Evolving Populations of Solved Subgraphs with Crossover and Constraint Repair

Jiwon Lee and Andrew M. Sutton

Algorithmic Evolution Lab
University of Minnesota Duluth
{lee02761,amsutton}@d.umn.edu
ORCID: 0009-0009-2250-3315, 0000-0003-1295-6715

Abstract. We introduce a population-based approach to solving parameterized graph problems for which the goal is to identify a small set of vertices subject to a feasibility criterion. The idea is to evolve a population of individuals where each individual corresponds to an optimal solution to a subgraph of the original problem. The crossover operation then combines both solutions and subgraphs with the hope to generate an optimal solution for a slightly larger graph. In order to correctly combine solutions and subgraphs, we propose a new crossover operator called *generalized allelic crossover* which generalizes uniform crossover by associating a probability at each locus depending on the combined alleles of the parents. We prove for graphs with n vertices and m edges, the approach solves the k-vertex cover problem in expected time $O\big(4^k m + m^4 \log n\big)$ using a simple RLS-style mutation. This bound can be improved to $O\big(4^k m + m^2 nk \log n\big)$ by using standard mutation constrained to the vertices of the graph.

1 Introduction

Many combinatorial problems involving graphs require finding a small set of components (e.g., vertices or edges) subject to some feasibility criterion. Examples include the k-vertex cover problem, where a solution is a set of vertices of size at most k that includes at least one endpoint of every edge in the graph, and the k-edge dominating set problem in which a solution is a set of edges of size at most k such that each edge not in the set is adjacent to at least one edge in the set.

When applying methods from evolutionary computation to solve such problems, a popular approach to handle infeasibility is to add a penalty term into the fitness function to ensure feasible solutions are preferred over infeasible solutions. A somewhat orthogonal approach is to employ some kind of *constraint repair* operation that repairs infeasible solutions that may have been produced by mutation or crossover. Recently, Branson and Sutton [1] introduced a focused jump-and-repair operation that tries to repair infeasible solutions by taking a focused jump in the fitness landscape and subsequently invokes a domain-specific

repair operator to transform offspring rendered infeasible by mutation into feasible individuals.

An open question from [1] is whether populations and crossover might be used together with constraint repair to efficiently solve parameterized problems. We address that question in this paper by developing a new population-based approach in which the population consists of feasible solutions to subgraphs of the original graph. We introduce a generalization of uniform crossover, called generalized allelic crossover, and show that this crossover together with mutation and constraint repair results in fixed-parameter tractable runtime for the k-vertex cover problem. In particular, on graphs with n vertices and m edges, our approach finds a vertex cover of size at most k (if one exists) within $O\big(4^k m + m^4 \log n\big)$ generations using RLS mutation. This bound can be improved to $O\big(4^k m + m^2 nk \log n\big)$ using standard mutation that only affects the segment of the bitstring corresponding to the vertices. Ignoring polynomial factors, these bounds are not as tight as the FPT bounds for (1+1) EA on the same problem [1], but are interesting for a number of reasons. First, the population-based approach can be easily parallelized to incur runtime speedups. Furthermore, even without parallelization we found the approach to be significantly more efficient than the (1+1) EA variant on empirical studies of randomly generated graphs (reported in Section 5).

1.1 Background

Constraint repair is one of the main techniques employed to address constrained optimization by randomized search heuristics [2]. Repair-based crossover operators were initially developed in the context of permutation-based encodings [7, 13, 15]. A repair mechanism based on local search was applied to vertex cover problems and examined empirically by Pelikan, Kalapala and Hartmann [19] for hierarchical Bayesian optimization (hBOA) and the simple genetic algorithm (SGA). The authors showed that these approaches (along with simulated annealing) produced optimal vertex covers on Erdős-Rényi random graphs significantly faster than branch-and-bound.

Finding minimal vertex covers has been an intense subject of research for evolutionary algorithms. Khuri and Bäck [10] investigated handling infeasible solutions by adding a penalty term to the fitness function that strongly discounts candidate solutions that are not vertex covers. They demonstrated that a genetic algorithm significantly outperforms the well-known 2-approximation based on maximal matching on random graphs and structured graphs introduced by Papadimitriou and Steiglitz [18]. This promising empirical performance of evolutionary methods on vertex cover influenced the development of theoretical work on the problem [5, 8, 17].

From a parameterized complexity perspective, the vertex cover problem was the first problem for which a fixed-parameter tractable evolutionary algorithm was developed in the work of Kratsch and Neumann [11]. They proved that Global SEMO using a tailored mutation operator has an expected optimization time bounded by $O(OPT \cdot n^4 + n \cdot 2^{OPT^2+OPT})$ on any graph G where OPT is

the size of a minimum vertex cover of G. When the objective function also uses cost of an optimal fractional cover (obtained by linear programming) the bound is improved to $O(n^2 \log n + OPT \cdot n^2 + 4^{OPT} n)$.

Branson and Sutton [1] recently showed that a focused jump-and-repair operation can probabilistically simulate *iterative compression*, which is an algorithm design technique for efficiently compressing a feasible solution into a slightly smaller solution. They proved that the (1+1) EA employing focused jump-and-repair and using a restarting framework results in a fixed-parameter tractable $O(2^{OPT} n^2 \log n)$ runtime bound on k-vertex cover problems. They also give fixed-parameter tractable bounds for the FeedbackVertexSet and OddCycleTransversal problems.

In this paper, we show how a carefully designed crossover operator can also leverage populations based on subgraphs to probabilistically simulate iterative compression. The effect of crossover on parameterized complexity has also been studied on the closest string problem [22].

2 Preliminaries

We consider undirected graphs $G = (V, E)$ where $V = \{v_1, \ldots, v_n\}$ is a set of n vertices and $E = \{e_1, \ldots, e_m\} \subseteq \binom{V}{2}$ is a family of m 2-element sets of V called edges. Given a vertex set $S_v \subseteq V$ and an edge set $S_E \subseteq E$, we take the intersection to be the set of vertices that appear in both sets (i.e., the edge set is "flattened"): $S_V \cap S_E = S_V \cap \left(\bigcup_{\{u,v\} \in S_E} \{u, v\}\right)$. The set difference $S_V \setminus S_E$ is defined analogously.

We construct subgraphs of $G = (V, E)$ using subsets of E. Specifically, for any edge set $S_E \subseteq E$, the unique subgraph of G corresponding to S_E is the graph $(V \cap S_E, S_E)$. The neighborhood of $v \in V$ is the set $N(v) \coloneqq \{u \in V \mid \{u, v\} \in E\}$.

A *vertex cover* of $G = (V, E)$ is a set $S \subseteq V$ such that for every $e \in E$, $e \cap S \neq \emptyset$. The problem of deciding if a graph has a vertex cover of a given size is NP-complete [9], and thus finding a minimum size vertex cover is NP-hard.

Large instances of NP-hard problems can often be solved in practice because real-world instances usually exhibit some kind of structure that can be leveraged by solvers. In these settings, worst-case complexity as a function of problem size alone is not as useful. Parameterized complexity theory [3, 4] refines classical complexity theory by factoring the running time of an algorithm into more than one parameter of the input. The aim is to extract the hardness of a problem class by isolating the superpolynomial contribution to the running time to a parameter independent of the problem size.

Formally, a parameterized problem is expressed as a language $L \subseteq \Sigma^* \times \mathbb{N}$ for a finite alphabet Σ. A problem L is *fixed-parameter tractable* if $(x, k) \in L$ can be decided in time $g(k) \cdot |x|^{O(1)}$ for some function g that depends only on k. The complexity class of fixed-parameter tractable problems is FPT. An algorithm is a *Monte Carlo FPT algorithm* for a parameterized problem L if it accepts $(x, k) \in L$ with probability at least $1/2$ in time $g(k) \cdot |x|^{O(1)}$ and accepts $x \notin L$ with probability zero.

Parameterized complexity theory is especially relevant to the analysis of evolutionary algorithms in the context of understanding the influence of problem structure on the running time of NP-hard optimization problems [16,22]. Assume $(x, k) \in L$ and let T be the optimization time of a randomized search heuristic (measured, e.g., by the number of calls to the fitness function) until it certifies $(x, k) \in L$. Any randomized search heuristic with a bound $\mathbb{E}[T] \leq g(k) \cdot |x|^{O(1)}$ on L can be transformed into a Monte Carlo FPT algorithm by stopping its execution after $2g(k) \cdot |x|^{O(1)}$ fitness function evaluations. We thus say a randomized search heuristic runs in *randomized FPT time* on a parameterized problem of size n when $\mathbb{E}[T] \leq g(k) \cdot n^{O(1)}$. The parameterized k-vertex cover problem is, given a graph G and an integer k, decide whether there is a vertex cover of size at most k.

2.1 Generalized Allelic Crossover

Let $x, y \in \{0,1\}^n$. With each locus $i \in [n]$, we associate three probabilities $p_i^{(0)}$, $p_i^{(1)}$ and $p_i^{(2)}$. The *generalized allelic crossover* operator (GAC) produces an offspring z from two parents x and y by setting for each $i \in [n]$

$$z_i = \begin{cases} 1 & \text{with probability } p_i^{(x_i+y_i)}, \\ 0 & \text{otherwise.} \end{cases}$$

Standard uniform crossover [23] can be considered a special case of GAC when $p_i^{(0)} = 0$, $p_i^{(1)} = 1/2$, and $p_i^{(2)} = 1$ for all $i \in [n]$. Moreover, GAC can also implement deterministic set operations such as union ($p_i^{(0)} = 0$, $p_i^{(1)} = p_i^{(2)} = 1$) and intersection ($p_i^{(0)} = p_i^{(1)} = 0$, $p_i^{(2)} = 1$). Keeping $p_i^{(0)} = 0$ and $p_i^{(2)} = 1$ but varying $p_i^{(1)}$ recovers the "parameterized uniform crossover" of Spears and De Jong [21] (if $p_i^{(1)} = p_j^{(1)}$ for all $i, j \in [n]$).

We note that GAC is allowed to deviate from common crossover design philosophies. Specifically, when $p_i^{(2)} < 1$ (or $p_i^{(0)} > 0$), it becomes possible for GAC to produce offspring that lie outside the smallest hyperplane containing both parents. Thus, with such settings, GAC is not a *forma-respecting* operator (also called *inheritance-respectful* by Friedrich et al. [6]), nor does it always *strictly transmit* in the sense of Radcliffe [20]. Similarly, it is not necessarily a *geometric* crossover operator in the sense of Moraglio [14], as it can, with certain settings, produce offspring that do not lie in the convex hull described by the parents.

One may argue that such allowances in some sense perverts the original philosophy of crossover, which is meant to share information possessed by all parents. Nevertheless, we show in this paper how it can be leveraged to achieve good results, at least on the vertex cover problem.

3 A Population-based Subgraph GA for k-Vertex Cover

The philosophy of our approach is to start with a large population of feasible solutions to small subgraphs of G, and allow these subgraphs to produce offspring that correspond to feasible solutions of slightly larger subgraphs. This process continues until a feasible solution is found for the entire graph G.

In the context of the k-vertex cover problem, a population of feasible solutions to subgraphs corresponds to a set of subgraphs, each with a valid vertex cover of size at most k. We represent each individual as a bit string of length $n+m$ in which the first n elements encode a candidate vertex cover and the last m elements encode a candidate subgraph. Given a bit string $x \in \{0,1\}^{n+m}$, we define the operators $S(x)$ and $E(x)$ that extract the candidate cover and the candidate edge set, respectively. In particular,

$$S(x) \coloneqq \{v_i \in V : i \in [n] \text{ and } x_i = 1\}$$
$$E(x) \coloneqq \{e_i \in E : i \in [m] \text{ and } x_{n+i} = 1\}$$

Using these operators, we can define the concept of feasibility as follows.

Definition 1. *Let $G = (V, E)$ be a graph. An individual $x \in \{0,1\}^{n+m}$ is* feasible *when the vertex set $S(x) \cap E(x)$ corresponds to a feasible k-vertex cover in the subgraph of G selected by $E(x)$, i.e., when*

1. *for each $e \in E(x)$, $e \cap S(x) \neq \emptyset$, and*
2. *$|S(x) \cap E(x)| \leq k$.*

Throughout the paper, we will assume that we are given a graph G that is guaranteed to have a vertex cover of size k. This is not strictly a limitation, since the runtime bounds provided can be used to design a search for the smallest k with probabilistic guarantees on the success of each run (cf the restart framework in [1]).

In each generation, an offspring is created via crossover or mutation. Survival selection proceeds similar to the Global SEMO algorithm from evolutionary multiobjective optimization: if the offspring is not dominated or tied by any individuals in the current population, it is included in the next population. Moreover, any individual in the current population that is dominated by the offspring does not survive.

Definition 2. *A solution x is* dominated *by a solution y, written as $x \prec y$, if and only if at least one of the following conditions holds.*

1. *x is infeasible but y is feasible,*
2. *$E(x) \subset E(y)$, or*
3. *$E(x) = E(y)$ and $|S(x)| > |S(y)|$.*

If $E(x) = E(y)$ and $|S(x)| = |S(y)|$, then x and y are tied. *If y dominates x or x and y are tied, we write $x \preceq y$. If x and y are not tied and x does not dominate y (and vice versa), then x and y are* incomparable.

Algorithm 1: Population-based subgraph GA

Input: A graph $G = (V, E)$ and a crossover probability p_c

```
1  Initialize P_0;
2  t ← 0;
3  while P_t does not contain an optimal solution do
4      Choose parents x, y ∈ P_t uniformly at random;
5      with probability p_c do
6          z ← Crossover(x,y);
7          z ← ConstraintRepair(z,x,y,G);
8      else
9          z ← Copy(x);
10         z ← Mutate(z);
11     if ∄w ∈ P_t s.t. z ⪯ w then
12         P_{t+1} = {w ∈ P_t | w ⋠ z} ∪ {z};
13     t ← t + 1;
```

With probability p_c, in line 6 an offspring is created from two parents by applying a crossover operator, and in line 7, the constraint-repair operation is called that attempts to repair an offspring that was possibly made infeasible by crossover. Otherwise, with probability $1 - p_c$, in line 9, no crossover occurs and an arbitrary parent is copied to the offspring and is varied according to a mutation operator in line 10. Finally, in line 12, the offspring is added to the population if it is not dominated or tied by any element of the population, and all elements of the population dominated by the offspring are removed.

3.1 Variation Operators and Controlling Population Growth

The crossover operation in Algorithm 1 is implemented as generalized allelic crossover (defined in Section 2.1) on bitstrings of length $n+m$ with the following probabilities. For all $1 \leq i \leq n+m$, we set $p_i^{(0)} = 0$. Otherwise,

$$p_i^{(1)} = p_i^{(2)} = \begin{cases} 1/2 & \text{for } 1 \leq i \leq n; \\ 1 & \text{for } n+1 \leq i \leq n+m. \end{cases}$$

We may thus think of bitstrings as separated into a length-n *vertex segment* consisting of the first n elements, and a length-m *edge segment* consisting of elements $n+1$ to $n+m$. The crossover probabilities are designed so that vertex segments are combined probabilistically and edge segments are combined deterministically. In particular, given the offspring z of two parents x and y, it always holds that $S(z)$ is a uniform random subset of $S(x) \cup S(y)$ and $E(z) = E(x) \cup E(y)$.

We consider two separate approaches to mutation. For RLS-Mutation, an index i is chosen uniformly at random in $\{1, \ldots, n+m\}$ and the single bit x_i is flipped. For Vertex-Mutation, we flip each bit in x_i with probability $1/n$, but only for indexes in $\{1, \ldots, n\}$ that correspond to the vertices of G.

We do not consider standard bit mutation on the entire bit string for the following reasons. We will require that crossover and mutation always create offspring that dominates its parent(s), or is always dominated by a parent. Enforcing this constraint ensures that the population size cannot increase during the execution of the algorithm (captured in Lemma 1 below). The Vertex-Mutation operator clearly creates an offspring with a subgraph equal to the one of its parent. The RLS-Mutation operator may add or delete exactly one edge (or none), in which case one subgraph contains the other. Using standard bit mutation on the entire bitstring allows for the chance to create an offspring with an edge set that is incomparable with respect to subset inclusion to the edge set of its parent. This offspring would also be incomparable to its parent in the sense of Definition 2. Allowing such incomparable offspring could cause uncontrolled population growth during the execution of the algorithm. While there may be specific techniques to mitigate this growth, in this paper we will restrict ourselves to the above defined mutation operators that guarantee a population size that does not grow.

Lemma 1. *Consider the execution of Algorithm 1 using either* RLS-Mutation *or* Vertex-Mutation *for its mutation operation. Let P_t denote the population in generation t. If all individuals in P_t are feasible, it holds that (1) all individuals in P_{t+1} are feasible, and (2) $|P_{t+1}| \leq |P_t|$.*

Proof. In line 12 of Algorithm 1, an offspring z is only added to P_{t+1} if it is not dominated nor tied by any individual P_t. The first condition trivially holds, as all infeasible solutions are automatically dominated by feasible solutions.

Let $x, y \in P_t$ be the parents selected in line 4 of Algorithm 1. We argue that either $x \prec z$, $z \prec x$, or x and z are tied. If z is not feasible, then the $z \prec x$ clearly holds, since x is feasible. Thus, assume that z is also feasible. It suffices to show that either $E(x) \subseteq E(z)$ or $E(z) \subseteq E(x)$. If one graph is a proper subset of the other, then we have dominance (of the superset graph). Otherwise, when $E(x) = E(z)$, then x and z are either tied (if $|S(x)| = |S(z)|$) or one dominates the other.

If z was created by crossover, then since $p_i^{(1)} = p_i^{(2)} = 1$ for all indexes $i \in \{n+1, \ldots, n+m\}$, it follows that $E(z) = E(x) \cup E(y)$, and therefore $E(x) \subseteq E(z)$. If z was created by RLS-Mutation, then either $E(x) = E(z)$ (when the flipped bit index is at most n), otherwise $E(z)$ has gained (respectively, lost) exactly one edge compared to $E(x)$. This means that $E(z) \subset E(x)$ (respectively, $E(x) \subset E(z)$). Finally, since Vertex-Mutation does not affect the indexes corresponding to the edges, we would have $E(x) = E(z)$.

Since z is added to the population only if it is not dominated or tied, it follows that $z \in P_{t+1} \iff x \notin P_{t+1}$ and thus we have $|P_{t+1}| \leq |P_t|$. □

3.2 Constraint Repair Operator

Crossover can produce infeasible solutions. We employ a constraint repair operation inspired by the iterative compression procedure that attempts to repair any crossover offspring that does not correspond to a vertex cover.

In particular, if crossover removes a vertex from $S(x) \cup S(y)$, then any uncovered edge can be repaired by adding the neighbors of that vertex back into the offspring's vertex set. The resulting offspring is guaranteed to be a vertex cover. However, it is not necessarily feasible in the sense of the k-vertex cover problem, as it may return a cover of size greater than k. We formalize the vertex cover repair operator in Algorithm 2.

Algorithm 2: ConstraintRepairVC (z, x, y, G)

Input: An offspring z, parents x, y and a graph $G = (V, E)$

```
if S(z) is a feasible cover for E(z) then return z;
// Store vertices that were removed from S(x) ∪ S(y)
A ← (S(x) ∪ S(y)) \ S(z);
foreach v ∈ A do
    S(z) ← S(z) ∪ N(v);
return z;
```

The intuition behind this procedure is that since $S(x)$ is a feasible cover for $E(x)$ and $S(y)$ is a feasible cover for $E(y)$, if v belongs to one of them, but not $S(z)$, then the neighbors of v in G may need to be added to potentially cover the edges in $E(z) = E(x) \cup E(y)$.

4 Runtime Analysis

Given an individual $x \in \{0,1\}^{n+m}$, since $S(x)$ is interpreted to be a candidate solution in the subgraph defined by $E(x)$, any $v \in S(x) \backslash E(x)$ does not contribute to the solution. This motivates the following definition.

Definition 3. *We say an individual $x \in \{0,1\}^{n+m}$ is* efficient *when $S(x) \setminus E(x) = \emptyset$. We call a population efficient when all of its individuals are efficient.*

We are interested in bounding the total expected number of generations that Algorithm 1 spends on populations that are not efficient. The mutation operator together with the fitness domination criteria listed above ensures that there is always selective pressure toward efficient populations, but in some cases the "inefficiency" of a population can increase. However, we show in the following theorem that efficiency is lost only in cases where we are in some sense making overall progress, and thus the number of times this occurs can be bounded.

Theorem 1. *Let $G = (V, E)$ be a graph. We assume that G is connected. Let P_0 be any set of feasible solutions with $|P_0| = \text{poly}(n)$ and $\bigcup_{x \in P_0} E(x) = E$. Let $p_c \in (0,1)$ be a constant. The expected number of generations of Algorithm 1 in which the population is not efficient is bounded above by $O\big(|P_0|^2 m^2 \log n\big)$ using* RLS-Mutation *and $O\big(|P_0|^2 kn \log n\big)$ using* Vertex-Mutation.

Proof. We design a nonnegative drift potential φ over populations as follows.

$$\varphi(P_t) = \sum_{x \in P_t} |S(x) \setminus E(x)|$$

. Clearly, $\varphi(P_t) = 0$ if and only if P_t is efficient.

This drift function can fluctuate during the course of the execution of Algorithm 1. However, we will later show that the number of times it can increase is strictly bounded. Thus we are able to bound the total time (in expectation) that the algorithm spends waiting for the potential to decrease to zero. Since the remainder of the time the potential would be at zero, the population must be efficient during those generations.

Let $A = \{\varphi(P_{t+1}) > \varphi(P_t)\}$ be the event that the offspring created from population P_t survives into P_{t+1} and results in a strict increase in potential. We first bound the conditional drift of φ on the complementary event $\overline{A}$, namely, the potential of P_{t+1} is not strictly greater than the potential of P_t. A sufficient event to *decrease* the potential is to (1) perform mutation on a single parent with probability $1 - p_c$, (2) select $x \in P_t$ with probability $1/|P_t|$, and (3) flip exactly one bit in $S(x) \setminus E(x)$. Summing the probability of these disjoint events over all possible individuals $x \in P_t$, for RLS-Mutation the conditional drift can be bounded as

$$\mathbb{E}\left[\varphi(P_t) - \varphi(P_{t+1}) \mid P_t, \overline{A}\right] \geq \sum_{x \in P_t} \frac{(1-p_c)|S(x) \setminus E(x)|}{|P_t|(n+m)} \geq \frac{(1-p_c)}{|P_0|(n+m)} \cdot \varphi(P_t)$$

since $|P_t| \leq |P_0|$, by Lemma 1. Similarly, for Vertex-Mutation, the probability of flipping exactly one particular bit is $(1/n)(1-1/n)^{n-1} \geq 1/e$, so we have

$$\mathbb{E}\left[\varphi(P_t) - \varphi(P_{t+1}) \mid P_t, \overline{A}\right] \geq \frac{(1-p_c)}{|P_0|en} \cdot \varphi(P_t).$$

The expected time until φ hits zero (or increases, if sooner) can be bounded above using multiplicative drift [12] as $O(|P_0|(n+m)\log(|P_0|n)) = O(|P_0|m \log n)$ for RLS-Mutation and $O(|P_0|n \log n)$ for Vertex-Mutation.

We now argue that the total number of times that this potential can increase is strictly bounded for the entire run. We consider two further events in the offspring creation process. Let B denote the event that z is created by mutation from parent $x \in P_t$ and survives into P_{t+1}, necessarily replacing x in the population. Let C denote the event that z is created by a successful crossover between parents x and y, again necessarily replacing x and y in the population.

Note that event C must reduce the population size by at least one because a new feasible subgraph is created from two parents, and those two parents would be dominated by the offspring. Thus the event $A \cap C$ can happen at most $|P_0| - 1$ times during the entire run.

Under RLS-Mutation, a necessary condition for the event $A \cap B$ is that the mutation occurs in the edge segment of the bitstring (indexes larger than n) and particularly, when a bit is flipped from zero to one. If the mutation producing

z had occurred in the vertex segment (indexes at most n) then the potential cannot increase, as this would imply $S(z) > S(x)$ and $E(z) = E(x)$, thereby z would be dominated by x. Similarly, if the mutation occurs in the edge segment and changes a one to a zero, then $E(z) \subset E(x)$ and again z would not survive to be included in P_{t+1}. Since edges can be added to a subgraph in the population at most $|P_0|m$ times, the event $A \cap B$ occurs at most $|P_0|m$ times during the entire run.

Under Vertex-Mutation, event $A \cap B$ only occurs when x is replaced by offspring z where $E(x) = E(z)$, and to compensate for the fact $|S(z) \setminus E(z)| > |S(x) \setminus E(x)|$, it must be true that $|S(z) \cap E(z)| < |S(x) \cap E(x)|$. However, both x and z are necessarily feasible, so $|S(x) \cap E(x)| \leq k$ and thus for any given subgraph in the population, event $A \cap B$ can occur at most k times. Since $B \cup C$ is necessary for A, it follows that $A \cap B$ and $A \cap C$ partition A, and thus the potential can only increase during these events.

Therefore, in the case of RLS-Mutation, the potential can reset at most $|P_0|(m+1)-1$ times, and for Vertex-Mutation, the potential can reset at most $|P_0|(k+1)-1$ times. The claimed bounds thus follow from the multiplicative drift arguments above, and by pessimistically assuming the potential always resets to the highest possible value and all possible resets occur. □

To bound the runtime of Algorithm 1, it remains only to estimate the total time spent on efficient populations. The proof of the following theorem establishes this bound by determining the probability that crossover successfully produces a dominating offspring from any parents in an efficient population. Such an event strictly reduces the population size by combining two solved subgraphs into a larger solved subgraph. The total waiting time for these events together with the time spent on inefficient populations yields the claimed bounds.

Theorem 2. *Let $G = (V, E)$ be a connected graph and let P_0 be any polynomial-size set of feasible individuals such that $E = \bigcup_{x \in P_0} E(x)$.*

Setting $p_c \in (0, 1)$ to be a constant, Algorithm 1 finds a k vertex cover of G (if one exists) in $O\big(4^k|P_0| + |P_0|^2 m^2 \log n\big)$ generations using RLS-Mutation *and in $O\big(4^k|P_0| + |P_0|^2 nk \log n\big)$ generations using* Vertex-Mutation.

Proof. By Theorem 1, the expected number of generations the algorithm spends on populations that are not efficient is at most $O\big(|P_0|^2 m^2 \log n\big)$ using RLS-Mutation and at most $O\big(|P_0|^2 kn \log n\big)$ using Vertex-Mutation.

We thus seek to bound the number of generations spent on efficient populations until an optimal solution is found. Suppose that P_t is efficient and let $x, y \in P_t$ with $E(x) \neq E(y)$. Since both x and y must be feasible, $S(x)$ is a cover of the subgraph $E(x)$ and $S(y)$ is a cover of the subgraph $E(y)$. Moreover, the feasibility of x and y together with the efficiency of P_t guarantees that $|S(x)|, |S(y)| \leq k$. Thus $S(x) \cup S(y)$ is a valid cover of the subgraph $E(x) \cup E(y)$ with $|S(x) \cup S(y)| \leq 2k$. We have also assumed there is a k-cover of the entire graph G, namely $S^* \subseteq V$ where $|S^*| \leq k$. Then $S^* \cap (E(x) \cup E(y))$ is also a cover of the subgraph $E(x) \cup E(y)$. Let $R = (S(x) \cup S(y)) \setminus S^*$ denote the

set of vertices that belong to $S(x) \cup S(y)$, but not to the optimal cover. Let $T = (S(x) \cup S(y)) \cap S^*$ be the set of vertices that belong to both covers.

We consider the application of generalized allelic crossover using x and y as parents to produce an offspring z. Note that $E(z) = E(x) \cup E(y)$ since $p_i^{(1)} = p_i^{(2)} = 1$ for all $n+1 \leq i \leq n+m$ in the edge segment of the bitstring. Similarly, for all $1 \leq i \leq n$ in the vertex segment of the bitstring, we have $p_i^{(1)} = p_i^{(2)} = 1/2$, so every vertex $v \in S(x) \cup S(y)$ belongs to to $S(z)$ with probability $1/2$. Since $p_i^{(0)} = 0$ for all i, any vertex (respectively, edge) not in $S(x) \cup S(y)$ (respectively, $E(x) \cup E(y)$) will not belong to $S(z)$ (respectively, $E(z)$).

Note that since $T \subseteq (S(x) \cup S(y))$, we have $S(z) = T$ with probability exactly $2^{-|S(x) \cup S(y)|} \geq 2^{-2k}$. We condition on this event for the remainder of the proof. Every edge in the subgraph $E(x) \cup E(y) = E(z)$ that is not covered by $S(z)$ must have one endpoint in S^* and one endpoint in R because both S^* and $(S(x) \cup S(y))$ are valid vertex covers of $E(x) \cup E(y)$.

After crossover, the repair operation listed in Algorithm 2 identifies the set of vertices removed from $S(x) \cup S(y)$, which in this csea corresponds exactly to the set R, and then add the neighbor set $N(R)$. This results in a repaired offspring z' with $S(z') = T \cup N(R) \subseteq S^*$ which must cover $E(x) \cup E(y) = E(z')$.

The fact $S(z') \subseteq S^*$ implies $|S(z')| \leq k$ and so it follows that $x, y \prec z'$, and since z' would not be dominated by any other element of the population, z' replaces x and y in P_{t+1}. This event, which occurs with probability at least 4^{-k}, results in a strictly smaller population $|P_{t+1}| < |P_t|$.

A feasible, efficient population containing more than one individual can always shrink with probability $\Omega(4^{-k})$ under the above sequence of events. It follows that the waiting time until an efficient population shrinks in this way is bounded above by $O(4^k)$. The population can shrink at most $|P_0| - 1$ times before it consists of a single feasible individual x^*. As GAC always composes subgraphs by union, it holds that $E(x^*) = E$, and since feasibility is maintained $S(x^*)$ is a vertex cover of size at most k for G.

Before generating x^*, the algorithm can spend at most $O(4^k|P_0|)$ generations on efficient populations and the total time spent on inefficient populations is bounded by Theorem 1, which yields the claimed result. □

Theorem 2 requires only an initial population of feasible subgraphs that compose into G. For specific graphs, this could be constructed by including promising subgraphs that are hoped to be "close" to an optimal cover. However, every graph at least has a natural initial population of size m in which each subgraph consists of a single unique edge from G (together with a cover that contains at least one vertex incident on that edge). This yields the following general bound.

Corollary 1. *Let $G = (V, E)$ be a graph on n vertices and m edges. Algorithm 1 finds a vertex cover of size at most k of G (if one exists) in $O\left(4^k m + m^4 \log n\right)$ generations using* RLS-Mutation *and $O\left(4^k m + m^2 nk \log n\right)$ generations using* Vertex-Mutation.

Proof. Construct P_0 from G as follows. For each edge $\{u, v\} \in E$, let x be any string in $\{0,1\}^{n+m}$ such that $x_u = 1$ and $E(x) = \{\{uv\}\}$. Then P_0 satisfies the conditions for Theorem 2. □

5 Experiments

To investigate the concrete running time of Algorithm 1, and to compare it with the similar repair-based (1+1) $\text{EA}_k^{\text{J+R}}$ introduced in [1], we performed a number of experiments on the planted vertex cover instances from [1]. Each of these instances were generated by randomly selecting a subset of k vertices and including an edge with probability p subject to having at least one end point in the subset.

The number of vertices n varies from 20 to 100 by 10, planted cover size k varies from 3 to 10, and edge probabilities are $p \in \{\frac{1}{10}, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}\}$. On each graph we ran each algorithm for 50 trials and measured the run time as the number of calls to the fitness function until a vertex cover of size at most k is found. For Algorithm 1, we set $p_c = 0.8$, and experimented with both RLS-Mutation and Vertex-Mutation. The median run times for $k = 10$ as a function of n are reported in Figure 1. We omit results for other k-values due to space constraints, but mention that the trend is identical. Note that after generating a random graph for a given n, we remove isolated vertices and the figures report the true n after removal. This explains the variability in n at low edge densities. In Figure 2, we plot the median run times as a function of k fixing $n = 100$. The bottom right plot shows the median as a function of k taken over all p and n. We also provide box plots of the running time of all three algorithms on graphs with $n = 100, k = 10$ over all edge densities in Figure 3.

Despite the fact that runtime bound of the (1+1) $\text{EA}_k^{\text{J+R}}$ from [1] is exponentially smaller in k than the one derived in Corollary 1, we see that Algorithm 1 with Vertex-Mutation scales better with both n and k on this class of graphs, and has smaller variability as measured by interquartile range. Not surprisingly, the variant using RLS-Mutation is strongly affected by the number of edges, and performs poorly on denser graphs, as can be seen in Figure 1.

Khuri and Bäck [10] conducted experiments on hard vertex cover instances using a GA with two-point crossover and proportional selection. In addition to (nonplanted) random graphs, they investigated two structured graph instances originally defined by Papadimitriou and Steiglitz [18] to demonstrate that greedy degree-heuristics fail to approximate minimum vertex covers. These instances (PS100 and PS202) have vertex counts $n = 100, 202$, edge counts $m = 1122, 4556$, and minimum vertex covers of size $k = 34, 68$. We also report the success rates of the population subgraph algorithm on these instances for different runtime budgets in Table 1 along with the success rates reported in [10]. Note that the minimum covers for these graphs are comparatively large. Nevertheless, we still observe surprisingly high success rates, even at runtime budgets much smaller than 4^k.

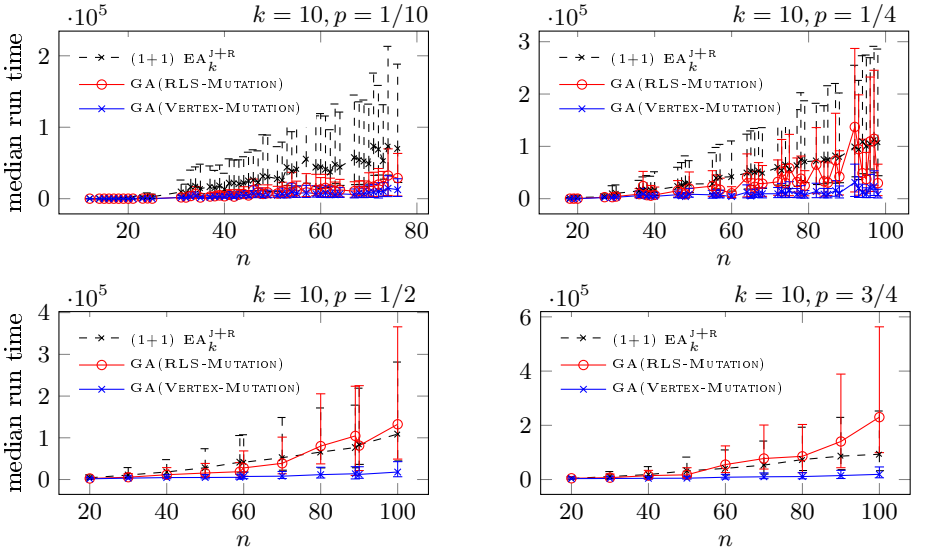

Fig. 1: Median run times of Algorithm 1 and the (1+1) EA$_k^{J+R}$ on random planted $k = 10$ vertex cover instances of varying edge density p as a function of n. Error bars denote interquartile range.

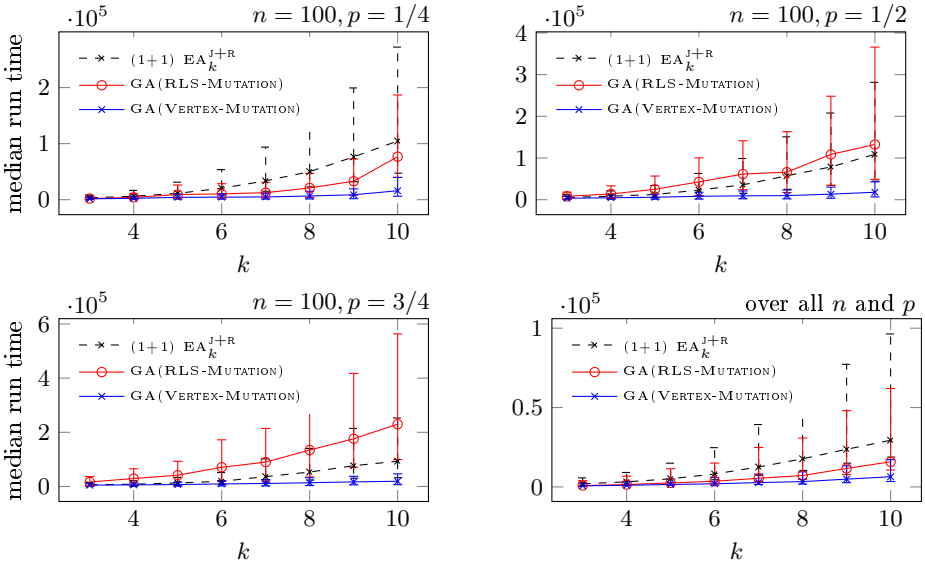

Fig. 2: Median run times of Algorithm 1 and the (1+1) EA$_k^{J+R}$ on random planted vertex cover instances of varying edge density p as a function of k. Error bars denote interquartile range.

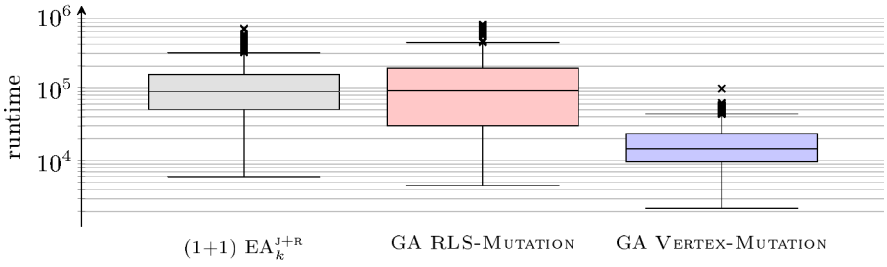

Fig. 3: Runtime statistics for $n = 100$ and $k = 10$ over all edge densities.

	Khuri & Bäck GA [10]		GA RLS-Mutation		GA Vertex-Mutation	
budget	PS100	PS202	PS100	PS202	PS100	PS202
$2 \cdot 10^4$	65%	—	18%	0%	97%	0%
$4 \cdot 10^4$	—	60%	63%	0%	98%	29%
10^6	—	—	100%	96%	100%	100%

Table 1: Success rates on Papadimitriou-Steiglitz instances PS100 and PS202 from [10] for different runtime budgets.

6 Conclusion

We have introduced a population-based technique designed to solve feasible component-selection problems in graphs. In this technique, one begins with a large population of solutions to small subgraphs, e.g., single edges or vertices. We showed that if a suitable constraint repair operation is used, the approach can achieve fixed-parameter tractable running time bounds on the NP-hard k vertex cover problem. Our results give insight into how crossover can be leveraged to exploit structure in hard combinatorial optimization problems. Moreover, experimental results suggest that the population-based approach can be more efficient than the (1+1) EA on certain classes of graphs.

There are a number of potential directions for future work. As yet, no lower bounds exist for FPT evolutionary algorithms on the k-vertex cover problem. This is rather difficult, as the structure of different kinds of graphs have varying and unpredictable degrees of influence on the assorted modules of evolutionary algorithms. Nevertheless, it would be interesting to obtain lower bounds in terms of k for certain graph categories. Moreover, the proposed subgraph approach leverages only a suitable constraint repair operations, so it could be easily extended to similar problems in which a small set of vertices or edges need to be selected subject to some feasibility criterion.

Acknowledgments. This research was funded by NSF grant 2144080.

Disclosure of Interests. The authors have no competing interests.

References

1. Branson, L., Sutton, A.M.: Focused jump-and-repair constraint handling for fixed-parameter tractable graph problems. In: Proceedings of the Sixteenth ACM/SIGEVO Conference on Foundations of Genetic Algorithms. Association for Computing Machinery, New York, NY, USA (2021), https://doi.org/10.1145/3450218.3477304
2. Coello Coello, C.A.: Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. Computer Methods in Applied Mechanics and Engineering **191**(11-12), 1245–1287 (Jan 2002). https://doi.org/10.1016/S0045-7825(01)00323-1
3. Downey, R.G., Fellows, M.R.: Parameterized Complexiy. Springer (1999)
4. Flum, J., Grohe, M.: Parameterized complexity theory. Springer-Verlag (2006)
5. Friedrich, T., Hebbinghaus, N., Neumann, F., He, J., Witt, C.: Approximating covering problems by randomized search heuristics using multi-objective models. In: Proceedings of the Conference on Genetic and Evolutionary Computation (GECCO), London, UK. pp. 797–804. ACM (2007). https://doi.org/10.1145/1276958.1277118
6. Friedrich, T., Kötzing, T., Radhakrishnan, A., Schiller, L., Schirneck, M., Tennigkeit, G., Wietheger, S.: Crossover for cardinality constrained optimization. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO). ACM (Jul 2022). https://doi.org/10.1145/3512290.3528713
7. Goldberg, D.E., Jr., R.L.: Alleles, loci, and the traveling salesman problem. In: Grefenstette, J.J. (ed.) Proceedings of the First International Conference on Genetic Algorithms and their Applications (ICGA), Pittsburgh, PA, USA. vol. 154, pp. 154–159. Lawrence Erlbaum, Hillsdale, NJ (1985)
8. Jansen, T., Oliveto, P.S., Zarges, C.: Approximating vertex cover using edge-based representations. In: Neumann, F., Jong, K.A.D. (eds.) Proceedings of the Twelfth Workshop on Foundations of Genetic Algorithms (FOGA XII), Adelaide, SA, Australia, January 16-20, 2013. pp. 87–96. ACM (2013). https://doi.org/10.1145/2460239.2460248
9. Karp, R.M.: Reducibility among combinatorial problems. In: Miller, R.E., Thatcher, J.W. (eds.) Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, USA. pp. 85–103. The IBM Research Symposia Series, Plenum Press, New York (1972). https://doi.org/10.1007/978-1-4684-2001-2_9
10. Khuri, S., Bäck, T.: An evolutionary heuristic for the minimum vertex cover problem. In: Kunze, J., Stoyan, H. (eds.) Workshops of the Eighteenth Annual German Conference on Artificial Intelligence (KI-94), Saarbrücken, Germany. pp. 86–90 (1994)
11. Kratsch, S., Neumann, F.: Fixed-parameter evolutionary algorithms and the vertex cover problem. Algorithmica **65**(4), 754–771 (May 2012). https://doi.org/10.1007/s00453-012-9660-4
12. Lengler, J.: Drift analysis. In: Doerr, B., Neumann, F. (eds.) Theory of Evolutionary Computation - Recent Developments in Discrete Optimization, pp. 89–131. Natural Computing Series, Springer (2020). https://doi.org/10.1007/978-3-030-29414-4_2
13. Mitchell, G.G., O'Donoghue, D., Barnes, D., McCarville, M.: Generepair - a repair operator for genetic algorithms. In: Late-Breaking Papers at the Genetic and

Evolutionary Computation Conference (GECCO), Chicago, IL, USA. pp. 235–239 (2003), http://mural.maynoothuniversity.ie/10351/
14. Moraglio, A.: Abstract convex evolutionary search. In: Proceedings of the Eleventh Workshop on Foundations of Genetic Algorithms (FOGA XI). ACM (Jan 2011). https://doi.org/10.1145/1967654.1967668
15. Mühlenbein, H.: Parallel genetic algorithms in combinatorial optimization. In: Balci, O., Sharda, R., Zenios, S.A. (eds.) Computer Science and Operations Research: New Developments in their Interfaces, pp. 441–453. Pergamon Press, Amsterdam (1992). https://doi.org/10.1016/b978-0-08-040806-4.50034-4
16. Neumann, F., Sutton, A.M.: Parameterized complexity analysis of randomized search heuristics. In: Doerr, B., Neumann, F. (eds.) Theory of Evolutionary Computation: Recent Developments in Discrete Optimization, pp. 213–248. Natural Computing Series, Springer International Publishing (2020). https://doi.org/10.1007/978-3-030-29414-4_4
17. Oliveto, P.S., He, J., Yao, X.: Analysis of the (1+1)-EA for finding approximate solutions to vertex cover problems. IEEE Transactions on Evolutionary Computation **13**(5), 1006–1029 (2009). https://doi.org/10.1109/tevc.2009.2014362
18. Papadimitriou, C.H., Steiglitz, K.: Combinatorial Optimization: Algorithms and Complexity. Prentice-Hall (1982)
19. Pelikan, M., Kalapala, R., Hartmann, A.K.: Hybrid evolutionary algorithms on minimum vertex cover for random graphs. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO), London, UK. pp. 547–554. ACM (2007). https://doi.org/10.1145/1276958.1277073
20. Radcliffe, N.J.: Forma analysis and random respectful recombination. In: Proceedings of the Fourth International Conference on Genetic Algorithms (ICGA) (1991)
21. Spears, W.M., Jong, K.A.D.: On the virtues of parameterized crossover. In: Proceedings of the Fourth International Conference on Genetic Algorithms (ICGA). pp. 230–236 (1991)
22. Sutton, A.M.: Fixed-parameter tractability of crossover: Steady-state GAs on the closest string problem. Algorithmica **83**(4), 1138–1163 (2021). https://doi.org/10.1007/s00453-021-00809-8
23. Syswerda, G.: Uniform crossover in genetic algorithms. In: Proceedings of the Third International Conference on Genetic Algorithms (ICGA). vol. 3, pp. 2–9 (1989)