to [19], the greedy algorithm (Algorithm 2) can determine the set $\bigcup_{i=1}^{k} A_i$ as a vertex cover. This set is a

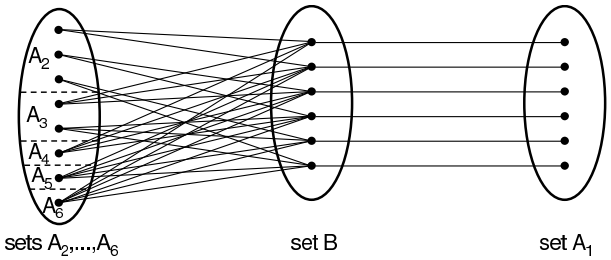

Figure 2: An illustration of Graph 2 for $k = 6$.

factor of $\Omega(\log n)$ away from the optimal vertex cover B. In contrast to the behavior of the (1+1) EA$_G$ on Graph 1, the (1+1) EA$_G$ determines the optimal vertex cover in expected time $O(n^3)$. We formulate this result in a more general way in the following theorem.

Theorem 2. *The expected optimization time of the (1+1) EA on Graph 2 is $O(n^3)$ for each initial search point.*

Proof. In [4] it is shown that the (1+1) EA produces a vertex cover in expected time $O(n \log n)$ regardless of the chosen initial solution. Therefore, we now assume that a vertex cover has been obtained and upper bound the expected time to produce an optimal solution which consists of the vertex set B. If the current cover is not a minimal one at least one vertex can be removed to achieve an improvement. The corresponding probability for such an event is $\Omega(1/n)$ and its expected waiting time is $O(n)$.

If the current solution is minimal but not a minimum vertex cover at least one vertex v_1 in A_2 has been chosen. Otherwise, all vertices of B must be selected (to cover all edges between A_2 and B) and no other vertex can be chosen due to the minimality of the vertex cover. But this is the mimimum vertex cover. The chosen vertex v_1 of A_2 has two neighbors v_2 and v_3 in B. In addition v_2 is adjacent to a vertex v_4 in A_1 and v_3 is adjacent to a vertex v_5 in A_1. To reduce the number of vertices in the case that the current solution is minimal but not a minimum vertex cover, we consider special 2-bit flips concerning the vertices of $\{v_2, v_3, v_4, v_5\}$. As the current solution is a cover which is minimal exactly two vertices of $\{v_2, v_3, v_4, v_5\}$ have been chosen and at least one of the chosen vertices belongs to A_1. Deleting a chosen vertex of A_1 and including its neighbor of B is always accepted and can be done by a specific 2-bit flip. Similarly, deleting a vertex of B and including its neighbor of A_1 can be done by a specific 2-bit flip. However, the acceptance of such a step is determined by the choice of the other vertices of $V \setminus (B \cup A_1)$, e. g., removing the vertex of B is not possible if at least one of its neighbors in $V \setminus (B \cup A_1)$ is unchosen. This shows that the probability of switching from a chosen vertex of A_1 to its neighbors in B is at least as high as the probability of switching from a chosen vertex of B to its neighbor of A_1.

Only steps affecting the vertices in $\{v_2, v_3, v_4, v_5\}$ are relevant for us. If not both vertices of B have been chosen, the probability of increasing the number of vertices in B in such a step is at least $1/2$ as at least one of the two vertices of B is missing. Two of such steps consecutively are sufficient to have both vertices of B chosen. Each such step chooses a missing B-vertex in favor of an A_1-vertex independently of earlier steps. Therefore, the probability that two consecutive such steps both increase the number of

B-vertices is at least $1/4$. The probability of carrying out a specific 2-bit flip mutation step is at least $1/(en^2)$. Hence, after an expected number of $O(n^2)$ steps both vertices of B are chosen. Now the solution is not minimal as the vertex v_1 can be removed. The probability that the vertex v_1 is removed before v_4 respectively v_5 are replaced by v_2 respectively v_3 is $1 - O(\frac{1}{n})$.

Thus, the expected waiting time for such an improvement is $O(n^2)$ in any case and the number of vertices is $O(n)$. This implies that a minimum vertex cover is obtained after an expected number of $O(n^3)$ steps. □

4 Analysis of Hybrid EAs with 2-Approximation Heuristics

It has been known for a long time (see e. g., [19]) that the vertex cover problem admits a 2-approximation using so-called *maximal matchings*. A matching is a subset of pairwise disjoint edges of a given undirected graph. Thus, the empty set is always a matching. A matching is called maximal if there is no edge left that can be added to the subset without violating the matching property. Therefore the maximal matching problem can be solved in linear time w. r. t. the number of edges by greedily adding edges to the current matching until the matching is maximal.

Note that a maximal matching is not necessarily a matching of maximum cardinality. From an algorithmic point of view, the latter problem—called the *maximum* matching problem—is more complicated yet can also be solved in polynomial time [19]. It is interesting that almost-maximum matchings can be found in polynomial time using pure evolutionary algorithms [5].

The proposed 2-approximation algorithm works as follows:

Algorithm 3. *Maximal Matching Vertex Cover*

1. *Compute a maximal matching.*
2. *Output all endpoints of the edges of the matching.*

The set of vertices we obtain is a vertex cover since otherwise, the matching would not have been maximal. Moreover, the set forms a 2-approximation since at least one endpoint of each edge in the matching must be chosen for a minimum vertex cover. There are examples, e. g., the bipartite graph consisting of $n/2$ vertex-disjoint edges, where the approximation ratio is tight since the matching chooses twice as many vertices as necessary. It is therefore interesting to study how such solutions can be improved by other heuristics.

We obtain a simple hybrid algorithm by computing the initial solution of the (1+1) EA using Algorithm 3. This is called the *Maximal Matching (1+1) EA* ((1+1) EA_M). Note that in fact, a family of algorithms is considered since the component computing the maximal matching has not been specified yet.

In the following, we study the interplay of the maximal matching component and the (1+1) EA in our hybrid algorithm. It is easy to see that for the above-mentioned graph on $n/2$ vertex-disjoint edges, the (1+1) EA_M finds a minimum vertex cover in expected polynomial time. Due to the NP-hardness of the problem, we however cannot expect the (1+1) EA_M to always find optimal solutions in expected polynomial

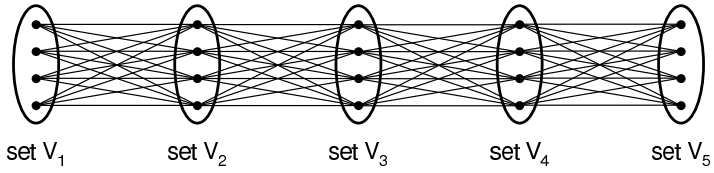

Figure 3: An illustration of Graph 3 for $k = 4$.

time. In fact, we do believe that there are instances where it fails to find in polynomial time solutions that are significantly better than 2-approximate. The best known polynomial-time approximation algorithm for vertex cover has an approximation ratio of $2 - \Theta(1/\sqrt{\log n})$ [11], so we cannot hope to show better results for our simple hybrid algorithm. However, a simple observation proves that the evolutionary component helps to improve solutions that are exactly 2-approximate.

Proposition 1. *With probability $\Omega(1)$, the (1+1) EA_M produces in expected polynomial time a vertex cover with an approximation ratio of $2 - 2/n$.*

Proof. Suppose that the optimal vertex cover has size k. If the solution of the maximal matching component is not $(2-2/n)$-approximate, $k \leq n/2$ must hold and the solution must be of cardinality greater than $2k - 2k/n$, hence of cardinality $2k$. This means that an optimal cover consists of exactly one endpoint of each edge in the matching. With probability $\Omega(1)$, the first step that flips one of the $2k$ bits of the solution flips only one of the bits that do not belong to the optimal vertex cover. The new solution of $2k - 1$ vertices is a $(2 - 2/n)$-approximation. □

The following considerations show that the choice of the maximal matching can be crucial for the performance of the hybrid algorithm and that the random search of the (1+1) EA can still result in very different solutions for a given initial matching. We describe two examples where specific matchings can lead the (1+1) EA into local optima of bad quality. The first one is composed of bipartite subgraphs on vertex sets of size $k = n/5$. These bipartite subgraphs are connected in a chain-like manner such that the whole graph still is bipartite. Figure 3 depicts the graph for $n = 20$.

Graph 3. *Let the vertex set be the union of*

$$V_i = \{v_{i,1}, \ldots, v_{i,k}\}$$

for $1 \leq i \leq 5$ and the edge set be the union of the sets

$$\big\{\{v_{i,r}, v_{i+1,s}\} \mid 1 \leq r, s \leq k\big\}$$

for $1 \leq i \leq 4$.

An optimal vertex cover for Graph 3 is obtained by choosing the sets V_2 and V_4. A suboptimal vertex cover of approximation ratio $3/2$ is given by $V_1 \cup V_3 \cup V_4$. Both vertex covers are likely to be reached if the initial solution of the (1+1) EA is $V_1 \cup V_2 \cup V_3 \cup V_4$. This initial solution is created by, e. g., the maximal matching

$$M^* := \bigcup_{r=1}^{k} \{v_{1,r}, v_{2,r}\} \cup \{v_{3,r}, v_{4,r}\}. \tag{1}$$

Theorem 3. *Suppose the maximal matching component of the (1+1) EA$_M$ creates the cover $V_1 \cup V_2 \cup V_3 \cup V_4$ for Graph 3. Then with probability $\Omega(1)$ each,*

- *the (1+1) EA creates the globally optimal solution $V_2 \cup V_4$ in polynomial time,*
- *stays at the locally optimal solution $V_1 \cup V_3 \cup V_4$ for a superpolynomial number of steps.*

Proof. We first observe that no vertex from V_4 can be removed from the cover unless a step adds all missing vertices from V_5 simultaneously to the cover. If the fitness function did not count the number of chosen vertices from V_5, this number would follow a random walk on the $2^{n/5}$ possible assignments to the bits corresponding to V_5, and the time to reach the all-one assignment would be $2^{\Omega(n)}$ with probability at least $1 - 2^{-\Omega(n)}$ using the arguments on needle-in-a-haystack functions in [25]. Since the fitness function is linear in the number of chosen vertices in V_5 and has to be minimized, the actual time to reach the all-one assignment is even larger. We therefore assume that no vertex from V_4 is removed from the cover in any phase of length 2^{cn} for small enough constant c, and hence do not have to take into account the sets V_4 in V_5 in the sequel. As a mutation step of the (1+1) EA$_M$ with probability at least $(1 - 1/n)^{2n/5} = \Omega(1)$ does not flip a bit corresponding to V_4 or V_5 and since these bits are treated independently of the bits corresponding to $V_1 \cup V_2 \cup V_3$, the assumptions will not affect our following asymptotic statements.

We are now faced with a situation like in the study of the (1+1) EA on the bipartite graph B in the paper by Friedrich et al. [4, Theorem 5]. The set $V' := V_1 \cup V_3$ of Graph 3 plays the role of V_2 of graph B and $V'' := V_2$ plays the role of V_1. With probability $\Omega(1)$, a vertex from V' of Graph 3 is removed before a vertex from V'' is removed. Then we apply the argumentation concerning the second phase in the proof of Theorem 5 in [4], which will show us that all vertices from V' are removed in polynomial time with constant probability. This will imply that the global optimum is reached with constant probability in polynomial time since the expected time to remove possible nodes from V_5 is also polynomial. More precisely, we consider a phase of $n^{3/2}$ mutation steps and show that all vertices of V' are removed with probability $\Omega(1)$. In this phase (all vertices of V'' and some vertices of V' chosen), the only mutation steps accepted by the (1+1) EA$_M$ are the following. Either all missing vertices of V' are chosen and at least as many vertices of V'' are removed, or all vertices of V'' are kept and the number of vertices in V' is decreased (or stays the same by adding and removing some vertices). The former mutation step has a probability of at most n^{-k}, where k denotes the current number of missing vertices in V'. For the latter kind of mutation steps we restrict ourselves to 1–bit flips reducing the number of vertices in V'. The probability for such a mutation step is at least $(1/n)(1 - 1/n)^{n-1} \geq 1/(en)$. For our calculations we take only those two kind of mutation steps into account, the "good event" with probability at least $1/(en)$ and the "bad event" with probability at most n^{-k} since all other accepted mutation steps reduce or preserve the number of vertices in V'. The probability that the "good event" occurs before the "bad event" is at least $\frac{1}{en}/(\frac{1}{en} + n^{-k}) = 1 - \frac{e}{n^{k-1}+e}$. Thus, the probability that the vertices of V' were all removed by the (1+1) EA$_M$ before the "bad event" occurs is at least

$$\prod_{k=1}^{2n/5} \left(1 - \frac{e}{n^{k-1} + e}\right) \geq \frac{1}{1+e} \left(1 - \frac{e}{n}\right)^{n-e} = \Omega(1).$$

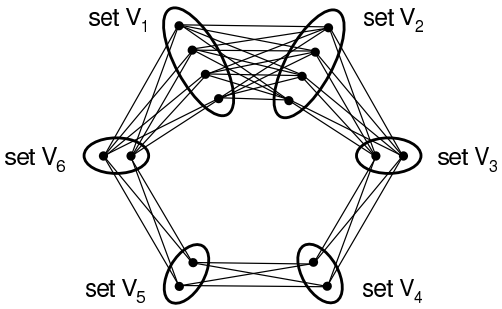

Figure 4: An illustration of Graph 4 for $k = 4$.

The expected waiting time for removing all vertices of V' by the (1+1) EA$_M$ is $O(n \log n)$ and, therefore, all vertices of V' are removed within $n^{3/2}$ steps with probability $1 - o(1)$ using Markov's inequality (always assuming that the "bad event" does not occur during this phase). Hence, the probability that the (1+1) EA$_M$ determines the global optimum as vertex cover in polynomial time is at least $\Omega(1)$.

The second claim follows by the same arguments. With probability $\Omega(1)$, a vertex from V_2 is removed first and the local optimum is reached with constant probability in polynomial time. To leave the local optimum, a step that flips k bits simultaneously has to occur. This has probability $n^{-\Omega(n)}$ and with probability $1 - n^{-\Omega(n)}$, does not happen in a superpolynomially long phase of 2^{cn} steps for a small enough constant $c > 0$. □

It seems tempting to decrease the size of V_2 in Graph 3 to trick the (1+1) EA into local optima with an even worse approximation ratio than $3/2$. This, however, does not work since the maximal matching component would choose only a subset of V_1 and V_3 then.

Using a less obvious, still bipartite graph, we can show that the (1+1) EA is very likely to get stuck at $(2 - o(1))$-optimal solutions if the initial maximal matching is chosen badly. The idea is to start from a big complete bipartite graph on vertex sets of size $k = \Omega(n)$ and to add small extra components of only $\log k$ vertices in a circular manner. Depending on the choice of the vertices from the small components, it can be necessary to add all vertices from the big bipartite graph to obtain a vertex cover. An optimal vertex cover, however, chooses only one side of the big bipartite graph.

Graph 4. *Let $n = 2k + 4 \log k$. Let the vertex set be the union of*

$$V_i = \{v_{i,1}, \ldots, v_{i,k}\}$$

for $1 \leq i \leq 2$ and

$$V_i = \{v_{i,1}, \ldots, v_{i,\log k}\}$$

for $3 \leq i \leq 6$. Define the edge set as the union of the sets

$$\big\{\{v, w\} \mid v \in V_i, w \in V_{(i \bmod 6)+1}\big\}$$

for $1 \leq i \leq 6$.

Figure 4 shows Graph 4 for $k = 4$.

To obtain a minimum vertex cover, we can choose the sets $V_1 \cup V_3 \cup V_5$ or $V_2 \cup V_4 \cup V_6$. An optimal vertex cover, therefore, is of size $k + 2\log k$. A bad initial solution for the (1+1) EA is produced if all vertices $V_1 \cup V_2$ are chosen but the vertices from $V_3 \cup V_6$ are missing. This happens, e. g., if the maximal matching chooses the edges between V_1 and V_2 and those between V_4 and V_5. The following theorem shows that it is really hard to obtain improvements from this initial solution.

Theorem 4. *Suppose the maximal matching component of the (1+1) EA$_M$ creates the cover $V_1 \cup V_2 \cup V_4 \cup V_5$ for Graph 4. Then with probability $1 - 2^{-\Omega(\log^2 n)}$, the (1+1) EA needs $2^{\Omega(\log^2 n)}$ steps to obtain a solution whose approximation ratio is better than $2 - O((\log n)/n)$.*

Proof. Note that the initial cover is of approximation ratio $(2k + 2\log k)/(k + 2\log k) = 2 - O((\log n)/n)$. To remove one of the vertices in V_1 or V_2 from the cover, all vertices from the neighboring sets V_6 and V_3, respectively, must be included in the cover. The (1+1) EA can include vertices in the cover only if the number of vertices in a different set is decreased at the same time. However, for the supposed cover, the situation is similar to the proof of Theorem 3. Vertices from a subset (e. g., V_6) can only be added by swapping all vertices from this subset with all vertices from the neighboring set (e. g., V_5) in a single step. The latter has probability at most $(1/n)^{\log k} = 2^{-\Omega(\log^2 n)}$. Hence, the probability of an improvement in a phase of $2^{c \log^2 n}$ steps is still $2^{-\Omega(\log^2 n)}$ if $c > 0$ is chosen small enough but constant. □

In the examples above, the search of the (1+1) EA$_M$ is likely to get stuck at local optima since a worst-case initial maximal matching is assumed. We now turn to a more general view. In the following, we use the above-mentioned greedy algorithm to compute the maximal matching in the (1+1) EA$_M$. This means that we choose uniformly free edges until there are no such edges left. Let the obtained algorithm be called *greedy (1+1) EA$_M$* ((1+1) EA$_{GM}$).

With respect to the previous example Graphs 3 and 4, the probability that the search of the (1+1) EA leads to an optimal vertex cover seems to become higher when using the greedy maximal matching algorithm. (We do not go into the details here.) As stated above, we however cannot expect the (1+1) EA$_{GM}$ to find efficiently optimal solutions on arbitrary instances. The following example shows when it is indeed likely to run into local optima of bad quality.

We define the bipartite graph $B(k, \ell)$, $\ell < k$, on $2k + \ell$ vertices as follows.

Graph 5. *Let the vertex set be*

$$\underbrace{\{v_1, \ldots, v_k\}}_{=:L} \cup \underbrace{\{v_{k+1}, \ldots, v_{2k}\}}_{=:R} \cup \underbrace{\{v_{2k+1}, \ldots, v_{2k+\ell}\}}_{=:C}$$

and the edge set be

$$\big\{\{v_i, v_{k+i}\} \mid i = 1, \ldots, k\big\}$$
$$\cup \big\{\{v_i, v_{2k+j}\} \mid i = 1, \ldots, k,\ j = 1, \ldots, \ell\big\}.$$

Hence, we obtain the whole edge set from the induced subgraph on $L \cup R$, which is a perfect matching, and the induced subgraph on $L \cup C$, which is a complete bipartite graph. A minimum vertex cover chooses all L-vertices while any vertex cover that

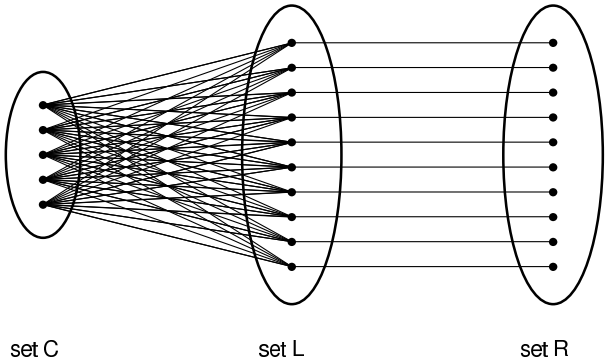

Figure 5: An illustration of Graph 5 for $k = 10$ and $\ell = 5$.

leaves out an L-vertex in favor of its adjacent R-vertex must choose all C-vertices. The following theorem (note that $k = \Omega(n)$) shows that the latter event can mislead the (1+1) EA_{GM} into a local optimum.

Theorem 5. *Let $\ell \leq k-2\log n$. Then with probability $\Omega(1-\ell/k)$, the (1+1) EA_{GM} on $B(k,\ell)$ needs at least $2^{\Omega(k-\ell)}$ steps to create a solution that is better than $(1+\ell/k)$-approximate.*

Proof. The proof outline is as follows. We first show that the maximal matching routine will choose all C-vertices with the claimed probability. Afterwards, it removes with this probability at least $\Omega(k-\ell)$ of the L-vertices from the cover before a C-vertex flips. From this situation, it takes with probability $\Omega(1)$ at least $2^{\Omega(k-\ell)}$ steps to obtain a cover that does not choose all C-vertices. Up to then, all covers contain at least $k+\ell$ vertices and are, therefore, no better than $(1+\ell/k)$-approximate.

We call the edges that are incident on C-vertices *C-edges* and the remaining edges, which are incident on R-vertices, *R-edges*. We are interested in matchings that consists of ℓ C-edges and $k-\ell$ R-edges. If already i C-edges and j R-edges have been chosen during the construction of the maximal matching, there are $\ell - i$ free C-vertices and $k-i-j$ free L-vertices left. Hence, the number of free C-edges equals $(\ell-i)(k-i-j)$ while there are only $k-i-j$ free R-edges left. The probability of choosing another R-edge in this situation is

$$\frac{(\ell-i)(k-i-j)}{(\ell-i)(k-i-j)+(k-i-j)} \;=\; \frac{\ell-i}{\ell-i+1}$$

Therefore, for $i < \ell$, the expected number of chosen R-edges between the i-th and the $(i+1)$-st choice of a C-edge is at most

$$\frac{\ell-i+1}{\ell-i} - 1 \;=\; \frac{1}{\ell-i}$$

since the random number of steps between the two choices follows a geometric distribution. Hence, the expected number of chosen R-edges until ℓ C-edges have been chosen is at most $\sum_{i=0}^{\ell-1} 1/(\ell-i) \leq (\ln \ell)+1$. By Markov's inequality, the number is at most $k-\ell$, i. e., ℓ C-edges are chosen, with probability at least $1-(\ln \ell+1)/(k-\ell)$. Due to our assumption on ℓ, the last expression is positive and bounded from below by $\Omega(1-\ell/k)$.

For the second part of the proof, we assume that a matching with ℓ C-edges and $k-\ell$ R-edges has been created and consider the subsets $L^* \subseteq L$ and $R^* \subseteq R$ consisting of the $k-\ell$ vertices in L resp. R on which the R-edges of the matching are incident. Starting from the vertex cover corresponding to the matching, the (1+1) EA accepts each step that flips only an L^*-bit. We consider the probability that an L^*-vertex flips before a C- or R^*-vertex flips. By simple calculations, this probability is bounded from below by $\Omega(1-k/\ell)$. We assume that such a flip occurs. Let v^* be the flipping L^*-vertex. As long as v^* is not added to the cover again, no C-vertices can be removed.

In the following, we are interested in the event of reducing the number of chosen L^*-vertices even further. Consider a phase consisting of the $n/4$ steps after v^* has been removed. With probability $\Omega(1)$, v^* is not chosen again during the phase. We assume this to happen. The expected number of flipping R^*-bits in the phase is bounded from above by $(k-\ell)/4$. By Chernoff bounds, the number is at most $(k-\ell)/2$ with probability $1-2^{-\Omega(k-\ell)}$. If this event holds, there are throughout the phase still $(k-\ell)/2-1$ L^*-vertices left that can be (or already have been) removed without violating the cover. We apply Chernoff-bound arguments again. Altogether, in the end of the phase, we have with probability $\Omega(1)-2^{-\Omega(k-\ell)}$ arrived at a cover that contains all C-vertices but misses at least $\Omega(k-\ell)$ of the L-vertices. We assume to be in such a situation.

To obtain a cover without C-vertices from this situation, all L-vertices have to be chosen at at least one point of time. We consider the $\Omega(k-\ell)$ R-edges whose L-vertex is unchosen. The random number of unchosen L-vertices can be increased or decreased. This process can be identified with the random walk of the (1+1) EA on a needle-in-a-haystack function where the R-edges correspond to bits and an unchosen L-vertex of an R-edge corresponds to a 0-bit and and unchosen R-vertex to a 1-bit. With $\Omega(k-\ell)$ bits and starting from $\Omega(k-\ell)$ 0-bits, the time until the all-ones string is reached is bounded from below by $2^{\Omega(k-\ell)}$ with probability $1-2^{-\Omega(k-\ell)}=\Omega(1-k/\ell)$ (using the results in [25]). □

Theorem 5 provides only a lower bound $\Omega(1-\ell/k)$ on the probability of reaching the local optimum which seems to be too pessimistic for small ℓ since the bound is only $\Omega(1)$ then. If ℓ/k converges to 0, we however conjecture that the probability converges to 1. This can be made precise for the special case $\ell=1$.

Theorem 6. *Let $\ell=1$. Then with probability $1-O(1/\sqrt{n})$, the (1+1) EA_{GM} on $B(k,\ell)$ needs at least $2^{\Omega(n)}$ steps to create a solution that is better than $(1+1/n)$-approximate.*

Proof. The proof follows the same lines as the one of Theorem 5. Using the arguments from the first part of the proof, it is easy to see that the greedy maximal matching algorithm chooses with probability at least $1-1/n$ one C-edge and $k-1$ R-edges. This case means that the initial solution of the (1+1) EA chooses the single C-vertex and $2(k-1)=n-3$ L- and R-vertices. We assume this to happen.

Next consider the phase of the $\sqrt{n}$ steps after initialization. With probability $1-O(1/\sqrt{n})$, the C-vertex does not flip in the phase. Using the Chernoff-bound arguments from the proof of Theorem 5, we prove that with probability $1-2^{-\Omega(\sqrt{n})}$, at least $\Omega(\sqrt{n})$ L-vertices are removed from the cover. Note that we need not assume that an L-vertex flips before the first R-vertex flips.

Assuming that $\Omega(\sqrt{n})$ L-vertices have been removed by the end of the phase, we apply the ideas from the last paragraph of the proof of Theorem 5. Hence, the proba-

bility that all L-vertices are in the cover again—which is necessary for the C-vertex to flip—before $2^{\Omega(n)}$ steps have elapsed is $2^{-\Omega(n)}$. □

We are left with several interesting open problems. It would be desirable to have an example where the (1+1) EA$_{GM}$ gets stuck at $(2-o(1))$-optimal solutions with probability $1-o(n^{-c})$ for any constant c, i. e., the success probability should be only super-polynomially small. Such examples should exist if the vertex cover problem does not admit approximations with a constant factor less than 2. Moreover, one could work on generalizations of Proposition 1. Since the (1+1) EA is able to flip any given subset of $\ln n/\ln\ln n$ bits in a single step with probability $n^{-O(1)}$, it might happen that it is able to explore in polynomial time the Hamming ball of radius $\ln n/\ln\ln n$ centered around the initial solution. This would imply that solutions with approximation ratio $2-O(\ln n/(n\ln\ln n))$ could be found in polynomial time with at least polynomially small probability.

5 Conclusions

In the case of combinatorial optimization problems often hybrid methods are used to obtain good solutions for a certain problem. The theoretical understanding of combining evolutionary algorithms with other methods is rather weak. We have made a first step into the rigorous analysis of such methods by considering the combination of well-known approximation algorithms with a simple evolutionary algorithm. In our analyses we have pointed out situations where approximate solutions can (or cannot) be improved by the search procedure of an EA.

There are several open questions concerning the topic of analyzing the combination of evolutionary algorithms with other methods. Some regarding the vertex cover problem and starting with solutions computed by the considered approximation algorithms have been pointed out in the Sections 3 and 4. We also want to state a more general question. The EAs considered in this paper only use an initial solution that has been computed by another method. In general a different optimization procedure is applied more often during the run of an EA. These so-called memetic approaches have already been analyzed w. r. t. their runtime on toy problems [22, 23]. It would be nice to have some results on such methods by rigorous analyses on a well-known combinatorial optimization problem.

Acknowledgments

J. He was supported by the UK Engineering and Physical Research Council under Grant No. EP/C520696/1. C. Witt was supported by the Deutsche Forschungsgemeinschaft (DFG) in terms of the Collaborative Research Center "Computational Intelligence" (SFB 531).

References

[1] Dakin, R. J. (1965). A tree-search algorithm for mixed integer programming problems. *The Computer Journal*, 8:250–255.

[2] Doerr, B., Hebbinghaus, N., and Neumann, F. (2006). Speeding up evolutionary

algorithms through restricted mutation operators. In *Proc. of PPSN '06*, volume 4193 of *LNCS*, pages 978–987.

[3] Evans, I. K. (1998). Evolutionary algorithms for vertex cover. In Porto, V. W., Saravanan, N., Waagen, D. E., and Eiben, A. E., editors, *Proc. of Evolutionary Programming VII*, volume 1447 of *LNCS*, pages 377–386, San Diego, CA, USA. Springer.

[4] Friedrich, T., He, J., Hebbinghaus, N., Neumann, F., and Witt, C. (2007). Approximating covering problems by randomized search heuristics using multi-objective models. In *Proc. of GECCO '07*, pages 797–804, London, UK. ACM Press.

[5] Giel, O. and Wegener, I. (2003). Evolutionary algorithms and the maximum matching problem. In *Proc. of STACS '03*, volume 2607 of *LNCS*, pages 415–426.

[6] Giel, O. and Wegener, I. (2006). Maximum cardinality matchings on trees by randomized local search. In *Proc. of GECCO '06*, pages 539–546. ACM Press.

[7] Glover, F. and Laguna, M. (1997). *Tabu Search*. Kluwer, Norwell, MA.

[8] Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Reading, MA.

[9] He, J., Yao, X., and Li, J. (2005). A comparative study of three evolutionary algorithms incorporating different amount of domain knowledge for node covering problems. *IEEE Trans. on Systems, Man and Cybernetics*, 35(2):266– 271.

[10] Holland, J. H. (1992). *Adaptation in Natural and Artificial System*. MIT Press, Cambridge, MA, second edition.

[11] Karakostas, G. (2005). A better approximation ratio for the vertex cover problem. In *Proc. of ICALP '05*, volume 3580 of *LNCS*, pages 1043–1050. Springer.

[12] Kehden, B. and Neumann, F. (2006). A relation-algebraic view on evolutionary algorithms for some graph problems. In *Proc. of EvoCop '06*, volume 3906 of *LNCS*, pages 147–158.

[13] Khuri, S. and Bäck, T. (1994). An evolutionary heuristic for the minimum vertex cover problem. In Hopf, J., editor, *Genetic Algorithms within the Framework of Evolutionary Computation – Proc. of the KI-94 Workshop*, pages 86–90, Saarbrücken, Germany.

[14] Michalewicz, Z. (1996). *Genetic Algorithms + Data Structure = Evolution Program*. Springer Verlag, New York, third edition.

[15] Neumann, F. (2004). Expected runtimes of evolutionary algorithms for the eulerian cycle problem. In *Proc. of CEC '04*, volume 1 of *IEEE Press*, pages 904–910.

[16] Neumann, F. (2007). Expected runtimes of a simple evolutionary algorithm for the multi-objective minimum spanning tree problem. *European Journal of Operational Research*, 181 (3):1620–1629.

[17] Neumann, F. and Wegener, I. (2004). Randomized local search, evolutionary algorithms, and the minimum spanning tree problem. In *Proc. of GECCO '04*, volume 3102 of *LNCS*, pages 713–724.

[18] Neumann, F. and Wegener, I. (2005). Minimum spanning trees made easier via multi-objective optimization. In *Proc. of GECCO '05*, pages 763–770. ACM Press.

[19] Papadimitriou, C. H. and Steiglitz, K. (1998). *Combinatorial Optimization: Algorithms and Complexity*. Dover, Mineola, NY.

[20] Scharnow, J., Tinnefeld, K., and Wegener, I. (2004). The analysis of evolutionary algorithms on sorting and shortest paths problems. *Journal of Mathematical Modelling and Algorithms*, pages 349–366.

[21] Schrijver, A. (1998). *Theory of Linear and Integer Programming*. John Wiley and Sons.

[22] Sudholt, D. (2006a). Local search in evolutionary algorithms: the impact of the local search frequency. In *Proc. of ISAAC '06*, volume 4288 of *LNCS*, pages 359–368. Springer.

[23] Sudholt, D. (2006b). On the analysis of the (1+1) memetic algorithm. In *Proc. of GECCO '06*, pages 493–500. ACM Press.

[24] Vazirani, V. (2001). *Appromixation Algorithms*. Springer.

[25] Wegener, I. and Witt, C. (2005). On the optimization of monotone polynomials by simple randomized search heuristics. *Combinatorics, Probability and Computing*, 14:225–247.

[26] Witt, C. (2005). Worst-case and average-case approximations by simple randomized search heuristics. In *Proc. of STACS '05*, volume 3404 of *LNCS*, pages 44–56.