A Fixed-Parameter Tractable GA for Data Clustering

Liam Gaeuman
University of Minnesota Duluth
Duluth, Minnesota, USA

Andrew M. Sutton
University of Minnesota Duluth
Duluth, Minnesota, USA

Abstract

Clustering is the process of grouping similar data points into distinct clusters. Graph theoretic techniques for data clustering attempt to find the most efficient way to convert a precomputed similarity graph into a cluster graph, which is a collection of disjoint cliques. In contrast, most existing evolutionary approaches to clustering are based on partitioning data points according to feature vectors. In this paper we present an evolutionary algorithm for tackling data clustering from the graph theoretic perspective. In particular, we adapt a genetic algorithm (the SubPopGA) that maintains a population of solved subgraphs to solve the NP-hard k-Cluster Deletion and k-Cluster Vertex Deletion problems. This adaptation is nontrivial, as it requires modifying the technique to work on induced subgraphs and designing a novel template parent mechanism for uniform crossover. We prove that the resulting SubPopGA has a fixed-parameter tractable running time on both problems. Given an arbitrary graph on n vertices and m edges, we prove that it solves k-Cluster Deletion in $O(n3^{4k} + mn^2 \log n)$ generations in expectation, and the k-Cluster Vertex Deletion problem in $O(n3^{5k} + n^3 \log n)$ in expectation. We also present results from a number of computational experiments that measure the running time of the SubPopGA on real-world biological data sets coming from protein-protein interaction networks.

CCS Concepts

• **Theory of computation → Theory of randomized search heuristics**.

Keywords

runtime analysis, fixed-parameter tractability, clustering

ACM Reference Format:
Liam Gaeuman and Andrew M. Sutton. 2025. A Fixed-Parameter Tractable GA for Data Clustering. In *Foundations of Genetic Algorithms XVIII (FOGA '25), August 27–29, 2025, Leiden, Netherlands.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3729878.3746632

1 Introduction

Data clustering is a technique for partitioning a data set into well-separated clusters that group data points by similarity. Organizing a data set in this way allows for the identification of patterns and relationships, even within large and complex data sets. Clustering is thus a fundamental problem in a wide variety of scientific applications.

There is a multitude of data clustering techniques, including hierarchical models, self-organizing maps, partition-based models and graph theoretic models. In this paper we consider a basic graph theoretic approach [27] in which one constructs a similarity graph whose vertices are data points, and an edge exists between two vertices if their similarity exceeds some predefined threshold. A perfect clustering would correspond to a similarity graph in which each connected component is a clique. Such a graph is called a *cluster graph.* However, in practice, due to the presence of noisy data, incorrectly set parameters or other empirical factors, the resulting similarity graph is not a cluster graph, but in some sense *close* to a cluster graph. This gives rise to so-called cluster editing problems in which one is interested in the smallest number of modifications necessary to transform a similarity graph into a cluster graph.

Two important cluster editing problems are Cluster Deletion, in which one seeks the smallest edge set to delete from a similarity graph to obtain a cluster graph, and Cluster Vertex Deletion, in which the graph modifications are vertex deletions. The parameterized versions of these problems are the k-Cluster Deletion and k-Cluster Vertex Deletion problems where one seeks a deletion set of size at most k. Both problems are NP-complete

In this paper, we introduce a genetic algorithm (GA) for both k-Cluster Deletion and k-Cluster Vertex Deletion based on the subgraph population GA of Lee and Sutton for the k-vertex cover problem [18]. However, in order to adapt their approach to clustering problems, we must resolve two technical issues. First, cluster deletion sets, unlike vertex covers, are not generally closed under union, which poses a challenge to the tractability of their approach. Second, in order to apply their technique, it is necessary to design efficient constraint repair operations. We solve both of these problems to adapt the subgraph population GA (SubPopGA) to the k-Cluster Deletion problem and the k-Cluster Vertex Deletion problem.

To solve the first problem, we introduce the idea of a *template parent* that is probabilistically constructed from the instance graph. Recombination then produces an offspring with uniform crossover between two individuals in the population and the third template parent. We solve the second problem by introducing two efficient repair methods for k-Cluster Deletion and k-Cluster Vertex Deletion. For the latter, we introduce an approach based on the efficient solution to a linear program.

We prove that the SubPopGA solves k-Cluster Deletion and k-Cluster Vertex Deletion in fixed-parameter tractable time. Specifically, we prove a $O(n3^{4k} + mn^2 \log n)$ bound for k-Cluster Deletion and a $O(n3^{5k} + n^3 \log n)$ bound for k-Cluster Vertex Deletion. To our knowledge, this is the first runtime result for any evolutionary algorithm on these problems. To investigate the tightness of our bounds, we conduct a number of experiments on real-world instances from biological data sets.

The remainder of this paper is organized as follows. In the next section, we provide some background and contextualize our approach. In Section 2 we detail mathematical preliminaries and introduce the problems. In Section 3 we introduce the SubPopGA and describe the modifications needed to adapt it to graph clustering problems. In Section 4 we provide FPT runtime bounds for the SubPopGA, followed by an experimental analysis in Section 5. Section 6 concludes the paper.

1.1 Background

Most genetic algorithms (GAs) applied to clustering typically use partition-based approaches that assign data points to clusters based on feature vectors (for a recent survey, see [25]). A few GAs work indirectly with similarity graphs. Tseng and Yang [32] introduced the CLUSTERING GA which computes a similarity graph based on average distances, and then runs a GA that merges the connected components of the similarity graph based on Euclidean distance. Casillas et al. [4] introduced a GA for document clustering that considers the raw distance graph on which they first compute a minimum spanning tree. A GA then evolves a partition of the spanning tree by deleting edges in order to maximize a distance criterion within and between partitions.

Data clustering is important to a large number of applications such as wireless sensor networks [21], protein sequence alignment [23], clustering gene expression data [28], document clustering [13] and data compression. Large-scale biological data can often be analyzed by translating similarity data into graphs, which can in turn provide insight into the structure and classification of functions (see Figure 1 for an example of such a graph).

Cluster Deletion arises naturally in DNA clone classification in which one partitions data produced by hybridization experiments into disjoint clusters to minimize the number of similar data points that are in different clusters [7, 11]. Cluster Vertex Deletion is the related problem of deleting potentially contaminated points in a data set to produce a pure cluster graph and can also be seen as the problem of making a symmetric relation transitive by omitting a minimum number of elements [16].

The Cluster Deletion problem is NP-hard to even approximate within a $(1+\epsilon)$ factor [27]. The best known FPT complexity of k-Cluster Deletion is $O(1.404^k n^{O(1)})$ due to Tsur [33]. Cluster Vertex Deletion is NP-complete [19] and its optimization variant is MaxSNP-Hard [20]. The best known FPT complexity of k-Cluster Vertex Deletion is $O(1.7549^k n^{O(1)})$ due to Tian, Xiao and Yang [31].

2 Preliminaries

We consider simple undirected graphs $G = (V, E)$ where V is a set of n vertices and E is a set of m edges. Given a vertex set $S \subseteq V$, we denote as $G[S]$ the subgraph of G induced by S. Given a vertex $v \in V$, we denote the neighborhood of v as $N(v) := \{u \in V : \{u, v\} \in E\}$. For any set of edges $F \subseteq E$, we denote the subgraph of G obtained from deleting the edges in F as $G - F$. We will often denote the vertex set (respectively, edge set) of a graph G as $V(G)$ (respectively, $E(G)$).

A *cluster graph* is a collection of vertex-disjoint cliques. Given an undirected graph $G = (V, E)$ a *cluster deletion* set (CD set) of G is a set $F \subseteq E$ of edges such that $G - F$ is a cluster graph. Similarly, a *cluster vertex deletion* set (CVD set) of G is set $U \subseteq V$ of vertices such that $G[V \setminus U]$ is a cluster graph.

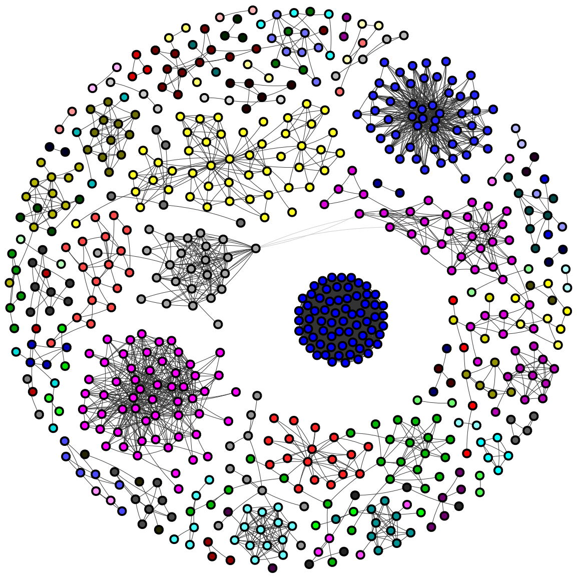


Figure 1: A large interaction graph of links inferred by protein tertiary structure in in *S. cerevisiae* yeast from WormNet database [5, 26].

The decision problems associated with these sets are defined as follows.

k-Cluster Deletion
Input: A graph $G = (V, E)$ and an integer k
Question: Does G have a cluster deletion set of size at most k?

k-Cluster Vertex Deletion
Input: A graph $G = (V, E)$ and an integer k
Question: Does G have a cluster vertex deletion set of size at most k?

A *path graph* is a graph P_n with vertices $V(P_n) = \{v_1, v_2, \ldots, v_n\}$ and $E(P_n) = \{\{v_i, v_{i+1}\} : i = 1, \ldots, n-1\}$. Cluster graphs cannot contain induced paths of more than two vertices, as this would imply the existence of at least two nonadjacent vertices in the same connected component. We formalize this in the following lemma, which we make frequent use of throughout our analysis.

Lemma 2.1. *A graph is a cluster graph if and only if it does not contain P_3 as an induced subgraph.*

The GA we present in this paper evolves a population of bit strings that correspond to a deletion set for induced subgraphs of a primary graph G. We identify each bit string of length n with a subset of $V(G)$ or a bit string of length m with a subset of $E(G)$. In particular, a bit string $x \in \{0,1\}^n$ indicates a subset S of V by

$$v_i \in S \iff x[i] = 1.$$

Similarly, a length-m bit string $x \in \{0,1\}^m$ indicates a subset of E. Because of this, we treat such bit strings identically to their corresponding subsets and the subset operations such as union, intersection and complement when applied to bit strings should be interpreted accordingly.

2.1 Parameterized Complexity

The theory of parameterized complexity [6, 9, 12] is a refinement to classical complexity theory in which algorithmic running time is decomposed into multiple parameters of the input. The goal is to isolate factors that contribute superpolynomially to the running time that are independent to input size. The theory helps to explain the practical performance of many algorithms on large real-world instances of NP-hard problems. Often such instances exhibit some kind of exploitable structure, and the worst-case complexity as a function of problem size alone is too pessimistic. Because of this, parameterized analyses are especially relevant to understanding the influence of problem structure on the running time of evolutionary algorithms on NP-hard optimization problems [22, 29].

Formally, a parameterized problem is 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.

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 $\mathbb{E}[T] \leq g(k) \cdot n^{O(1)}$.

3 The Subgraph Population GA

Recently, Lee and Sutton [18] introduced a genetic algorithm for graph problems that maintains a population consisting of feasible solutions to subgraphs of a primary graph G. The GA starts with a population of small easy-to-solve subgraphs of G, and solutions to larger and larger subgraphs of G are built up over time using a specialized crossover operation followed by a domain-specific repair operator. They proved that this technique solves k-vertex cover in FPT time.

In this paper, we adapt their algorithm to tackle graph clustering problems on an undirected graph G. Our adaptation maintains a population of individuals where each individual represents a feasible solution to an induced subgraph of G. This is already a slight departure from [18] where subgraphs were represented as edge sets. However, that representation is not suitable for Cluster Deletion and Cluster Vertex Deletion since cluster graphs are not closed under edge deletions.

To represent candidate solutions in this framework, we must represent a *deletion set* (a set of either vertices or edges, depending on which clustering problem we are solving), and a corresponding subgraph. Each individual genotype g is thus represented as a pair (x, G') where x is a bit string and G' is some induced subgraph of G. In this way, the string x is a candidate solution to the subgraph G'. We say that a genotype $g = (x, G')$ is feasible when x is a k-deletion set for its corresponding subgraph. Specifically,

(1) Deleting the elements chosen by x from G' results in a cluster graph, and
(2) The total number of elements deleted from G' is at most k.

Elements selected by x that are not part of the subgraph G' are ignored in the determination of feasibility.

Survival selection is performed by calculating a domination relation that takes into account feasibility, induced subgraph containment and candidate solution size, defined as follows.

Definition 3.1. A genotype $g = (x, G')$ is *dominated* by a genotype $h = (y, G'')$, written as $g \prec h$, if

(1) g is infeasible but h is feasible,
(2) both g and h are feasible but $V(G') \subset V(G'')$, or
(3) both g and h are feasible, $G' = G''$ and $|x| > |y|$.

If both g and h are feasible, $G' = G''$ and $|x| = |y|$, then g and h are *tied*. If h dominates g or g and h are tied, we write $g \preceq h$. If g and h are not tied and neither dominate the other, we say they are *incomparable*.

The genetic algorithm proceeds as follows. In each generation, with probability $0 < p_c < 1$, two parents are selected uniformly at random to undergo recombination followed by a feasibility repair operator to produce an offspring which subsequently undergoes mutation. Otherwise, with probability $1 - p_c$, one parent is selected to produce an offspring via mutation alone.

Rather than employing the so-called generalized allelic crossover operator as the authors in [18], the SubPopGA for clustering chooses parents (x, G') and (y, G'') from the population uniformly at random and constructs a child (z, H) by (1) creating the graph H as the induced subgraph of the union of the parent subgraphs $H = G[V(G') \cup V(G'')]$, and (2) producing an offspring bit string z by performing a three-way uniform crossover on the parent bit strings and a third "template" parent that is constructed in a manner discussed below in Section 3.1. Given three bit strings a, b and c, three-way uniform crossover produces a bit string $s \leftarrow \textsc{UniformCrossover}(a,b,c)$ by choosing each s_i uniformly at random from $\{a_i, b_i, c_i\}$. The following lemma will be useful for analyzing the effectiveness of the crossover operator.

Lemma 3.2. *Let a, b and c be bit strings, each of the same length l. If for all $i \in \{1, \ldots, l\}$, $0 \in \{a_i, b_i, c_i\}$, then for any $s \subset (a \cup b \cup c)$,* $\Pr(s = \textsc{UniformCrossover}(a,b,c)) \geq (1/3)^{|a|+|b|+|c|}$.

Proof. Fix $s \in (a \cup b \cup c)$. Clearly for all $i \notin (a \cup b \cup c)$, $s_i = a_i = b_i = c_i = 0$ with probability one. Thus, suppose $i \in (a \cup b \cup c)$. If $s_i = 0$, then since $0 \in \{a_i, b_i, c_i\}$, crossover produces a zero in position i with probability at least $1/3$. If $s_i = 1$, crossover produces a one in position i with probability at least $1/3$, since $i \in (a \cup b \cup c) \implies 1 \in \{a_i, b_i, c_i\}$. There are at most $|a| + |b| + |c|$ such positions $i \in (a \cup b \cup c)$, which yields the claim. □

After an offspring is produced, the process undergoes survival selection similar to the SEMO/GSEMO algorithms from evolutionary multiobjective optimization [17]. In particular, if the generated offspring is not dominated or tied (in the sense of Definition 3.1)

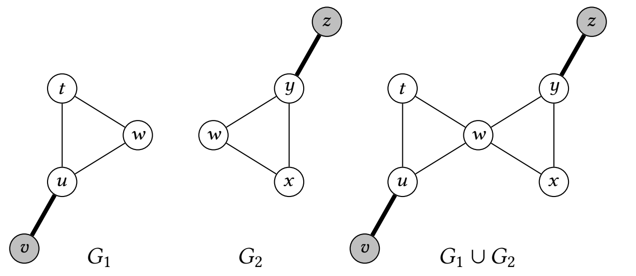


Figure 2: Two induced subgraphs G_1 and G_2 and their union $G_1 \cup G_2$. $\{uv\}$ is a feasible CD set for G_1 and $\{yz\}$ is a feasible CD set for G_2, but $\{uv, yz\}$ is not a feasible CD set for $G_1 \cup G_2$. Similarly, $\{v\}$ is a feasible CVD set for G_1 and $\{z\}$ is a feasible CVD set for G_2, but $\{v, z\}$ is not a feasible CVD set for $G_1 \cup G_2$.

by any individuals in the current population, it survives to the next population. Any individual in the current population that is dominated by the generated offspring does not survive. The SubPopGA for clustering is listed in Algorithm 1.

Algorithm 1: SubPopGA for clustering

```
Input: A graph $G = (V, E)$, a crossover probability $p_c$ and a
       mutation rate $p_m$
1  Initialize $P_0$
2  $t \leftarrow 0$
3  while $P_t$ does not contain optimum do
4      Choose distinct parents $(x, G'), (y, G'') \in P_t$ u.a.r.
5      with probability $p_c$ do
6          $H \leftarrow G[V(G') \cup V(G'')]$
7          Create a template parent $\tau$ based on $H$ and $x$ and $y$
8          $z \leftarrow$ UniformCrossover$(x, y, \tau)$
9          $z \leftarrow$ ConstraintRepair$(x, y, z, \tau, H)$
10     else
11         $(z, H) \leftarrow (x, G')$
12     flip each bit of $z$ w/ prob. $p_m$
13     Let $h := (z, H)$ denote the resulting offspring
14     if $\nexists g \in P_t$ s.t. $h \preceq g$ then
15         $P_{t+1} = \{g \in P_t \mid g \not\prec h\} \cup \{h\}$
16     $t \leftarrow t + 1$
```

3.1 Template Parents and Constraint Repair

A technical problem arises when trying to adapt the population-based subgraph technique of Lee and Sutton [18] to graph clustering problems. In particular, unlike vertex covers, cluster deletion sets are in general not closed under union (see Figure 2). This property is critical in order for crossover and constraint repair to probabilistically simulate a classical technique called *iterative compression.*

In order to address this issue, we introduce the concept of an additional *template parent* that is injected into the recombination process. The template parent is constructed by a domain-specific routine in such a way that ensures that three-parent uniform crossover among the two original parents and the template parent has a reasonable chance of producing an offspring that can be efficiently repaired to make evolutionary progress. Later, we will prove that the addition of such a template parent results in an FPT run time guarantee.

The template parent generation procedure and the repair procedure are the only elements of Algorithm 1 that require domain-specific information for their design, and we will define these components for k-Cluster Deletion and k-Cluster Vertex Deletion in Sections 4.1 and 4.2, respectively. After a template parent τ is created in line 7 of the SubPopGA, three-way uniform crossover is applied to the three strings comprised of the two parents together with the template parent.

The second requirement of the SubPopGA is a constraint repair operator that attempts to repair infeasible offspring, that is genotypes (z, H) produced by crossover such that either z is not a deletion set for the induced subgraph H, or it is a deletion set, but it is too large. The repair operator is deterministic, and attempts to find a deletion set disjoint from the elements that were unique to the parents, i.e., appeared in at least one parent, but not the offspring. We describe the repair procedures in Sections 4.1 and 4.2.

3.2 Initial Population and Feasibility

A trivial subgraph consisting of a single vertex is a cluster graph. Thus for any $v \in V(G)$, the genotype $(x, G[\{v\}])$ is feasible for arbitrary x. We set the initial population in line 1 to

$$P_0 := \bigcup_{v \in V(G)} \{(x, G[\{v\}])\}, \tag{1}$$

where x is a bit string initialized uniformly at random. Since each $G[\{v\}]$ is already a cluster graph, it follows that P_0 is feasible.

The following lemma is adapted from [18] to our setting.

Lemma 3.3. *Let P_t denote the population of Algorithm 1 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. Let $h = (z, H)$ be the offspring produced in line 13 of Algorithm 1. In line 15, h is only added to P_{t+1} if it is not dominated or tied by any individual P_t. The first condition trivially holds, as all infeasible solutions are automatically dominated by feasible solutions.

Let $g = (x, G') \in P_t$ be the parent selected in line 4 of Algorithm 1. We argue that either $g \prec h$ or $h \prec g$. If h is not feasible, then the latter clearly holds, since g is feasible. Thus, assume that h is also feasible. Note that $V(G') \subseteq V(H)$, regardless of whether h was created by mutation only or by crossover and mutation. If $G' \neq H$, then $g \prec h$. If $G' = H$, then since $|x| \neq |z|$ (otherwise g and h would be tied), if $|x| < |z|$, then $h \prec g$, otherwise $g \prec h$.

In either case, it follows that $h \in P_{t+1} \iff g \notin P_{t+1}$ and thus we have $|P_{t+1}| \leq |P_t|$. □

A feasible genotype (x, G') may contain "non-coding" bits in x that do not correspond to elements of G' when G' is a proper subgraph of G. For example, position i is non-coding when $x_i = 1$ but $i \notin V(G')$ when x selects vertices, or $i \notin E(G')$ if x selects edges. These non-coding bits can become problematic during crossover because they could be inherited from one parent while damaging the feasibility of the graph inherited from the other. The third domination criterion of Definition 3.1 introduces selective pressure to minimize non-coding bits. Note that if $|x| > k$, the genotype must contain non-coding bits, otherwise it would not be feasible. These bits can always be removed by mutation.

Definition 3.4. A genotype $g = (x, G')$ is *efficient* when $|x| \le k$. A feasible genotype that is not efficient contains at least $|x| - k$ non-coding bits.

We prove the following general theorem about the SubPopGA.

Theorem 3.5. *Consider the SubPopGA (Algorithm 1) evolving bit strings of length ℓ and let $0 < p^* < 1$ be a lower bound on the probability that crossover results in a feasible offspring by recombination between any two (non-optimal) parents with no non-coding bits. Then the expected runtime of the SubPopGA is at most*

$$\frac{(n^2+n)\ell\,(1+2\ln\ell)}{2\,(1-p_c)\,p_m\,(1-p_m)^{\ell-1}} + \frac{n-1}{p_c p^*}.$$

Proof. The initial population P_0, described in Equation (1) is feasible and the vertices of the subgraphs form a partition of the vertices of the primary graph G. Thus, by Lemma 3.3, during the run of the SubPopGA, the population size cannot increase and always remain feasible. Moreover, a feasible population of size one corresponds to a feasible solution to the primary graph G, as subgraphs are always combined by vertex union in line 6 of Algorithm 1.

Suppose at some time s we have $|P_s| = N > 1$, and let $T_N \coloneqq \inf\{\zeta \in \mathbb{N} : |P_{s+\zeta}| < N\}$. A necessary and sufficient condition for the population to shrink in size is that crossover occurs and produces a feasible offspring that replaces both parents. We divide the time interval $s \le t \le s + T_N$ into two phases. In the first phase, the population contains genotypes that are not efficient. In the second phase, all genotypes in the population are efficient (note that one phase may be empty). By the third domination criterion in Definition 3.1, the number of one-bits in a particular individual cannot increase in any step where crossover does not occur. Thus, once all the genotypes in the population are efficient, they will remain so until a successful crossover operation occurs at time $s + T_N$.

Let $(X_t)_{t\in\mathbb{N}}$ be the sequence of random variables where X_t counts all the non-coding bits in the population at time t:

$$X_t \coloneqq \sum_{(x,G')\in P_t} \max\{0, |x| - k\}.$$

We pessimistically assume that all individuals must become efficient before an improving crossover occurs (i.e., the second phase is nonempty). Note that X_t cannot increase before $s + T_N$, and thus we seek to bound its drift during the first phase.

To do so, pick an arbitrary $(x, G') \in P_t$. If $\max\{0, |x| - k\} = 0$, then x contributes nothing to X_t, nor will it contribute to the drift, since a successful mutation cannot increase the number of one bits. Thus, assume that $\max\{0, |x| - k\} > 0$. Since x must be feasible, there are at least $\max\{0, |x| - k\}$ non-coding one bits in x. Hence, there are at least $\max\{0, |x|-k\}$ single-bit neighbors of x that would be viable offspring if the process (1) selected x as a parent, (2) did not perform crossover and (3) produced one of these neighbors. The probability of such an event is at least $(1/N)(1-p_c)(1-p_m)^{\ell-1}p_m$. Since we picked x arbitrarily, by linearity of expectation, the drift $\mathbb{E}[X_t - X_{t+1} \mid X_t]$ is at least

$$\sum_{(x,G')\in P_t} \frac{\max\{0, |x| - k\} p_m (1-p_c)(1-p_m)^{\ell-1}}{N}.$$

This yields the drift lower bound

$$\mathbb{E}[X_t - X_{t+1} \mid X_t] \ge \left(\frac{p_m(1-p_c)(1-p_m)^{\ell-1}}{N}\right) \cdot X_t.$$

Once $X_t = 0$, all individuals are efficient and the second phase begins. By the multiplicative drift theorem [8], the expected time for the first phase is at most

$$\frac{N\,(1+2\ln\ell)}{p_m\,(1-p_c)\,(1-p_m)^{\ell-1}}$$

In the second phase, all individuals in the population are efficient, and the probability of a successful crossover is at least $p_c p^*$. The time in the second phase is geometrically distributed with expectation at most $(p_c p^*)^{-1}$.

Let $T = \inf\{t \in \mathbb{N} : |P_t| = 1\}$. Then

$$\mathbb{E}[T] = \sum_{N=2}^{n} \mathbb{E}[T_N] \le \sum_{N=2}^{n} \left(\frac{N\,(1+2\ln\ell)}{p_m\,(1-p_c)\,(1-p_m)^{\ell-1}} + \frac{1}{p_c p^*}\right),$$

which yields the claimed bound. □

4 FPT Runtime

In this section, we prove the SubPopGA solves both k-Cluster Deletion and k-Cluster Vertex Deletion in randomized FPT time. We always assume that the primary graph G always has a k-deletion set, namely a set $F \subseteq E$ where $G - F$ is a cluster graph and $|F| \le k$ in the case of k-Cluster Deletion and set $S \subseteq V$ where $G[V \setminus S]$ is a cluster graph and $|S| \le k$ in the case of k-Cluster Vertex Deletion. This assumption is justified because the FPT runtime bounds provided in this section can be used to design a Monte-Carlo restart framework to search for the smallest k with probabilistic guarantees on the success of each run [3].

4.1 Cluster Deletion

In the k-Cluster Deletion problem, we are given a graph G and we seek a set of edges whose deletion from G results in a cluster graph. In this setting, each individual genotype is represented as a pair $g = (x, G')$ where $x \in \{0,1\}^m$ (indicating a subset of $E(G)$) and G' is an induced subgraph of G. Then g is feasible when $G' - x$ is a cluster graph and $|E(G') \cap x| \le k$.

To generate a template parent for crossover with parents (x, G') and (y, G''), we compute a length-m bit string τ that corresponds to a set of edges from induced P_3 graphs in $H = G' \cup G''$. This is detailed in Algorithm 2.

The following lemma bounds the number of edges selected by the template parent constructed in Algorithm 2.

Lemma 4.1. *Suppose G has a CD set of size at most k. Let G' be an induced subgraph of G. Then the cardinality of the template parent computed by Algorithm 2 is at most $|\tau| \le 2k$.*

Proof. Every edge pair $e, f \in E(G')$ from an induced P_3 in G' also appears in a P_3 in G. Let F^* be an optimal CD set for G, i.e.,

Algorithm 2: Template parent for Cluster Deletion

```
Input: A graph $H$ and two length-$m$ bit strings $x$ and $y$
1 $\tau \leftarrow 0^m$
2 while $H - (x \cup y \cup \tau)$ contains an induced $P_3$ do
3     let $\{e, f\} \in E(H)$ be a $P_3$ in $H - (x \cup y \cup \tau)$
4     $\tau \leftarrow \tau \cup \{e, f\}$
5 return $\tau$
```

$G - F^*$ is a cluster graph and F^* is minimal. Every P_3 of G must have at least one edge in F^*, thus there are at most $|F^*|$ disjoint P_3s in F^*. The while loop in Algorithm 2 executes at most $|F^*|$ times before the resulting graph is P_3-free. Since each P_3 removal adds two edges to τ, it follows that $|\tau| \leq 2|F^*| \leq 2k$. □

We model the repair operator for Cluster Deletion after the one for k-Vertex Cover [18]. For vertex cover, if crossover "decides" a vertex v should not be in the cover, the repair operator must ensure all the elements in $N(v)$ are in the cover to enforce a feasible cover. Similarly, in Cluster Deletion, if crossover removes an edge e that results in a P_3 $\{e, f\}$, then f must be placed in the deletion set to enforce feasibility.

Algorithm 3: Repair for Cluster Deletion

```
Input: An offspring $(z, H)$, and three length-$m$ bit strings
       $x, y, \tau$ and a primary graph $G = (V, E)$
1 if $z$ indicates a feasible $k$-CD set for $H$ then return $z$
2 $A \leftarrow (x \cup y \cup \tau) \setminus z$
3 foreach edge $e \in A$ do
4     if $\exists f \in E(H) \setminus A$ such that $\{e, f\}$ is a $P_3$ then
          $z \leftarrow z \cup \{f\}$
5 return $z$
```

The success of the repair operator depends directly on the outcome of uniform crossover. We prove in the following lemma that crossover between two efficient parents has a decent chance of creating input to the repair operator that directly results in an improving offspring.

Lemma 4.2. *If (x, G') and (y, G'') are both efficient and are chosen as parents in line 4 of the SubPopGA, the probability that the produced offspring $h = (z, H)$ in line 13 dominates both parents is at least $p_c 3^{-4k}$.*

Proof. We assume that G has a k-CD set S^*. Note that $H - S^*$ must be a cluster graph since otherwise an induced P_3 in H not removed by S^* would not be removed from G. For the remainder of the proof, we assume that crossover occurs, and this is accounted for by the p_c factor in the claimed bound.

Let $F = x \cup y \cup \tau$ be the edge set constructed by taking the union of the parents and the template parent created in line 7 of Algorithm 1 by the template parent creation procedure for Cluster Deletion (Algorithm 2). By construction, F is a CD-set for H.

As (x, G') and (y, G'') are feasible and efficient, $|x|, |y| \leq k$. Moreover, since by Lemma 4.1, $|\tau| \leq 2k$, it follows that $|F| \leq 4k$. In Algorithm 2, an edge is only added to the template parent if it is not already in one of the parents. Thus for all $i \in \{1, \dots, m\}$ there is a zero in position i in at least one of the three bit strings x, y and τ. By Lemma 3.2, the three-way uniform crossover operation in line 8 of the SubPopGA produces the string $z = F \cap S^*$ with probability at least $3^{-|F|} \geq 3^{-4k}$.

If z is a feasible k-CD set for H, the proof is complete since H is a supergraph of both G' and G''. Otherwise, assume $H - z$ contains an induced $P_3 = (uv, vw)$. This means $uw \notin E(G)$. One of the edges of the P_3 must have been in $F \setminus S^*$ (since F was a CD-set). It follows that the other would be in S^*, otherwise there would be an induced P_3 in $G - S^*$. Since this is true for every P_3 in $G - z$, it follows that constraint repair would add all of the edges in $S^* \cap E(H)$. In particular, after constraint repair, we would have $|z| = |S^* \cap E(H)| \leq |S^*| \leq k$, and z would be a feasible CD-set for H. □

We are now ready to prove an FPT runtime result of the SubPopGA on k-Cluster Deletion, which is a direct result of Theorem 3.5.

Theorem 4.3. *Suppose $G = (V, E)$ is a graph on n vertices and m edges. If G has a CD set of size at most k, then the SubPopGA (Algorithm 1) with constant crossover probability $0 < p_c < 1$ and $p_m = 1/m$ solves the k-Cluster Deletion problem on G in expected time $O(mn^2 \log n + 3^{4k} n)$.*

Proof. Lemma 4.2 yields a 3^{-4k} lower bound on the probability of a successful crossover between nonoptimal efficient parents. Applying Theorem 3.5 with $p_c = \Theta(1)$, $p_m = 1/m$ and $p^* = 3^{-4k}$ proves the claim. □

4.2 Cluster Vertex Deletion

In the k-Cluster Vertex Deletion problem, we seek for a primary graph G on n vertices and m edges a set of vertices whose deletion from $V(G)$ results in a cluster graph. An individual genotype is a pair $g = (x, G')$ where $x \in \{0, 1\}^n$ (indicating a subset of $V(G)$) and G' is an induced subgraph of G. In this setting, g is feasible when $G[V(G') \setminus x]$ is a cluster graph and $|V(G') \cap x| \leq k$.

Similar to k-Cluster Deletion, we generate a template parent for crossover with parents (x, G') and (y, G''), by constructing a length-n bit string τ that corresponds to a set of vertices from induced P_3 graphs in $H = G' \cup G''$. We list the template parent construction procedure in Algorithm 4.

Algorithm 4: Template parent for Cluster Vertex Deletion

```
Input: A graph $H$ and two length-$n$ bit strings $x$ and $y$
1 $\tau \leftarrow 0^n$
2 while $H[V(H) \setminus (x \cup y \cup \tau)]$ contains an induced $P_3$ do
3     Choose vertices $\{u, v, w\}$ that form an induced $P_3$ in
        $H[V(H) \setminus (x \cup y \cup \tau)]$
4     $\tau \leftarrow \tau \cup \{u, v, w\}$
5 return $\tau$
```

In line 7 of Algorithm 1, the template parent is constructed from the subgraph $H = G' \cup G''$ which is formed by the union of the parent genotypes. This union graph is used to create the template parent, which is guaranteed to form a CVD set for H together with the initial parent strings x and y. Similar to the case with Cluster Deletion, we may bound the cardinality of the template parent.

Lemma 4.4. *Suppose G has a CD set of size at most k. Let G' be an induced subgraph of G.*

If G' is an induced subgraph of a graph G with an optimal CVD set of size at most k, then the cardinality of the template parent returned in Algorithm 4 is bounded by $|\tau| \leq 3k$.

Proof. If there is a set of vertices $\{u, v, w\} \in V(G')$ that induces a P_3 in G', then it must also induce a P_3 in G. Let S^* be a k-CVD set for G. Since every P_3 in G must have at least one vertex in S^*, it follows that $|S^* \cap \{u, v, w\}| > 0$ and there can be at most $|S^*|$ disjoint P_3s in G, and consequently G'. Thus the while loop in Algorithm 4 executes at most $|S^*|$ times before the resulting graph is P_3-free. Since each time through the loop adds three vertices to τ, we have $|\tau| \leq 3|S^*| \leq 3k$, as claimed. □

The repair operator for k-Cluster Vertex Deletion must attempt to build a CVD set D that is distinct from the set of vertices that were removed from one of the parents by crossover. In order to do this, we take an approach functionally similar to a critical step in the iterative compression algorithm for Cluster Vertex Deletion introduced by Hüffner et al. [16]. However, instead of constructing an auxiliary graph in which to find a maximal matching, we minimize a corresponding linear program. We find this perspective slightly more natural, as the procedure seeks to minimize the size of the resulting CVD set.

Let (x, G') and (y, G'') be the crossover parents, τ be the template parent generated by Algorithm 4, and suppose uniform crossover in line 8 of Algorithm 1 produces a string z. The resulting offspring genotype is (z, H) where $H = G[V(G') \cup V(G'')]$.

Let A be the set

$$A \coloneqq (x \cup y \cup \tau) \setminus z.$$

consisting of the vertices in at least one of the parents, but not the resulting offspring. If $G[A]$ is not a cluster graph, we simply give up the repair attempt. Otherwise, we are able to start constructing a CVD set D that is disjoint from A. Let $A_1, A_2, \ldots, A_\ell$ be the clusters of $G[A]$. Note that if a vertex $v \in V(H) \setminus A$ is adjacent to a vertex $u \in A_i$ and a vertex $w \in A_j$ where $i \neq j$ it must belong to every CVD set disjoint from A because $\{u, v, w\}$ induces a P_3.

Similarly, if a vertex $v \in V(H) \setminus A$ is adjacent to a vertex $u \in A_i$ but *not* adjacent to a vertex $w \in A_i$, then v must belong to every CVD set disjoint from A, again because $\{u, v, w\}$ induces a P_3. Thus we may safely add all such vertices to D. It remains to classify the remaining vertices, the set of which we call B. Note that $B \subseteq V(H) \setminus (x \cup y \cup \tau)$ and, by construction in Algorithm 4, $(x \cup y \cup \tau)$ is a CVD set for H. It follows that $G[B]$ must be a cluster graph.

We construct two partitions of B. Let $\mathcal{B} = \{B_1, \ldots, B_m\}$ be the partitions of B corresponding to the clusters of $G[B]$. Let $\mathcal{C} = \{C_0, C_1, \ldots, C_\ell\}$ be a partition of B where $v \in C_j, j > 0$ if v is adjacent to all vertices in cluster A_j. Similarly, C_0 consists of all vertices in B not adjacent to any vertex in A. We have the following lemma.

Lemma 4.5. *Let A and B be disjoint vertex sets such that $G[A]$ and $G[B]$ are both cluster graphs and no vertex in B is adjacent to different clusters in $G[A]$, or adjacent to only part of a cluster in $G[A]$. Consider the partitions $\mathcal{B}$ and $\mathcal{C}$ as defined above.*

A set $S \subseteq B$ is a CVD set for the graph $G[A \cup B]$ disjoint from A if and only if the following two conditions hold.

Condition 1. *For all i, $S \supseteq B_i \setminus C_j$ for some j,*
Condition 2. *For all $j > 0$, $C_j \setminus S \subseteq B_i$ for some j.*

Proof. Every induced P_3 in $G[A \cup B]$ must contain exactly one vertex in A. Since $G[A]$ and $G[B]$ are both already cluster graphs, there cannot be a P_3 contained only within A or only within B. Moreover, any P_3 involving two vertices in A are ruled out by the conditions on membership in B.

We point out that Condition 1 of the lemma holds if and only if, after deleting S, there is no induced P_3 involving two vertices u and v within some cluster of $G[B]$. In particular, for all $i \in [m]$ and all $u, v \in B_i$, if u and v are in different $\mathcal{C}$-partitions, then at least one of them must belong S. Otherwise, since at least one of them is adjacent to a vertex w in $G[A]$, and they both cannot be adjacent to the same cluster in $G[A]$, it follows that $\{u, v, w\}$ would form a P_3 in $G[A \cup B]$.

Similarly, Condition 2 holds if and only if, after deleting S, there is no induced P_3 involving vertices u and v in different clusters of $G[B]$. For all $j \in [\ell]$, if $u, v \in C_j$, but $u \in B_i$ and $v \in B_{i'}$ for $i \neq i'$, then at least one of u, v must belong to S. Otherwise, since $j > 0$, u and v are both adjacent to a vertex w in A_j, and since they are in different clusters of $G[B]$, they are not adjacent. It follows that $\{u, v, w\}$ would form a P_3 in $G[A \cup B]$. □

It follows from Lemma 4.5 that in order to find a CVD set of $G[A \cup B]$ disjoint from A, it suffices to find a set $S \subseteq B$ that satisfies the conditions of the lemma. The minimum of such sets can be found from the solution to the linear program:

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{m} \sum_{j=0}^{\ell} X_{i,j} |B_i \setminus C_j| \\ \text{subject to} \quad & \sum_{j=0}^{\ell} X_{i,j} = 1, \qquad i = 1, \ldots, m \\ & \sum_{i=1}^{m} X_{i,j} \leq 1, \qquad j = 1, \ldots, \ell \\ & 0 \leq X_{i,j} \leq 1, \qquad i = 1, \ldots, m; j = 0, \ldots, \ell \end{aligned} \tag{2}$$

The first constraint of the linear program corresponds to Condition 1 of Lemma 4.5, and the second constraint to Condition 2 of the lemma. The constraint matrix for the linear program in Equation (2) is totally unimodular [15] and therefore the optimal solution is integral. Thus the set of minimum size that satisfies the conditions of Lemma 4.5 is exactly

$$S = \bigcup_{X_{i,j}=1} \left(B_i \setminus C_j\right).$$

The entire constraint repair procedure for Cluster Vertex Deletion is listed in Algorithm 5.

The above observation about the linear program described in Equation (2) together with Lemma 4.5 yields the following.

Algorithm 5: Repair for Cluster Vertex Deletion

```
Input: An offspring $(z, H)$, parents $x, y, \tau$ and a graph
       $G = (V, E)$
1  if $z$ is a feasible $k$-CVD set for $H$ then return $z$
2  $A \leftarrow (x \cup y \cup \tau) \setminus z$
3  if $G[A]$ is not a cluster graph then return failure
   // Classify remaining vertices
4  $B \leftarrow D \leftarrow \emptyset$
5  foreach $v \in V(H) \setminus (A \cup z)$ do
6      if $v$ is adjacent to more than one cluster in $G[A]$ or $v$ is
       adjacent to some but not all of a cluster in $G[A]$ then
       $D \leftarrow D \cup \{v\}$
7      else $B \leftarrow B \cup \{v\}$
8  Let $\mathcal{A} = \{A_1, A_2, \dots, A_\ell\}$ be the clusters of $G[A]$
9  Let $\mathcal{B} = \{B_1, B_2, \dots, B_m\}$ be the clusters of $G[B]$
10 Let $C = \{C_0, C_1, C_2, \dots, C_\ell\}$ be a partition of $B$ such that
       $v \in C_i \iff \begin{cases} N(v) \cap A = \emptyset & \text{if } i = 0, \\ N(v) \cap A = A_i & \text{if } i > 0 \end{cases}$
11 Compute the solution $X_{i,j}$ to the linear program in Eq. (2)
12 return $z \cup D \cup \left( \bigcup_{\{i,j\,:\, X_{i,j}=1\}} (B_i \setminus C_j) \right)$
```

Lemma 4.6. *Algorithm 5, when called with offspring (z, H) and parents x, y, τ, returns a bit string corresponding to the smallest CVD set in H disjoint from $(x \cup y \cup \tau) \setminus z$.*

Proof. Algorithm 5 sets $A = (x \cup y \cup \tau) \setminus z$ and sets B to $V(H) \setminus (A \cup z \cup D)$ where D contains vertices guaranteed to be in every CVD set disjoint from A. To complete a CVD set, the remaining vertices from B are selected to satisfy Conditions 1 and 2 of Lemma 4.5, and the LP solution to Equation (2) guarantees the minimum number of vertices that satisfy the conditions. □

We now show that uniform crossover between efficient parents has a reasonable chance of producing a string that can be repaired by Algorithm 5 to result in an improving offspring.

Lemma 4.7. *If (x, G') and (y, G'') are both efficient and are chosen as parents in line 4 of the SubPopGA, the probability that the offspring $h = (z, H)$ in line 13 dominates both parents is at least $p_c 3^{-5k}$.*

Proof. We assume that G has a k-CVD set S^*. We argue that $G[V(H) \setminus S^*]$ must be a cluster graph. Suppose for contradiction that $\{u, v, w\} \in V(H) \setminus S^*$ form an induced P_3. Then $\{u, v, w\}$ must also induce a P_3 in $G[V \setminus S^*]$, which contradicts that S^* is a CVD set. For the remainder of the proof, we assume that crossover occurs, and this is accounted for by the p_c factor in the claimed bound.

Let $S = x \cup y \cup \tau$ be the vertex set constructed by taking the union of the parents and the template parent created in line 7 of Algorithm 1 by the template parent creation procedure for Cluster Vertex Deletion (Algorithm 4). By construction, S is a CVD-set for H.

Since (x, G') and (y, G'') are feasible and efficient bits, $|x|, |y| \le k$ and by Lemma 4.4, $|\tau| \le 3k$, it follows that $|S| \le 5k$. In Algorithm 4, an vertex is only added to the template parent if it is not already in one of the parents. Thus for all $i \in \{1, \dots, n\}$ there is a zero in position i in at least one of the three bit strings x, y and τ. By Lemma 3.2, the three-way uniform crossover operation in line 8 of the SubPopGA produces the string $z = S \cap S^*$ with probability at least $3^{-|S|} \ge 3^{-5k}$.

Under this event, by Lemma 4.6, the repair operation in Algorithm 5 computes the minimum size CVD-set for H disjoint from $S \setminus z = S \setminus (S \cap S^*) = S \setminus S^*$.

Since S^* is also a CVD set for H, we know that there is a CVD set for H that is disjoint from $S \setminus S^*$ and is size at most k. Thus the repair operation adds no more than $k - |S \cap S^*|$ vertices into z. Since z began with $|S \cap S^*|$ vertices prior to the call to repair, it follows that under this event $|z| \le k$ after repair. □

Similar to Theorem 4.3, the FPT result for the SubPopGA on k-Cluster Vertex Deletion now follows easily from Theorem 3.5.

Theorem 4.8. *Suppose G is a graph on n vertices and m edges. If G has a CVD set of size at most k, then the SubPopGA (Algorithm 1) with constant crossover probability $0 < p_c < 1$ and $p_m = 1/n$ solves the k-Cluster Vertex Deletion problem on G in expected time $O(n^3 \log n + 3^{5k} n)$.*

Proof. Lemma 4.7 yields a 3^{-5k} lower bound on the probability of a successful crossover between nonoptimal efficient parents. Applying Theorem 3.5 with $p_c = \Theta(1)$, $p_m = 1/n$ and $p^* = 3^{-5k}$ proves the claim. □

5 Computational results

In this section we report the results on experiments that evaluate the SubPopGA on a real-world biological data set. The data set comes from the connected components of a similarity graph constructed from 138,458 proteins from 66 genomes of unicellular organisms from the COG database [30]. We chose this particular data set because it has been used to test exact and heuristic algorithms on other cluster editing problems, most notably Weighted Cluster Editing [1, 24], Cluster Editing [2, 14] and 2-Club Cluster Vertex Deletion [10].

The original data yields a set of 3964 nontrivial connected components in a weighted similarity graph. The weights correspond to similarity scores calculated by bidirectional BLAST hits with a threshold of 10^{-10} (for details, see [24]). Because k-Cluster Deletion and k-Cluster Vertex Deletion instances are unweighted, we convert the weighted similarity graph to an unweighted similarity graph using the threshold approach of Hartung and Hoos [14] for Cluster Editing by including only (unweighted) edges for pairs with similarity scores in the top c% for $c = 33, 66$. This approach was also used by Figiel et al. on 2-Club Cluster Vertex Deletion [10] (a variant of the vertex deletion problem in which cluster graph components are allowed to have a diameter of at most two). We also removed any vertices of zero degree that remained after the conversion. This resulted in 7928 undirected graphs in total, 3964 for each c value.

On each of the 7928 graphs, we ran a MIP solver to determine the true optimal solution for both Cluster Vertex Deletion and Cluster Deletion. For the $c = 33$ graphs, the MIP solver successfully found the optimal CVD set size for 3936 of the 3964 graphs

	CVD-Bio33			CVD-Bio66			CD-Bio33		
k	**solved**	**total**	**rate**	**solved**	**total**	**rate**	**solved**	**total**	**rate**
1	10250	10250	100%	11790	11790	100%	6326	6330	99.94%
2	4350	4350	100%	5920	5920	100%	2924	2970	98.45%
3	2500	2500	100%	3190	3190	100%	2075	2110	98.34%
4	1680	1680	100%	2130	2130	100%	956	1010	94.65%
5	1320	1320	100%	1500	1500	100%	970	1070	90.65%
6	1100	1100	100%	1149	1150	99.91%	682	870	78.39%
7	739	740	99.86%	990	990	100%	436	650	67.08%
8	571	590	96.78%	810	820	98.78%	280	550	50.91%
9	415	474	87.55%	613	692	88.58%	253	600	42.17%
10	373	520	71.73%	339	476	71.22%	79	370	21.35%

Table 1: Runs solved, total runs and success rate for each k value for the SubPopGA solving k-Cluster Vertex Deletion and k-Cluster Deletion.

within the allotted time, and we hereafter refer to this data set as CVD-Bio33. For the $c = 66$ graphs, the MIP solver successfully found the optimal CVD set size for all but five of the 3964 graphs, and we refer to this data set as CVD-Bio66. For Cluster Deletion, we only report data for the $c = 33$ graphs. On these graphs, the MIP solver was only able to find an optimal CD set of edges on 3932 of the 3964 graphs and we refer to this data set as CD-Bio33.

For each of the three data sets, we filtered out the instances with $k \in \{1, \ldots, 10\}$. The CVD-Bio33 set had 3244 graphs with $k \leq 10$, but only 2353 of these had $k > 0$ (meaning that the remaining 891 instances were already trivially cluster graphs). The CVD-Bio66 set had 3290 graphs with $k \leq 10$, with 2867 nontrivial instances. The CD-Bio33 set had 2538 graphs with $k \leq 10$, with only 1653 nontrivial instances.

On each of the filtered graphs, for the corresponding k value, we ran the SubPopGA ten times to solve k-Cluster Vertex Deletion with a cutoff value of 10^6. We record the run time as the number of generations until a k-CVD set was found for the input graph. For $k < 9$, all or nearly all runs successful solved the problem within the cutoff. For $k = 9$, roughly 90% of the runs were successful and for $k = 10$, only around 70% of the runs were successful within the cutoff. The details are reported in columns 2–7 of Table 1. We repeated these experiments for k-Cluster Deletion on the CD-Bio33 data set. Interestingly, the SubPopGA did not perform as well on this problem set, only solving at least 90% of the runs where $k < 6$. For $k = 8$, only half of the runs were successful, and only 21% of the runs succeeded on instances where $k = 10$. The detailed numbers are reported in the last three columns of Table 1.

We also report empirical run time distribution functions calculated for each group of graphs of a particular k-value for in Figure 3. We report in Figure 4 the average run time of the SubPopGA over all graphs as a function of k (removing failed runs). The error bars denote standard error. As expected, the GA runtime has a strong exponential dependence on k.

To investigate the dependence on the size of the graph, we plot the mean runtime for a collection of k-values in 5. To keep the plots uncluttered, we only plot the results for even values of k. For the k-Cluster Vertex Deletion instances, the graph size is measured as the number of vertices n. For the k-Cluster Deletion instances, the graph size is measured as the number of edges m. Note that here we use a logarithmic axis due to the high variability on the biological data, and we again include error bars denoting standard error. To observe the run time in the context of the $O(n^3 \log n)$ upper bound proved in Theorem 4.8, we repeat this plot in Figure 6, dividing the runtime by $n^3 \ln n$, this time omitting the error bars for clarity.

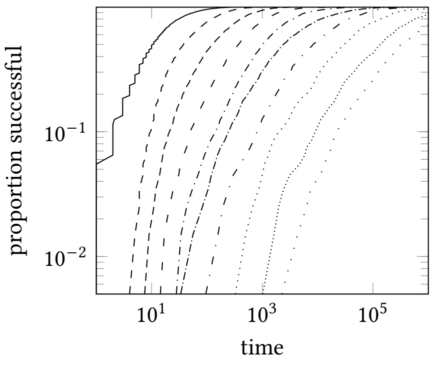
The running time as function of graph size exhibits much better scaling behavior with graph size than with deletion set size. Moreover, in Figures 5a and 5b, there is a clear separation by k for the different run times. This separation is not as pronounced in Cluster Deletion as can be observed in Figure 5c. In Figure 6, all algorithms appear to have a region near $n, m \approx 20$ which is somewhat difficult. This may arise from some constrainedness property of the graphs in this reason, and this phenomenon may warrant further investigation.

6 Conclusion

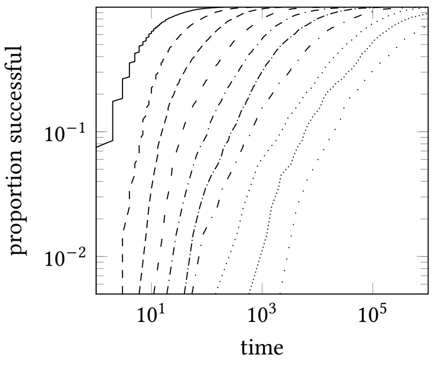
In this paper, we have adapted a population-based technique for solving component-selection problems on graphs to tackle data clustering problems from a graph theoretic perspective. This requires designing constraint repair operators for these problems that are able to efficiently identify deletion subsets that are disjoint from a given set.

To ensure tractability on cluster editing problems, we introduced a domain-specific template parent generation procedure that ensures uniform crossover has a reasonable chance to create a string that, when delivered to the constraint repair operator, produces a strictly improving offspring. We prove that the algorithm is a *randomized fixed-parameter tractable* algorithm in the sense that it has expected FPT runtime. For k-Cluster Deletion, we proved an upper bound of $O(n3^{4k} + mn^2 \log n)$, and for k-Cluster Vertex Deletion we proved an upper bound of $O(n3^{5k} + n^3 \log n)$. To our knowledge, this is the first runtime result for any evolutionary algorithm on these problems.

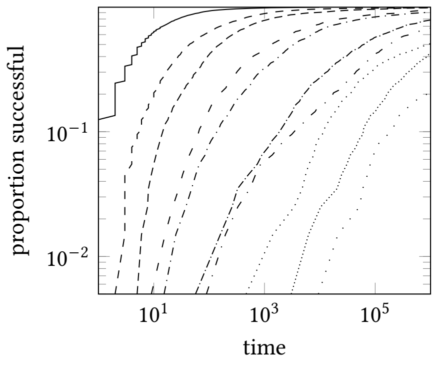
Finally, we have presented experimental results on a biological data set based on protein-protein interaction networks. These results show a strong run time dependence on k, with a milder dependence on graph size.



(a) CVD-Bio33 data set



(b) CVD-Bio66 data set



(c) CD-Bio33 data set

Figure 3: Cumulative success rate of SubPopGA for each distinct value of k: ($k = 1$ ——), ($k = 2$ - - -), ($k = 3$ - - - -), ($k = 4$ - -), ($k = 5$ -·-·-), ($k = 6$ -·-·-·), ($k = 7$ - · -), ($k = 8$ ······), ($k = 9$ ·········), ($k = 10$ · · · ·).

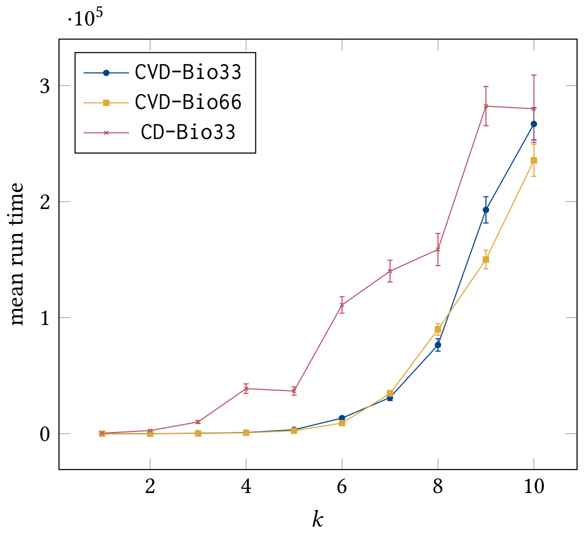


Figure 4: Mean (successful) runtime of SubPopGA as a function of k. Error bars denote standard error.

This paper provides insights into ways crossover can be leveraged to exploit structure in hard combinatorial optimization problems. Many directions for future work exist. The proved bounds have a quite heavy exponential dependence on k. Though they are unlikely to be tight, due to several pessimistic assumptions in the analysis, it would be interesting to see how much they could be improved, perhaps even obtaining lower bounds for the SubPopGA. From an algorithm engineering perspective, it might also be possible to improve different modules such as template parent generation to gain more efficiency on one or both problems. Finally, a remaining open question is how to adapt the approach to the more general problem of Cluster Editing, in which edge additions are also allowed.

Acknowledgments

This work was supported by NSF grant 2144080. Useful discussions leading up to this paper took place in the summer of 2024 at the Dagstuhl seminar 24271 *Theory of Randomized Optimization Heuristics.*

References

[1] Sebastian Böcker, Sebastian Briesemeister, Quang Bao Anh Bui, and Anke Truß. 2009. Going weighted: Parameterized algorithms for cluster editing. *Theoretical Computer Science* 410, 52 (2009), 5467–5480. doi:10.1016/J.TCS.2009.05.006

[2] Sebastian Böcker, Sebastian Briesemeister, and Gunnar W. Klau. 2011. Exact Algorithms for Cluster Editing: Evaluation and Experiments. *Algorithmica* 60, 2 (2011), 316–334. doi:10.1007/S00453-009-9339-7

[3] Luke Branson and Andrew M. Sutton. 2023. Focused jump-and-repair constraint handling for fixed-parameter tractable graph problems closed under induced subgraphs. *Theoretical Computer Science* 951 (2023), 113719. doi:10.1016/J.TCS.2023.113719

[4] Arantza Casillas, Mayte Teresa González de Lena, and Raquel Martínez. 2003. Document Clustering into an Unknown Number of Clusters Using a Genetic Algorithm. In *Text, Speech and Dialogue*, Václav Matoušek and Pavel Mautner (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 43–49. doi:10.1007/978-3-540-39398-6_7

[5] Ara Cho, Junha Shin, Sohyun Hwang, Chanyoung Kim, Hongseok Shim, Hyojin Kim, Hanhae Kim, and Insuk Lee. 2014. WormNet v3: a network-assisted hypothesis-generating server for Caenorhabditis elegans. *Nucleic acids research* 42, W1 (2014), W76–W82. https://doi.org/10.1093/nar/gku367

[6] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. 2015. *Parameterized Algorithms.* Springer. doi:10.1007/978-3-319-21275-3

[7] Anders Dessmark, Jesper Jansson, Andrzej Lingas, Eva-Marta Lundell, Mia Persson, and Gramm. 2007. On the Approximability of Maximum and Minimum Edge Clique Partition Problems. *International Journal of Foundations of Computer Science* 18, 2 (2007), 217–226. doi:10.1142/S0129054107004656

[8] Benjamin Doerr, Daniel Johannsen, and Carola Winzen. 2012. Multiplicative Drift Analysis. *Algorithmica* 64 (2012), 673–697. https://doi.org/10.1007/s00453-012-9622-x

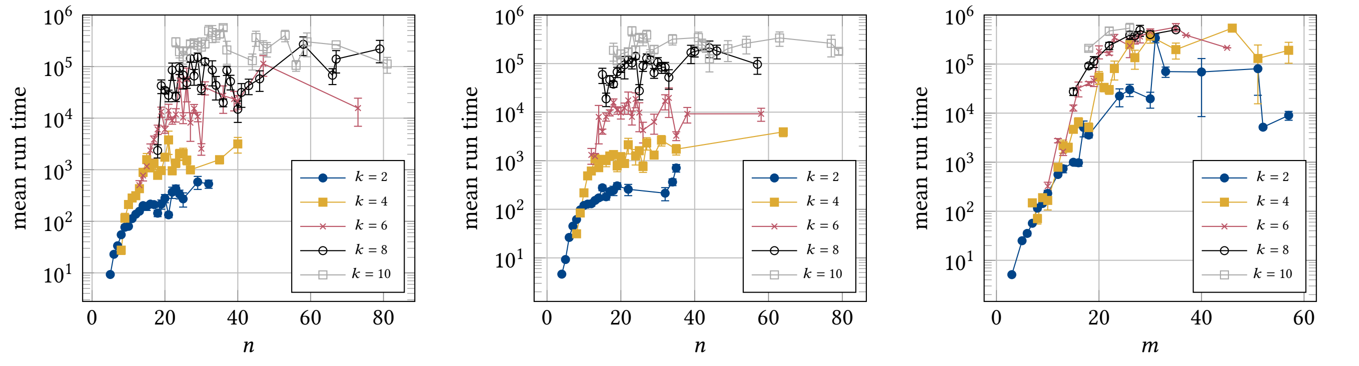
[9] Rodney G. Downey and Michael R. Fellows. 1999. *Parameterized Complexiy.* Springer. https://doi.org/10.1007/978-1-4612-0515-9

[10] Aleksander Figiel, Anne-Sophie Himmel, André Nichterlein, and Rolf Niedermeier. 2021. On 2-Clubs in Graph-Based Data Clustering: Theory and Algorithm Engineering. *Journal of Graph Algorithms and Applications* 25, 1 (2021), 521–547. doi:10.7155/JGAA.00570

[11] Andres Figueroa, James Borneman, and Tao Jiang. 2004. Clustering Binary Fingerprint Vectors with Missing Values for DNA Array Data Analysis. *Journal of Computational Biology* 11, 5 (2004), 887–901. doi:10.1089/CMB.2004.11.887

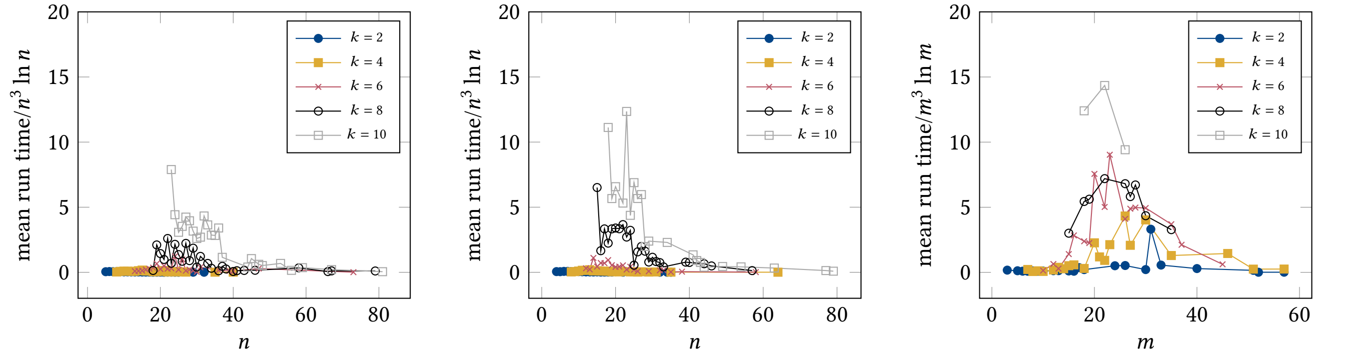
[12] Jörg Flum, Martin Grohe, and Gramm. 2006. *Parameterized complexity theory.* Springer. https://doi.org/10.1007/3-540-29953-X

[13] Jens Gramm, Jiong Guo, Falk Hüffner, and Rolf Niedermeier. 2005. Graph-Modeled Data Clustering: Exact Algorithms for Clique Generation. *Theory of Computing Systems* 38, 4 (2005), 373–392. doi:10.1007/S00224-004-1178-Y



(a) CVD-Bio33 data set **(b) CVD-Bio66 data set** **(c) CD-Bio33 data set**

Figure 5: Mean (successful) runtime of SubPopGA with respect to graph size. Error bars denote standard error.



(a) CVD-Bio33 data set **(b) CVD-Bio66 data set** **(c) CD-Bio33 data set**

Figure 6: Mean (successful) runtime (divided by $n^3 \ln n$) of SubPopGA solving k-Cluster Vertex Deletion and k-Cluster Deletion.

[14] Sepp Hartung and Holger H. Hoos. 2015. Programming by Optimisation Meets Parameterised Algorithmics: A Case Study for Cluster Editing. In *Learning and Intelligent Optimization - 9th International Conference, LION 9, Lille, France, January 12-15, 2015. Revised Selected Papers (Lecture Notes in Computer Science, Vol. 8994)*, Clarisse Dhaenens, Laetitia Jourdan, and Marie-Eléonore Marmion (Eds.). Springer, 43–58. doi:10.1007/978-3-319-19084-6_5

[15] Isidore Heller. 1957. On linear systems with integral valued solutions. *Pacific J. Math.* 7, 3 (1957), 1351–1364. https://doi.org/10.2140/pjm.1957.7.1351

[16] Falk Hüffner, Christian Komusiewicz, Hannes Moser, and Rolf Niedermeier. 2010. Fixed-Parameter Algorithms for Cluster Vertex Deletion. *Theory of Computing Systems* 47, 1 (2010), 196–217. doi:10.1007/S00224-008-9150-X

[17] Marco Laumanns, Lothar Thiele, and Eckart Zitzler. 2004. Running time analysis of multiobjective evolutionary algorithms on pseudo-Boolean functions. *IEEE Transactions on Evolutionary Computation* 8, 2 (2004), 170–182. doi:10.1109/TEVC.2004.823470

[18] Jiwon Lee and Andrew M. Sutton. 2024. Evolving Populations of Solved Subgraphs with Crossover and Constraint Repair. In *Parallel Problem Solving from Nature - PPSN XVIII - 18th International Conference, PPSN 2024, Hagenberg, Austria, September 14-18, 2024, Proceedings, Part III (Lecture Notes in Computer Science, Vol. 15150)*, Michael Affenzeller, Stephan M. Winkler, Anna V. Kononova, Heike Trautmann, Tea Tusar, Penousal Machado, and Thomas Bäck (Eds.). Springer, 133–148. doi:10.1007/978-3-031-70071-2_9

[19] John M. Lewis and Mihalis Yannakakis. 1980. The Node-Deletion Problem for Hereditary Properties is NP-Complete. *J. Comput. Syst. Sci.* 20, 2 (1980), 219–230. doi:10.1016/0022-0000(80)90060-4

[20] Carsten Lund and Mihalis Yannakakis. 1993. The approximation of maximum subgraph problems. In *Proceedings of the Twentieth International Colloquium on Automata, Languages and Programming (ICALP)*, Andrzej Lingas, Rolf Karlsson, and Svante Carlsson (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 40–51. https://doi.org/10.1007/3-540-56939-1_60

[21] Basilis Mamalis, Damianos Gavalas, Charalampos Konstantopoulos, and Grammati Pantziou. 2009. Clustering in Wireless Sensor Networks. In *RFID and Sensor Networks*, Yan Zhang, Laurence T. Yang, and Gramm (Eds.). CRC Press, 323–354. doi:10.1201/9781420077780

[22] Frank Neumann and Andrew M. Sutton. 2020. Parameterized Complexity Analysis of Randomized Search Heuristics. In *Theory of Evolutionary Computation: Recent Developments in Discrete Optimization*, Benjamin Doerr and Frank Neumann (Eds.). Springer International Publishing, 213–248. doi:10.1007/978-3-030-29414-4_4

[23] Peter Pipenbacher, Alexander Schliep, Sebastian Schneckener, Alexander Schönhuth, Dietmar Schomburg, and Rainer Schrader. 2002. ProClust: improved clustering of protein sequences with an extended graph-based approach. *Bioinformatics* 18, sup. 2 (10 2002), S182–S191. doi:10.1093/bioinformatics/18.suppl_2.S182

[24] Sven Rahmann, Tobias Wittkop, Jan Baumbach, Marcel Martin, Anke Truß, and Sebastian Böcker. 2007. Exact and Heuristic Algorithms for Weighted Cluster Editing. In *Computational Systems Bioinformatics*. World Scientific Publishing, 391–401. doi:10.1142/9781860948732_0040

[25] Hermes Robles-Berumen, Amelia Zafra, and Sebastián Ventura. 2024. A survey of genetic algorithms for clustering: Taxonomy and empirical analysis. *Swarm and Evolutionary Computation* 91 (2024), 101720. doi:10.1016/j.swevo.2024.101720

[26] Ryan A. Rossi and Nesreen K. Ahmed. 2015. The Network Data Repository with Interactive Graph Analytics and Visualization. In *AAAI*. https://networkrepository.com

[27] Ron Shamir, Roded Sharan, and Dekel Tsur. 2004. Cluster graph modification problems. *Discrete Applied Mathematics* 144, 1-2 (2004), 173–182. doi:10.1016/J.DAM.2004.01.007

[28] Roded Sharan and Ron Shamir. 2002. Algorithmic Approaches to Clustering Gene Expression Data. In *Current Topics in Computational Molecular Biology*, Tao

Jiang, Ying Xu, and Michael Q. Zhang (Eds.). The MIT Press, 269–299.

[29] Andrew M. Sutton. 2021. Fixed-Parameter Tractability of Crossover: Steady-State GAs on the Closest String Problem. *Algorithmica* 83, 4 (2021), 1138–1163. doi:10.1007/s00453-021-00809-8

[30] Roman L. Tatusov, Natalie D. Fedorova, John D. Jackson, Aviva R. Jacobs, Boris Kiryutin, Eugene V. Koonin, Dmitri M. Krylov, Raja Mazumder, Sergei L. Mekhedov, Anastasia N. Nikolskaya, B. Sridhar Rao, Sergei Smirnov, Alexander V. Sverdlov, Sona Vasudevan, Yuri I. Wolf, and Darren A. Yin, Jodie J.and Natale. 2003. The COG database: an updated version includes eukaryotes. *BMC Bioinformatics* 4, 1 (11 Sep 2003), 41. doi:10.1186/1471-2105-4-41

[31] Kangyi Tian, Mingyu Xiao, and Boting Yang. 2024. Parameterized Algorithms for Cluster Vertex Deletion on Degree-4 Graphs and General Graphs. In *Computing and Combinatorics*, Weili Wu and Guangmo Tong (Eds.). Springer Nature Switzerland, Cham, 182–194. https://doi.org/10.1007/978-3-031-49190-0_13

[32] Lin Yu Tseng and Shiueng Bien Yang. 2001. A genetic approach to the automatic clustering problem. *Pattern Recognition* 34, 2 (2001), 415–424. doi:10.1016/S0031-3203(00)00005-4

[33] Dekel Tsur. 2022. Cluster deletion revisited. *Inform. Process. Lett.* 173 (2022), 106171. https://doi.org/10.1016/j.ipl.2021.106171