# Analysis of Population-based Evolutionary Algorithms for the Vertex Cover Problem

Pietro S. Oliveto, *Student Member, IEEE*, Jun He, *Member, IEEE*, Xin Yao, *Fellow, IEEE*

***Abstract*— Recently it has been proved that the (1+1)-EA produces poor worst-case approximations for the vertex cover problem. In this paper the result is extended to the (1+$\lambda$)-EA by proving that, given a polynomial time, the algorithm can only find poor covers for an instance class of bipartite graphs. Although the generalisation of the result to the ($\mu$+1)-EA is more difficult, hints are given in this paper to show that this algorithm may get stuck on the local optimum of bipartite graphs as well because of premature convergence. However a simple diversity maintenance mechanism can be introduced into the EA for optimising the bipartite instance class effectively. It is proved that the diversity mechanism combined with one point crossover can change the runtime for some instance classes from exponential to polynomial in the number of nodes of the graph.**

## I. Introduction

Evolutionary Algorithms (EAs) are randomised search heuristics often used for solving combinatorial optimisation problems [1]. Although the theoretical analysis of EAs is way behind compared to the amount of practical applications, in recent years a fair amount of work has been accomplished in the study of the computational complexity of the (1+1)-EA for combinatorial optimisation problems [2]. However, theoretical results about population-based EAs are fewer compared to those related to the (1+1)-EA. Jansen et. al. have analysed the (1+$\lambda$)-EA on pseudo-boolean functions such as Onemax and Leading ones [3]. Similar work has been produced by Witt concerning the ($\mu$+1)-EA [4]. He and Yao have compared single individuals against populations also on pseudo-boolean problems [5]. The latter work only considers single bit mutations. The results available about population-based EAs for "real-world" combinatorial optimisation problems are even fewer. Some results are available about the (1+$\lambda$)-EA for the spanning tree problem [6], and about the ($\mu$+1)-EA for finding Cliques on semi-random sparse graphs [7]. An analysis of a (2N+2N)-EA on some instances of the subset sum problem is the only available result about population-based EAs for NP-Hard problems [8].

In this paper we make a first step in the analysis of population-based EAs for vertex cover, a well known NP-Hard combinatorial optimisation problem that has applications in fields like scheduling or networking. When analysing an algorithm for an NP-Hard problem, it cannot be expected that for every instance the optimal solution is found in polynomial time, unless P=NP. As a consequence, an important theoretical result is that of finding the worst-case approximation ratio of the algorithm for the problem. Recently results in this direction have been achieved concerning the (1+1)-EA. In particular, Friedrich et al. have proved that the worst case approximation ratio of the (1+1)-EA can be arbitrarily bad [9]. Oliveto et al. also show that the previous result does not hold if a restart strategy is used [10] and that the (1+1)-EA cannot achieve better approximations compared to problem specific algorithms even if a clever restart strategy is used [11]. This paper presents a first extension of the (1+1)-EA results to population-based EAs. To gain a better understanding of how multiple individuals explore the landscape, *parent* and *offspring* populations are analysed separately. Our main concern, in this first step, is to understand if populations can overcome the problems that single individuals encounter when optimising a vertex cover problem. After having understood if such limitations also hold for populations then we may attempt to search for other characteristics that may trouble population-based algorithms for vertex covering.

The paper is structured as follows. Section II introduces the vertex cover problem and the algorithms analysed in this paper. In section III *offspring* populations are analysed and it is shown that the (1+$\lambda$)-EA can produce arbitrarily bad approximate solutions, just like the (1+1)-EA. In section IV *parent* populations are examined and it is shown that also the ($\mu$+1)-EA may have to use a restart strategy to avoid getting trapped in local optima. However, the use of a very simple diversity mechanism may assure that the ($\mu$+1)-EA avoids the problem. In section V a first example is given of how the simple diversity mechanism combined with one point crossover may drastically change the runtime of the ($\mu+1$)-EA from exponential to polynomial in the number of nodes of the graph. In the last section we draw our conclusions and discuss further work.

## II. Preliminaries

Given an undirected graph $G(V, E)$ with $V$ the set of vertices and $E$ the set of edges, the vertex cover problem is that of finding the minimum subset $C$ of $V$ such that for any edge $e \in E$ at least one of its endpoints is in $C$. All the subsets $C$ of $V$ having, for each edge $e \in E$, one of its endpoints in $C$ are called *covers*. All the other subsets are called *infeasible* solutions. In order to try and understand how and why populations may be useful for optimising covering problems, in this paper two very simple population-based algorithms, derived from the field of Evolution Strategies (ESs), will be mainly analysed.

Pietro S. Oliveto, Jun He and Xin Yao are with the Centre of Excellence for Research in Computational Intelligence and Applications (CERCIA), School of Computer Science, University Of Birmingham, Edgbaston,Birmingham, B15 2TT, U.K. email: {P.S.Oliveto,J.He,X.Yao}@cs.bham.ac.uk.

978-1-4244-1823-7/08/$25.00©2008 IEEE

The (1+$\lambda$)-EA is probably the most simple *offspring* population algorithm. It works as follows:

**(1+$\lambda$)-EA**

```
1) Choose $x \in \{0,1\}^n$ uniformly at random.
2) Repeat: For each $i \in 1 \dots \lambda$ do:
    a) Create $x'_i \in \{0,1\}^n$ by flipping each bit in $x$ with probability $1/n$.
    b) Choose $z$ uniformly from $\{x'_1, x'_2, \dots, x'_\lambda\}$ with minimal fitness.
    c) If $f(z) \leq f(x)$ then let $P_{t+1} = \{x'_1, x'_2, \dots, x'_\lambda\}$.
    d) Otherwise, let $P_{t+1} = \{x, x'_1, x'_2, \dots, x'_\lambda\} \setminus \{z\}$.
```

The most simple *parent* population-based EA is probably the ($\mu$+1)-EA which is also derived from the ES field. The algorithm works as follows:

**($\mu$+1) EA**

```
1) Choose an initial population $P_0$ of $\mu$ individuals uniformly at random.
2) Repeat:
    a) Choose $x \in P_t$ uniformly at random
    b) Create $y$ by flipping each bit in $x$ with probability $1/n$.
    c) Choose $z \in P_t$ with minimal fitness
    d) If $f(y) \leq f(z)$, then $P_{t+1} = P_t \setminus \{z\} \cup \{y\}$,
    e) otherwise $P_{t+1} = P_t$.
```

A Random Local Search (RLS) version of both the above algorithms is obtained if the mutation operator just flips one bit of the parent chosen randomly instead of each bit with probability $1/n$.

The algorithms are described with their common uniform initialisation. However, for simpler analyses and easier proof understanding, in this paper the algorithms will be initialised with all the nodes in the cover. Previous work for the (1+1)-EA has shown that there is no significant difference in the performance between empty, full or uniform initialisation [11]. The above algorithms will be compared to the (1+1)-EA on significant instance classes to try and understand when populations are able to overcome the problems encountered by single individuals. A simple diversity mechanism will also be considered to avoid the premature convergence of the algorithms. This mechanism will be presented in section IV and in section V will be combined with One-point crossover to present an example of a vertex cover landscape where both populations and crossover are essential for finding the minimum cover.

For the algorithms to be adapted to optimise a vertex cover instance, each subset $C \in V$ is represented by a bit-string $(s_1 \dots s_n)$ where $n$ is the number of vertices $v \in V$ of the graph $G = (V, E)$ connected by edges $e \in E$. For each node $v_i$ belonging to the subset $C$ the relative bit $s_i$ is set to 1. Otherwise it is set to 0. Given the above representation, the same fitness function used in the analysis of the (1+1)-EA for vertex cover will be adopted [9], [10]:

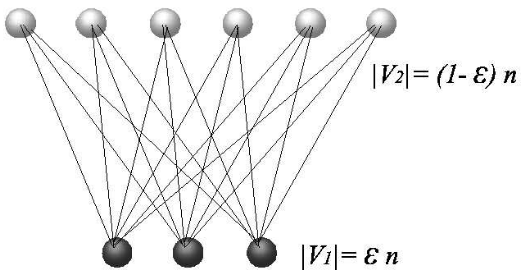

Fig. 1. The optimal cover of the bipartite graph with $n = 9$ and $\epsilon = \frac{1}{3}$.

$$f(C) = \sum_{i=1}^{n} \left( s_i + n(1-s_i) \sum_{j=1}^{n} (1-s_j) e_{i,j} \right)$$

Here the first part of the formula counts the number of nodes in the cover and the second part gives a penalty to each uncovered edge $e$. As a result, any *cover* has a better fitness value than that of an *infeasible* solution.

In general, the performance of the algorithms will be evaluated according to their worst-case *approximation ratio* for the vertex cover problem and the expected runtime required for the obtainment of a given approximation quality. The worst-case approximation ratio of an algorithm $A$ on a minimisation problem $R$ is defined as

$$\max_{I \in R} \frac{A(I)}{OPT(I)}$$

where $A(I)$ is the solution obtained by $A$ on the instance $I$ and $OPT(I)$ is the value of the best solution of $I$. As usually done in the computational complexity analyses of EAs we will use the expected number of fitness function evaluations for an event to happen for the first time (i.e. the global optimum or an approximate solution) as the expected runtime of the algorithm relative to the event [2].

## III. Offspring populations

A bipartite instance class of graphs has been introduced in [9] to show that the approximation ratio of the (1+1)-EA for the vertex cover can be arbitrarily bad. The instance class is depicted in figure 1. There are two covers the (1+1)-EA may find in polynomial time: a global optimum containing the subset of nodes $V_1$ of size $\epsilon n$ and a local optimum containing the subset $V_2$ of size $(1-\epsilon)n$, with $\epsilon < 1/2$. There is a constant probability that the algorithm finds either cover hence if $\epsilon$ is sufficiently small the approximate solution found in polynomial time may be very bad with constant probability. This leads to an exponential expected optimisation time for the (1+1)-EA. In [10] it has been pointed out that if a restart strategy is used then the (1+1)-EA can find the minimum cover of the instance class in time $O(n \log n)$. The proof exploits the fact that there is a constant probability of finding the global optimum in $O(n \log n)$ time, hence a constant number of restarts are sufficient to optimise the graph with the same asymptotic time. We will try to understand if by using a population it is possible to avoid the restart strategy. Since various individuals explore the landscape at the same time, the hope would be that both optima are eventually found by different individuals.

In this section we analyse the (1+$\lambda$)-EA, a simple algorithm that uses *offspring* populations. *Parent* populations will be considered in the next section. In the Evolutionary Computation (EC) field it is generally accepted that *offspring* populations should be used to give greater emphasis to local exploration. In fact, at each step, the (1+$\lambda$)-EA creates $\lambda$ offspring that are local neighbours of their parent and selects the best found solution in the neighbourhood for the following generation. As a consequence, it would not be expected that an algorithm like the (1+$\lambda$)-EA may have greater global search capabilities compared to the (1+1)-EA. The following two theorems prove that this is the case for the (1+$\lambda$)-RLS and the (1+$\lambda$)-EA for the bipartite instance class.

*Theorem 1:* The (1+$\lambda$)-RLS algorithm, initialised with all nodes in the cover, has infinite expected optimisation time for the bipartite instance class. With probability at least $1-\epsilon$, the (1+$\lambda$)-RLS algorithm finds the minimum cover in time $O(\lambda n + n\log n)$.

*Proof:* For each individual generated at the first step, the probability it has a $V_2$ node removed is $\frac{(1-\epsilon)n}{n} = 1-\epsilon$ and that it is selected for the next generation is $1/\lambda$. So the probability that after the first generation an individual without a $V_2$ node survives (Event $E_1$) is

$$\frac{1-\epsilon}{\lambda} + \frac{1-\epsilon}{\lambda} + ... + \frac{1-\epsilon}{\lambda} = \frac{(1-\epsilon)\lambda}{\lambda} = 1-\epsilon$$

Once this event has occurred no moves removing any $V_1$ node will be accepted because an infeasible solution would be created. The expected time for the remaining $V_2$ nodes to be removed from the cover is $O(\lambda n + n\log n)$, with similar arguments to those used for the proof of the (1+$\lambda$)-EA for Onemax [3]. However, with probability at least $\epsilon$ event $E_1$ does not happen, while an individual without a $V_1$ cover node is generated and selected for survival at the first step. In such a case, the (1+$\lambda$)-EA will end up on the local optimum and will never escape because it only flips one bit at a time. ■

*Theorem 2:* Let $\lambda \le n$. Then the (1+$\lambda$)-EA has an exponential expected optimisation time for the bipartite instance class. With constant probability the (1+$\lambda$)-EA finds the minimum cover in time $O(\lambda n + n\log n)$.

*Proof:* The following phases are considered:

1) After two generations, with constant probability, the individual selected for survival has two $V_2$ nodes removed from the cover.
2) With a probability greater than $1/e$ the algorithm removes other $\frac{n}{2e}-1$ $V_2$ nodes before removing any $V_1$ node.
3) With overwhelming probability the algorithm finds the global optimum in time $O(\lambda n + n\log n)$.

With probability at least

$$\frac{(1-\epsilon)n}{n}\left(1-\frac{1}{n}\right)^{n-1} \ge \frac{1-\epsilon}{e} \ge \frac{1}{2e} = c'$$

an individual without a $V_2$ node is generated and selected for survival.

With probability at least $\frac{(1-\epsilon)n-1}{ne} = c''$ an individual without another $V_2$ node is generated and selected for the second generation. Hence with constant probability $c = c' \cdot c''$ the first phase occurs.

In the second phase, the expected time for removing another $V_2$ node (Event $E_2$) is greater than

$$\frac{(1-\epsilon)n-2}{en} \ge \frac{(1-\epsilon)n}{2en} > \frac{1}{4e}$$

Here the first inequality holds as long as $n$ is not really small (i.e. $(1-\epsilon)n > 3$).

On the other hand the highest probability that the direction of the process is inverted to head towards the local optimum occurs by flipping four bits (if another $V_2$ node is not removed from any other individual heading towards the global optimum) or five bits, three from the set $V_2$ and two from the set $V_1$ (Event $E_1$). Such a probability is at most:

$$\lambda\binom{\epsilon n}{2}\frac{1}{n^4}\left(1-\frac{1}{n}\right)^{n-4} \le \frac{n(\epsilon n)(\epsilon n-1)}{2n^4} \le \frac{1}{n}$$

So, the probability that $E_2$ happens before $E_1$ once is bounded below by

$$P(E_2|E_1 \cup E_2) \ge \frac{1/(4e)}{1/(4e)+1/n} = \frac{n}{n+4e}$$

$$= 1 - \frac{4e}{n+4e} \ge 1 - \frac{4e}{2n} = 1 - \frac{2e}{n} \qquad (1)$$

If event $E_2$ does occur before event $E_1$, the inversion probabilities get lower since an extra $V_1$ bit has to flip for the inversion to occur, while the probability of removing an extra $V_2$ node remains constant. Hence inequality 1 is also lower bound for the probability of removing the other $V_2$ nodes before an inversion occurs. Then the probability of removing another $\frac{n}{2e}-1$ $V_2$ nodes before any $V_1$ node is greater than

$$\left(1-\frac{2e}{n}\right)^{\frac{n}{2e}-1} \ge \frac{1}{e}$$

This proves that phase 2. happens with probability at least $1/e$.

At this point, there is exponential expected time before any $V_1$ node is removed (i.e. at least $\frac{\epsilon n}{2e}$ nodes have to flip) while, in a similar way to that of the (1+$\lambda$)-EA for Onemax [3], it is proved that in time $O(\lambda n + n\log n)$ the global optimum will have been found.

The proof for the algorithm ending on the local optimum instead of the global one is similar. When the algorithm is on the local optimum, the probability it escapes is less than $\binom{(1-\epsilon)n}{\epsilon n}\frac{1}{n^{2\epsilon n}}$ leading to an exponential expected time for any improving move to be accepted, if $\epsilon$ is not too small. ■

Theorems 1 and 2 extend the worst-case approximation ratio result of the (1+1)-EA [9] to the (1+$\lambda$)-RLS algorithm and to the (1+$\lambda$)-EA. The approximation of the algorithm for the instance class is:

$$\frac{(1-\epsilon)n}{\epsilon n} = \frac{1-\epsilon}{\epsilon}$$

The smaller the $\epsilon$ value, the worse is the approximation.

Theorem 2 only holds for $\lambda \leq n$. However, in [3] it is proved that for the Onemax function, if the population size is greater than $\Theta(\frac{(\log n)(\log n \log n)}{\log\log\log n})$ then the optimisation time increases significantly. Such a result is easily extended to the bipartite instance class. Hence, if $\lambda > n$ and assuming the algorithm finds the global optimum, the runtime would be higher than that of the (1+1)-EA using a restart strategy (i.e. $O(n \log n)$ [10]). As a consequence, in this "worst-case scenario", a (1+1)-EA with a restart strategy has a better performance compared to the (1+$\lambda$)-EA.

## IV. Parent populations

If the results of the previous section may not be particularly surprising, it is more interesting to analyse *parent* populations for the global exploration of landscapes. The most simple *parent* population-based EA is the ($\mu$+1)-EA. If there are many individuals exploring the landscape at the same time there should be greater hope that both covers, the local and the global optimum, are eventually found. The following theorem proves that this does not happen, at least if the difference in cover size between the local optimum and the global one is not too big.

*Theorem 3:* Let $G(n)$ be the bipartite instance class with $|V_2| - |V_1| = c$ where $c > 1$ is a constant, $n = |V_1| + |V_2|$ and $\mu = o(\frac{n}{\log^2 n})$. If the local optimum is found before the minimum cover, then with probability $1 - o(1)$ all the $\mu$ individuals reach the local optimum before finding the optimal cover. Conditional to the above event, the expected runtime of the ($\mu$+1)-EA is at least $n^{\Omega(n)}$. However, the expected runtime of the ($\mu$+1)-EA to reach either the local or the global optimum is $O(\mu n + n \log n)$.

*Proof:* We assume the local search cover (i.e. the local optimum) is found before the minimum cover (i.e. the global optimum). Let $x_1$ be the first individual to reach the local search cover.

Furthermore, let the *takeover time* $T$ be the time for all the individuals heading towards the global optimum (i.e. removing $V_2$ nodes) to be substituted by offspring heading towards the local optimum.

The probability that $x_1$ creates a clone is $\frac{1}{\mu}(1-\frac{1}{n})^n$. When there are $i$ $V_1$ individuals on the local search cover, the probability of creating a clone is $\frac{i}{\mu}(1-\frac{1}{n})^n$. Hence the expected takeover time $T$ is:

$$E(T) \leq (1-1/n)^{-n} \sum_{i=1}^{\mu} \frac{\mu}{i} = O(\mu \log \mu) = O(\mu \log n)$$

By Markov's inequality, the probability the takeover requires more than $\mu \log^2 n$ steps is:

$$P(T \geq \mu \log^2 n) \leq \frac{\mu \log n}{\mu \log^2 n} = \frac{1}{\log n} = o(1)$$

However, in the mean time the *takeover* may have been avoided because an individual heading towards the optimum *catches up*. This may only happen if such an individual has removed at least $|V_1|$ nodes from the $V_2$ subset. This would mean the individual has the same fitness as the local optimum or less. In the following, it will be proved that this only happens with a probability of $o(1)$.

Optimistically assuming that the individuals heading towards the global optimum are only one fitness level behind the local optimum individual, the probability that at least another $V_2$ node is removed from the cover, hence an individual catches up by reaching at least the same fitness level of $x_1$ is at most

$$\frac{\mu-1}{\mu} \sum_{i=1}^{c+1} \binom{c+1}{i} \frac{1}{n^i} \leq \sum_{i=1}^{c+1} \left(\frac{e(c+1)}{i}\right)^i \frac{1}{n^i}$$
$$= \sum_{i=1}^{c+1} \left(\frac{e(c+1)}{in}\right)^i \leq \sum_{i=1}^{c+1} \frac{e(c+1)}{n} = \frac{e(c+1)^2}{n}$$

The last inequality holds if $n$ is large enough.

So the probability a $V_2$ individual *catches up* in a phase of $\mu \log^2 n$ steps is:

$$\mu \log^2 n \frac{e(c+1)^2}{n} = o\left(\frac{n}{\log^2 n}\right) \frac{e(c+1)^2 \log^2 n}{n} = o(1)$$

This proves that with probability $1 - o(1)$ all the individuals reach the local optimum in time $O(\mu \log^2 n)$. The closest cover with at least the same fitness value is obtainable by flipping $2|V_1|$ nodes. These have to be all the nodes belonging to the subset $V_1$ and at least other $|V_1|$ nodes belonging to the subset $V_2$. The probability this happens is bounded above by $\frac{\mu}{\mu} \frac{1}{n^{|V_1|}}$ because at least all the $V_1$ nodes have to flip at the same time. Since, $|V_1| = n/2 - c/2$ the expected time for the above event to happen is at least $n^{\Omega(n)}$.

On the other hand, the expected runtime for finding the minimum either cover. The proof of the last statement is similar to that of the ($\mu$+1)-EA for Onemax [12]. ■

The theorem shows that the same problem encountered by the (1+1)-EA may also happen for the ($\mu$+1)-EA. Since the local and global cover sizes are not too different the result is not strong enough to guarantee poor approximation ratios. Furthermore, Theorem 3 assumes the local optimum is found before the optimal cover. Since the two cover sizes only differ in a constant number of nodes the assumption may be reasonable. However, a greater understanding of the dynamics of the ($\mu$+1)-EA is needed to derive the probability it finds the local optimum before the global one. In any case, in the following it will be shown how a simple diversity mechanism avoids the problem.

Rather than replacing the worst individual of the population, in the following algorithm each offspring replaces its parent if its fitness is at least as good.

**($\mu$+1)-EA with a simple diversity mechanism**

```
1) Choose an initial population $P_0$ of $\mu$ individuals uniformly at random;
2) Repeat
    a) Choose $x \in P_t$ uniformly at random;
    b) Create $y$ by flipping each bit in $x$ with probability $1/n$;
    c) If $f(y) \leq f(x)$, then $P_{t+1} = P_t \setminus \{x\} \cup \{y\}$;
    d) otherwise $P_{t+1} = P_t$.
```

*Theorem 4:* With probability at least $1 - (1 - \frac{1}{2e})^{\mu} = 1 - c^{\mu}$, at least one individual of the ($\mu$+1)-EA with the simple diversity mechanism, initialised with all the nodes in the cover, finds the minimum cover of the bipartite instance class. The expected time for each individual to find either the local or the global optimum is $O(\mu n \log n)$.

*Proof:* The main idea behind the proof is that each individual of the population evolves *independently* from the others because of the diversity mechanism. All there is to prove is that at least one individual will reach the minimum cover with very high probability.

The probability an individual removes at least a $V_2$ node from the cover before a $V_1$ node is greater than $\frac{(1-\epsilon)n}{n} = 1-\epsilon > 1/2$. This is true whatever number of bits flip because given an $i$-bitflip $\binom{|V_2|}{i} > \binom{|V_1|}{i}$ and $\epsilon < 1/2$. At this point, the minimum number of bits that have to flip to remove any $V_1$ node is two, the fixed one from the $V_2$ subset and any other from the $V_1$ subset. Since at least one precise bit from $V_2$ has to flip, the probability is bounded from above by $1/n$. In a similar way as for the (1+$\lambda$)-EA it is proved that with probability at least $1/e$ other $\epsilon n$ nodes of $V_2$ are removed from the cover.

The probability another $V_2$ node is removed is $\frac{(1-\epsilon)n-1}{n} \geq 1/2$. Hence an inversion may happen with probability at most

$$\frac{1/n}{1/n + 1/2} = \frac{2}{n+2} \leq \frac{2}{n}$$

and it does not happen with probability at least $p_{V_2} = (1 - 2/n)$. From this point of time on, at least two *fixed* bits will have to flip for an inversion to occur. This means that $P_{V_2}$ is an upper bound on the probability that an extra $V_2$ node is removed before an inversion occurs at any time-step. It follows that with probability at least

$$\left(1 - 2/n\right)^{\epsilon n} \geq \left(1 - 2/n\right)^{\frac{n}{2}-1} \geq \frac{1}{e}$$

another $\epsilon n$ nodes belonging to the subset $V_2$ are removed from the cover.

Multiplying we get a probability of at least $1/(2e)$ for each individual to reach this situation ($S_1$). Each individual that reaches this point will find the global optimum with probability greater than $1 - e^{-\Omega(n)}$ because at least $2\epsilon n = \Omega(n)$ bits have to flip to invert the direction of the process. On the other hand, in at most $O(\mu n \log n)$ steps all the other $V_2$ nodes will have been removed. Hence, it has to be proved that $S_1$ happens for at least one individual. The probability $S_1$ does not happen for one individual is $c \leq 1 - 1/(2e)$, and with probability

$$c^{\mu} \leq \left(1 - \frac{1}{2e}\right)^{\mu}$$

it does not happen for any individuals in the population. This proves that with probability at least $1-c^{\mu}$ the minimum cover will be found. ■

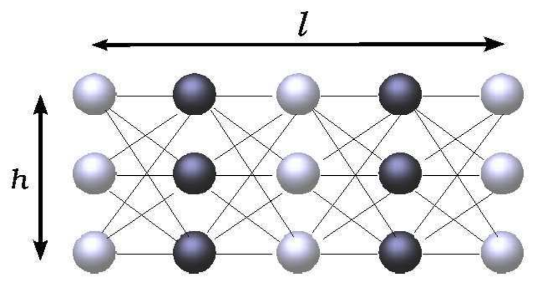

Fig. 2. The optimal cover for the new $G_{3,5}$ graph.

The presented diversity mechanism is so simple it may be argued it just resembles a *parallel* (1+1)-EA. In fact the ($\mu$+1)-EA with such diversity mechanism does have $\mu$ individuals that explore the landscape independently. However in the following section it is shown how, by combining the diversity mechanism with other operators, the individuals may cooperate together and be even more effective.

## V. Crossover

The $G_{h,l}$ instance class, illustrated in figure 2, was presented in [11]. It was proved that, if parameters $h$ and $l$ both grow with the instance size $n$, then the (1+1)-EA requires exponential time to optimise the graph class, even if a restart strategy is used. Since even a restart strategy is not effective, there are little hopes that just by incrementing the population size the results may get better. The following theorem shows that the ($\mu$+1)-RLS with the diversity mechanism and one point crossover may drastically turn the runtime from exponential to polynomial. The algorithm works as follows:

**($\mu$+1)-RLS with a simple diversity mechanism and one point crossover**

```
1) Initialise a population of $\mu$ individuals;
2) While the termination condition is not satisfied do:
    a) With probability $p_c$ do steps:
        i) Crossover: Choose two individuals $x_1$ and $x_2$ at random and apply one point crossover to obtain $x'_1$ and $x'_2$;
        ii) Selection: If $x'_1$ and $x'_2$ are feasible solutions and if $f(x'_1) \leq f(x_1)$ or $f(x'_1) \leq f(x_2)$ or $f(x'_2) \leq f(x_1)$ or $f(x'_2) \leq f(x_2)$ then replace $x_1$ and $x_2$ with $x'_1$ and $x'_2$;
```

b) **Mutation:** Choose one individual $x$ at random and mutate each of its bits with probability $p_m$ to obtain $x'$;
c) **Selection:** If $f(x') \leq f(x)$ then replace $x$ with $x'$;

3) Return the best individual in the population.

The instance class is encoded using a *column-wise* representation. The first $h$ bits of the bit-string representing a candidate solution refer to the $h$ nodes of the first column. The next $h$ bits refer to the second column nodes and so on.

*Theorem 5:* With probability at least $(1-2p_c)^{\frac{\mu l}{4}} - (2/3)^{\mu}$ the RLS algorithm with the diversity mechanism and the one-point crossover operator finds the minimum vertex cover of the $G_{h,l}$ graph, in polynomial time $O(\frac{\mu^2 n}{p_c} + \mu n \log n)$ if $h = l$ and $h \cdot l = n$ where $n$ is the size of the graph.

The theorem presents a relationship between the probability that the algorithm finds the optimum (i.e. $(1-2p_c)^{\frac{\mu l}{4}} - (2/3)^{\mu}$) and the runtime for the minimum cover to be obtained (i.e. $O(\frac{\mu^2 n}{p_c})$). The success probability depends on the crossover probability $p_c$ and on the population size. The greater the population size, the higher are both the optimisation probability and the runtime. Let, for example, $\mu = \sqrt{n}$. Then $(2/3)^{\sqrt{n}}$ is exponentially small. Hence, with a probability of about $(1-p_c)^{n/4}$ the algorithm finds the optimum in time $O(\frac{n^2}{p_c})$. Now the relationship between the crossover probability and the runtime should be clearer. The smaller is $p_c$ the higher are both the success probability and the runtime. Let, for example $p_c = \frac{2}{n}$. Then with a constant probability of $(1-\frac{4}{n})^{n/4} \approx 1/e$ the algorithm finds the minimum cover in time $O(n^3)$. If, instead, $p_c = \frac{2}{n^2}$, then the runtime is $O(n^4)$ with a probability of at least

$$\left(1-\frac{4}{n^2}\right)^{\frac{n}{4}} \geq 1 - \frac{4}{n^2}\frac{n}{4} = 1 - \frac{1}{n} = 1 - o(1)$$

Higher success probabilities can be obtained with smaller values of $p_c$.

It can be proved that after a crossover has occurred, the selection operator accepts the obtained individuals as long as they are both feasible. The following lemma explains how the first selection operator works after two individuals have been crossed over.

*Lemma 1:* The individuals produced by the crossover operator are always accepted by the selection operator if they are two feasible solutions.

*Proof:* The selection operator accepts the two new created individuals if the two following conditions both hold:

1) The two individuals are both feasible solutions;
2) At least one of the offspring has a fitness value that is not higher than that of one of its parents.

The second condition has to be proved. We obviously assume the two obtained solutions are feasible otherwise the hypothesis of the theorem are not satisfied. Let $x$ and $y$ be the parents, and $x_1$ be the "left" chromosome part of $x$ and $x_2$ be the "right" part. Furthermore, let $|x|$ be the number of ones in chromosome $x$. So $z_1 = x_1 y_2$ is one of the offspring and $z_2 = y_1 x_2$ is the other one. If $|x_1| \leq |y_1|$ then $f(z_1) = f(x_1 y_2) \leq f(y)$. On the other hand if $f(y_1) \leq f(x_1)$ then $f(z_2) = f(y_1 x_2) \leq f(x)$. Hence one of the offspring is always at least as fit as one of its parents. ■

In [10] it has been proved that local search covers of this instance class are at most of size $(2/3)lh$. This worst "local search cover", is composed of two columns of nodes in the cover out of each subset of three adjacent columns. In any case each column either has all its nodes in the "local search cover" or all its nodes are not in the "local search cover". The optimum, as depicted in figure 2, has $\lfloor \frac{l}{2} \rfloor$ columns of nodes in the cover.

The crossover operator has no influence in the process until at least two individuals reach their respective "local search covers". At this point, each of them will have a certain number of subsets of three adjacent columns with two columns of nodes in the cover (i.e. if there are no such subsets then the individual either represents the global optimum or its opposite and the probability either of these is obtained is exponentially small; there are at most $l/3$ of these subsets [10]). The main idea behind the proof is that the crossover operator can rearrange the order of these columns so that three adjacent column are obtained. When this happens, the mutation operator can remove the middle column nodes as they are not connected with any node that is not in the cover. When all the middle nodes are removed, the current solution will have been improved by a value of $h = \sqrt{n}$. In the following, a representation similar to that used in the schema theory [13] will be used to make the above ideas precise.

Let $S_1$ be the following schema: $0110 * \cdots *$ of length $l$. Here the character at position $i$ in the schema represents the value of all the $h$ bits of the $i_{th}$ column. The considered schema is called $S_1$ because the pattern $0110$ starts at position 1. So, the schema $*^{i-1}0110 * \cdots *$ will be called $S_i$ since the $0110$ pattern starts at position $i$. Furthermore, let an individual of schema $S_i$ be "improved" when its defining pattern (i.e. $0110$) is changed into either of the two following patterns: 0101 or 1010. If the defining pattern of an $S_i$ schema is turned into a $01110$ pattern, then it is called "nearly improved" because the middle 1-column may be "quickly" removed by single bit mutations. If this happens, the pattern will end up being "improved".

*Lemma 2:* The only way the crossover operator may disrupt a schema $S_i$ of a local optimum is by crossing it over with an $S_{i-1}$ or an $S_{i+1}$ schema. The two produced individuals are "nearly improved".

*Proof:* The proof follows from the following two points:

1) If the crossover point is chosen before or after the $0110$ pattern, then the schema is not disrupted.
2) If the crossover point is chosen in the $0110$ pattern, then the only schemata that may produce a pair of different feasible individuals are $S_{i+1}$ and $S_{i-1}$.

Point 1. is obvious. Point 2. is true because the only patterns that can occur in feasible local optima are 1010, 0101, 1011, 1101 and obviously 0110 (two adjacent 0-columns represent an infeasible solution). The first two, according to which crossover point is chosen, and in which position are overlapped may either produce infeasible solutions or the original bit-strings. The second two patterns in position $i-1$ or $i+1$ produce the same schemata as that of their parents or infeasible ones, while in position $i-2$ they are actually the $S_{i-1}$ and $S_{i+1}$ schemata, the first is followed by a 0 (i.e. 10110) and the second has a 0 before it (i.e. 01101) (otherwise the individuals would not be on a local optimum). Hence, the only schemata that may produce a feasible solution through crossover are the $S_{i-1}$ and $S_{i+1}$ schemata. The two produced patterns in replacement of the 0110 ones are, are 0101 and 01110 or the same. ■

For the two new individuals to be "improved" rather than "nearly improved" the middle one-column of the second pattern has to be flipped into a zero-column.

In order to guarantee that the optimum is found by combining "useful" crossovers and mutations of the middle columns, it has to be guaranteed that all the $S_i$ schemata are in the population. In this case the crossover operator will have a chance of improving all the sub-patterns and eventually reach the global optimum.

*Proof:* [Of Theorem 5] First it will be proved that with overwhelming probability all the necessary schemata are generated in the population. As always, the algorithm is initialised with a full cover. Let $E_i$ be the event that column $i$ has a node removed before columns $i-1$ and $i+1$. Event $E_i$ happens with probability $p(E_i) \geq \frac{h}{3h} = \frac{1}{3}$. So the probability that $E_i$ does not happen is $1-p(E_i) \leq 2/3$, and the probability that the event does not happen for any individual in the population is less than $(2/3)^{\mu}$. If $\mu$ is not too small, then $E_i$ happens to at least one individual of the population with overwhelming probability.

The crossover operator is not able to remove the 0-bit of the schema from the population. The only way the 0-bit may be "lost" is that the individual containing it is removed from the population. This only happens when another individual, at least as fit, is generated by mutation on itself (i.e. because of the diversity mechanism). Since the algorithm only flips one bit per generation and since if the zero-bits are flipped into one-bits, then the obtained individual would have lower fitness, the local optimum of a 4-column pattern will be reached. Only two possible schemata can be obtained starting with a 0-column at position $i$: $*^{i-1}0101*...*$ and $*^{i-1}0110*...*$. The former pattern has the defining schema already optimised. If the latter pattern is obtained, then it will have to be "improved" by a crossover with an $S_{i-1}$ or an $S_{i+1}$ schema (lemma 2).

With the same probability (i.e. $1-(2/3)^{\Omega(n)}$), individuals reaching the local optimum pattern of all the other $S_i$ schemata will be created, so will also be in the population. Hence all the schemata $S_1 \ldots S_l$ will be in the population when all the $\mu$ individuals reach their respective local optimum points. The expected time for this to happen is $O(\mu n \log n)$.

Once all the $\mu$ individuals have reached their local optima no bit flip mutations will be accepted hence any improvement relies on the crossover operator. The only accepted crossovers involve individuals belonging to the $S_i$ schema with individuals of either the $S_{i-1}$ or the $S_{i+1}$ schemata (lemma 2).

The probability that the crossover operator does an "improving" move is at least

$$P_{\text{imp}} \geq \frac{1}{\mu(\mu-1)}\frac{h}{n} \geq \frac{1}{\mu^2 l}$$

Two individuals $x$ and $y$ are created. Let $y$ be the one with the 01110 pattern.

At this stage, two other "kinds" of crossover of $y$ concerning the same pattern may occur. The first, a crossover with a 0101 pattern might reconstruct the old patterns (i.e. $*...*0110*...*$) while the other, which is more dangerous, occurs if the 01110 pattern is crossed over with another 0110 pattern with starting point shifted of one position to the left or to the right. In this case a 011110 pattern may be created, and consequently the original schema $S_i$ may be lost. The probability that $y$ is crossed over with a 0101 pattern or with a 0110 pattern is at most

$$P_{\text{badcross}} \leq \frac{1}{\mu}\frac{2h}{n}p_c$$

The probability that a node in the middle one-column of $y$ is removed from the cover by mutation, "securing" the pattern by guaranteeing it will be optimised, is $P_{\text{sec}} = \frac{1}{\mu}\frac{h}{n}$.

The probability that the bad crossover, which may disrupt the original defining schema $S_i$, occurs before the mutation operator removes one node from the middle column is:

$$\frac{\frac{2h}{\mu n}p_c}{\frac{2h}{\mu n}p_c + \frac{h}{\mu n}} = \frac{2p_c}{2p_c+1} \leq 2p_c$$

So the probability that the mutation occurs before any bad crossover is at least $1-2p_c$. Each individual may need at most $l/4$ improving crossovers. Hence the maximum number of improving crossovers that may occur in one run of the algorithm is $\frac{\mu l}{4}$. This gives a probability of $(1-2p_c)^{\frac{\mu l}{4}}$ for the algorithm of performing all the improving crossovers without any schemata $S_i$ being lost through "bad" crossovers.

The expected time for the whole column represented by the middle one-bit to be "emptied" is $O(\mu n \log n)$. Hence the expected time to improve a pattern is

$$O(\mu^2 l(1/p_c) + \mu n \log n)$$

Since each "improving" crossover adjusts 4 bits, the total number of "improving" crossovers needed to reach the global optimum is at most $l/4$. Hence the total expected time is at most

$$O\left(\frac{\mu^2 l}{p_c}\frac{l}{4} + \mu n \log n\right) = O\left(\frac{\mu^2 n}{p_c} + \mu n \log n\right)$$

■

The proof can probably be extended to the ($\mu$+1)-EA with the diversity mechanism and crossover. The main extra challenge would be to guarantee that also with flips of more than one bit all the *schema* patterns are contained in the population once the local search covers are reached.

## VI. Conclusions

A first step in the analysis of population-based EAs for the vertex cover problem has been accomplished.

Concerning offspring populations, it has been proved that the (1+$\lambda$)-EA is a poor worst-case approximation algorithm for vertex cover. This result is obtained by proving that with constant probability only low quality covers can be obtained by the algorithm on a bipartite instance class of graphs in polynomial time. Even though parent populations are generally considered useful for *global* exploration, it is proved that the ($\mu$+1)-EA may also get stuck on local search covers of the bipartite graphs due to premature convergence. This example gives a reason for using diversity mechanisms with population-based EAs for combinatorial optimisation. In fact, it is shown that even a simple diversity mechanism can assure that the algorithm does not get stuck on the bipartite graph landscape by avoiding that the whole population converges to the local optimum. Furthermore, the first proof is given that crossover can considerably improve the performance of EAs on a previously studied "difficult" vertex cover landscape.

As further work we plan to examine in greater depth the worst case approximation ratio of the ($\mu$+1)-EA for covering problems. In particular the analysis presented here gives hints that the algorithm may get stuck on local search covers, however no proof is yet available regarding poor approximation ratios. Another step we will take in the future is that of analysing EAs that simultaneously make use of parent and offspring populations as well as a crossover operator.

## Acknowledgment

This work was supported by an EPSRC grant (EP/C520696/1). The authors are grateful to Dr. Per Kristian Lehre, Dr. Carsten Witt and Dirk Sudholt for insightful discussions.

## References

[1] R. Sarker, M. Mohammadian, and X. Yao, Eds., *Evolutionary Optimization*. Norwell, MA, USA: Kluwer Academic Publishers, 2002.

[2] P. S. Oliveto, J. He, and X. Yao, "Computational complexity analysis of evolutionary algorithms for combinatorial optimization: A decade of results," *International Journal of Automation and Computing*, vol. 4, no. 3, pp. 281–293, 2007.

[3] T. Jansen, K. A. De Jong, and I. Wegener, "On the choice of the offspring population size in evolutionary algorithms," *Evol. Comput.*, vol. 13, no. 4, pp. 413–440, 2005.

[4] C. Witt, "Runtime analysis of the ($\mu$+1) EA on simple pseudo-boolean functions," *Evolutionary Computation*, vol. 14, no. 1, pp. 65–86, 2006.

[5] J. He and X. Yao, "From an individual to a population: an analysis of the first hitting time of population-based evolutionary algorithms." *IEEE Trans. Evolutionary Computation*, vol. 6, no. 5, pp. 495–511, 2002.

[6] F. Neumann and I. Wegener, "Randomized local search, evolutionary algorithms, and the minimum spanning tree problem," *Theoretical Computer Science*, vol. 378, no. 1, pp. 32–40, 2007.

[7] T. Storch, "Finding large cliques in sparse semi-random graphs by simple randomised search heuristics," Fachbereich Informatik, Universitat Dortmund, Tech. Rep. Nr. CI-211/06, June 2006.

[8] J. He and X. Yao, "Drift analysis and average time complexity of evolutionary algorithms," *Artificial Intelligence*, vol. 127, no. 1, pp. 57–85, 2001.

[9] T. Friedrich, J. He, N. Hebbinghaus, F. Neumann, and C. Witt, "Approximating covering problems by randomized search heuristics using multi-objective models," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO2007)*, London, July 2007, pp. 1219–1225.

[10] P. S. Oliveto, J. He, and X. Yao, "Evolutionary algorithms and the vertex cover problem," in *Proceedings of the 2007 Congress on Evolutionary Computation (CEC2007)*, Singapore, September 2007, pp. 1430–1438.

[11] P. Oliveto, J. He, and X. Yao, "Analysis of the (1+1)-ea for finding approximate solutions to vertex cover problems," University of Birmingham, School of Computer Science, Tech. Rep. CSR-07-10, December 2007. [Online]. Available: ftp://ftp.cs.bham.ac.uk/pub/tech-reports/2007/CSR-07-10.

[12] C. Witt, "Worst-case and average-case approximations by simple randomized search heuristics," in *Proc. of the 22nd Annual Symposium on Theoretical Aspects of Computer Science (STACS '05), LNCS 3404*. Springer, 2005, pp. 44–56.

[13] D. E. Goldberg, *Genetic Algorithms for Search, Optimization, and Machine Learning*. Addison-Wesley, 1989.