

Linear Kernels in Linear Time, or How to Save k Colors in $O(n^2)$ Steps

Benny Chor¹, Mike Fellows², and David Juedes³

¹ School of Computer Science, Tel Aviv University, Tel Aviv, 69978, Israel
`benny@cs.tau.ac.il`

² School of EE & CS, University of Newcastle, Callaghan NSW 2308, Australia
`mfellows@cs.newcastle.edu.au`

³ School of EE & CS, Ohio University, Athens, Ohio 45701, USA
`juedes@ohio.edu`

Abstract. This paper examines a parameterized problem that we refer to as $n - k$ GRAPH COLORING, i.e., the problem of determining whether a graph G with n vertices can be colored using $n - k$ colors. As the main result of this paper, we show that there exists a $O(kn^2 + k^2 + 2^{3 \cdot 8161k}) = O(n^2)$ algorithm for $n - k$ GRAPH COLORING for each fixed k . The core technique behind this new parameterized algorithm is kernalization via maximum (and certain maximal) matchings.

The core technical content of this paper is a near linear-time kernelization algorithm for $n - k$ CLIQUE COVERING. The near linear-time kernelization algorithm that we present for $n - k$ CLIQUE COVERING produces a linear size $(3k - 3)$ kernel in $O(k(n + m))$ steps on graphs with n vertices and m edges. The algorithm takes an instance $\langle G, k \rangle$ of CLIQUE COVERING that asks whether a graph G can be covered using $|V| - k$ cliques and reduces it to the problem of determining whether a graph $G' = (V', E')$ of size $\leq 3k - 3$ can be covered using $|V'| - k'$ cliques. We also present a similar near linear-time algorithm that produces a $3k$ kernel for VERTEX COVER. This second kernelization algorithm is the *crown reduction rule*.

1 Introduction

Graph coloring is one of the hardest NP-complete problems under a variety of measures. It is well-known [1] that determining whether a graph can be colored using $k = 3$ colors is NP-complete, and the best-known polynomial-time algorithm for graph coloring, due to Halldórsson [2], produces a coloring that is only guaranteed to be within $O(|V|(\log \log |V|)^2 / \log^3 |V|)$ of optimal. It is known that approximating $\chi(G)$, the *chromatic number* of a graph G , to the ratio $|V|^\delta$ is NP-hard [3, 4]. Furthermore, it is known from the work of Feige and Kilian [5] that no polynomial-time algorithm can approximate $\chi(G)$ to within $|V|^{1-\epsilon}$ for any $\epsilon > 0$ unless $\text{NP} \subseteq \text{ZPP}$. Hence, it does not appear likely that we will be able to find a much better approximation algorithm for graph coloring.

This extended abstract explores graph coloring from another, more tractable, perspective. Since it is easy to see that any graph $G = (V, E)$ can be colored

via a trivial coloring that uses n colors and colors each vertex using a separate color, the alternative perspective is to ask whether G can be colored using $n - k$ colors. We refer to this problem as $n - k$ GRAPH COLORING. The parameter k corresponds to the how many colors can be “saved” when coloring a graph $G = (V, E)$ over the trivial coloring of G . It is easy to see that this problem is NP-complete. However, in contrast to the usual optimization measure for graph coloring, this version is much easier to approximate. A sequence of papers by Demange, Grisoni and Paschos [6], Hassin and Lahav [7], and Halldórsson [8, 9], and Duh and Fürer [10] examined approximation algorithms for graph coloring under this non-standard measure. The best-known polynomial-time approximation algorithm, due to Duh and Fürer [10], approximates $n - k$ GRAPH COLORING to the ratio $\frac{360}{280} \approx 1.246$.

This paper explores exact algorithms for $n - k$ GRAPH COLORING. In particular, we show that it is possible to determine whether a graph G can be colored using $n - k$ colors in $O(n^2)$ steps for any fixed k . Hence, $n - k$ GRAPH COLORING is *fixed parameter tractable* [11]. Our algorithm exploits the following, well-known result concerning the relationship between graph coloring and clique covering, $\chi(G) = \bar{\chi}(\bar{G})$, i.e., that the minimum number of colors needed to color G is equal to the minimum number of cliques needed to cover the complement of G . The main technical contribution of this work is a linear-time algorithm that kernelizes instances of $n - k$ CLIQUE COVERING, i.e., the problem of determining whether a graph G with n vertices can be covered by $n - k$ cliques. The general kernelization approach proceeds as follows.

1. Compute a maximal matching M with no M augmenting path of length 3 or shorter on the input graph G .
2. Identify an independent set I of vertices in the reduced input graph by examining the maximal matching M .
3. Compute a maximum matching M' in the bipartite graph formed by I its edges to the rest of the graph.
4. Eliminate all vertices not covered by either the first or second matching.

This general kernelization approach can be used to achieve a kernel of size $\leq 3k - 3$ for $n - k$ CLIQUE COVERING. We use a similar approach to give a *crown reduction rule* for VERTEX COVER that achieves a $\leq 3k$ kernel.

Once a graph G is kernelized, determining whether G has a clique covering of size $n - k$ can be solved in $f(k) = O(k^2 + 2^{3 \cdot 8161k})$ steps by first converting the kernelized instance $\langle G', k \rangle$ into its complement $\langle \bar{G}', k \rangle$ and then applying the best-known exact algorithm for graph coloring due to Eppstein [12]. This approach leads to parameterized algorithms running in time $O(k(n + m) + k^2 + 2^{3 \cdot 8161k})$ for $n - k$ CLIQUE COVERING and time $O(k \cdot n^2 + k^2 + 2^{3 \cdot 8161k})$ for $n - k$ GRAPH COLORING. The algorithm for $n - k$ GRAPH COLORING is depicted in Figure 1.

We note that the use of maximum matchings in fixed parameter tractable algorithms is not new. For instance, Papadimitriou and Yannakakis [13] used a maximum matching based approach to show that VERTEX COVER can be solved in polynomial-time for all $k \leq \log n$. Their work implicitly gives a $O(3^k n)$

parameterized algorithm for VERTEX COVER. Similarly, the best-known parameterized algorithm for VERTEX COVER [14] constructs a $2k$ kernel for VERTEX COVER using a maximum matching based technique of Nemhauser and Trotter [15]. However, our kernelization technique appears to be more general since it is easily applied to both VERTEX COVER and $n - k$ CLIQUE COVERING. Moreover, this approach achieves linear kernels in near linear-time.

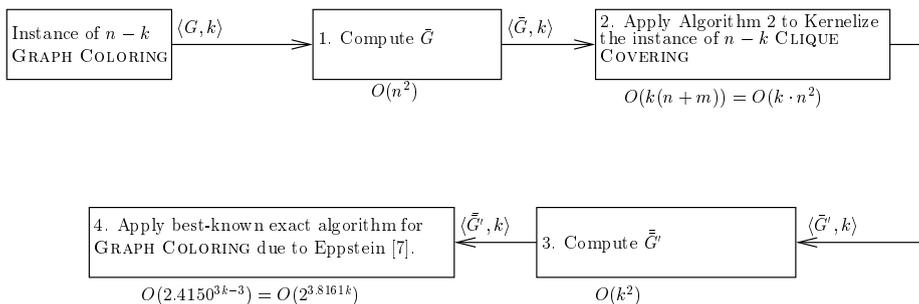


Fig. 1. A $O(kn^2 + k^2 + 2^{3.8161k})$ algorithm for $n - k$ GRAPH COLORING that uses the identity $\chi(G) = \bar{\chi}(\bar{G})$ twice.

The extended abstract is organized as follows. In section 2, we review necessary notation and prior work on maximum and maximal matchings. In section 3, we examine the combinatorial properties of maximal matchings without short augmenting paths. In section 4, we give the kernelization algorithm for $n - k$ CLIQUE COVERING and prove that the algorithm converts an instance of $\langle G, k \rangle$ to an equivalent instance $\langle G', k \rangle$ of size $\leq 3k - 3$. In section 5, we use the same technique to give a *crown reduction rule* for VERTEX COVER that produces a $\leq 3k$ kernel. In section 6, we use the kernelization algorithm and the best-known exact algorithm for graph coloring by Eppstein [12] to give fast parameterized algorithms for $n - k$ CLIQUE COVERING and $n - k$ GRAPH COLORING that run in the times mentioned above. Finally, in section 7, we provide some concluding remarks.

2 Preliminaries

In this section, we review the notation and the relevant prior work on maximum matchings that we use in this paper. Interested readers are directed to the survey paper by Galil [16] or the comprehensive book by Lovász and Plummer [17] for further details.

Given a graph $G = (V, E)$, a *matching* is a set of edges $M \subseteq E$ such that no two edges in M share an endpoint. A vertex $v \in V$ is said to be *covered* if it is an endpoint of an edge in M . If v is not covered, then we say that v is *exposed*.

Most algorithms that find maximum matchings do so by explicitly constructing *augmenting paths* [16]. Given a matching M , an M -*alternating path* is a sim-

ple path $P = \langle v_1, v_2, \dots, v_n \rangle$ in G such that P consists of edges that alternate between edges in M and edges outside of M . An M -augmenting path is an M -alternating path that begins and ends with an exposed vertex. It is well-known that the existence of an M -augmenting path P implies that M is not a maximum matching since it is possible to create a larger matching by swapping the unmatched edges in P for the matched edges in P to create a larger matching. This observation leads to the following theorem, due to Berge [18].

Theorem 1. *M is a maximum matching in G if and only if there is no M -augmenting path in G .*

Fast algorithms to compute maximum matchings were first discovered for bipartite graphs. The best-known algorithm for maximum matching in bipartite graphs is due to Hopcroft and Karp [19] and takes time $O(m\sqrt{n})$ on graphs with m edges and n vertices. The best-known algorithm for maximum matching in general graphs comes from the work of Micali and Vazirani [20] and also takes time $O(m\sqrt{n})$.

To achieve linear-time kernelization here, we will employ Hopcroft and Karp's algorithm on bipartite graphs and a somewhat straightforward algorithm for finding maximal matchings without short augmenting paths in general graphs. To see that Hopcroft and Karp's algorithm runs in linear-time in our case, we need to briefly describe the operation of this algorithm. The algorithm by Hopcroft and Karp employs an $O(m)$ algorithm to find a maximal set of vertex disjoint augmenting paths of minimum length. The algorithm proceeds in multiple passes where at each pass the algorithm augments the matching M via the maximal set of vertex disjoint M -augmenting paths of minimum length. This process continues until no M -augmenting path is found, and hence by Theorem 1 M is a maximum matching in G . The algorithm executes at most $O(\sqrt{n})$ passes because, as Hopcroft and Karp [19] prove in Theorem 2 and Corollaries 3 and 4, once a maximal set of vertex disjoint M augmenting paths of length l is found and M is augmented to M' , the shortest M' -augmenting path in the graph has length $l + 1$ or longer.

In the bipartite graphs that we use here, one of the two partitions will be of size $2k$ or smaller. Since no M augmenting path in such a graph can be of length longer than $4k$ edges, the algorithm of Hopcroft and Karp performs at most $4k$ augmenting passes. Therefore, Hopcroft and Karp's algorithm will complete in $O(km)$ steps.

The ideas behind Hopcroft and Karp's algorithm can be used to find maximal matchings in general graphs with no augmenting path of length 3 in $O(n + m)$ steps. To see this, consider the following approach. First, compute a maximal matching M in G using the standard $O(n + m)$ greedy algorithm. Next, compute a maximal set S of vertex disjoint M -augmenting paths of length 3 in G by examining each edge $e \in M$. For each edge $e = (u, v) \in M$, compute c_u , the number of exposed vertices connected to u and c_v , the number of exposed vertices connected to v . If either c_u or c_v equals zero, no augmenting path of length 3 that uses e exists. If $c_u = c_v = 1$, then a path exists if and only if these vertices are different. Finally, if $c_u \geq 2$ and $c_v \geq 1$ (or vice-versa), then an augmenting path

through e must exist and can be found easily. Mark the endpoints of any such found augmenting path as “covered.” It is easy to see that this approach finds a maximal set of vertex disjoint augmenting paths of length 3 in time $O(n + m)$ since each edge in G is visited at most $O(1)$ times. Finally, the results of Hopcroft and Karp tell us that augmenting the matching M with S produces a matching M' with no augmenting path of length 3. Such matchings have properties that we exploit in the next section.

3 Maximal Matchings Without Short Augmenting Paths and Their Combinatorial Properties

Given a maximal matching M in a graph $G = (V, E)$, it is natural to partition V into the *covered* vertices and the *exposed* vertices. In this fashion, we partition V into $I_M = \{v \in V | \{u, v\} \in M\}$ and $O_M = V - I_M$. The set O_M forms an independent set because if there were an edge between any pair of vertices in O_M , then M would not be a maximal matching.

Given a matching M with no augmenting path of length 3, we partition the matched edges into classes to exploit certain combinatorial properties. In this respect, we partition M into three classes C_1, C_2 , and C_3 based on how the endpoints of the edges in M connect to the exposed vertices. Figures 2–4 illustrate the three classes of matched edges.

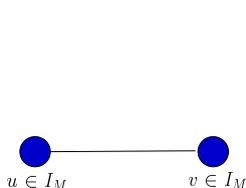


Fig. 2. The class 1 “boring” edges.

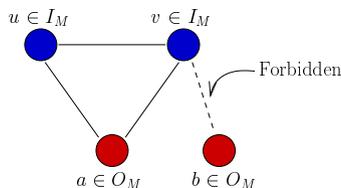


Fig. 3. The class 2 edges.

We refer to the first class of matched edges C_1 as the “boring edges” since they are not connected to any exposed vertex. In the second class of matched edges C_2 , both endpoints are connected to vertices in O_M . As shown in Figure 3, both endpoints must be connected to the same vertex $a \in O_M$, or there will exist an M -augmenting path in G of length 3. Finally, the third class of edges in M , C_3 , contains edges where only one of the two endpoints are connected to exposed vertices. This case is shown in Figure 4.

The main combinatorial property of maximal matchings M with no augmenting paths of length 3 that we use in this paper is the following lemma that relates the size of M to the size of the maximum matching M' in the bipartite graph formed by letting $V_1 = O_M$, $V_2 = N(O_M)$, and only including edges from O_M to $N(O_M)$ in G . We call this graph $G[O_M, N(O_M)]$.

Lemma 1. *Let M be a maximal matching in G with no length 3 M -augmenting path. Let M' be a maximum matching in $G[O_M, N(O_M)]$. Then $|M'| \leq |M|$.*

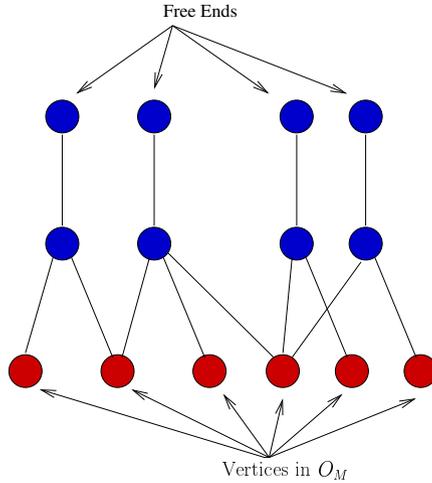


Fig. 4. The class 3 edges.

Proof. The bipartite graph $G' = G[O_M, N(O_M)]$ contains edges from exposed vertices to the end points of matched edges in M . It is clear by inspection that no more than one endpoint of matched edges in M can appear as an endpoint of a matched edge in G' since (i) for each edge in C_3 only one endpoint is connected to any vertex in O_M , and (ii) for each edge in C_2 , both endpoints are connected to a single vertex a in O_M and hence at most one of these endpoints can be matched to a . It follows that $|M'| \leq |M|$.

4 A Kernelization Algorithm for Clique Covering

In this section, we develop a kernelization algorithm for CLIQUE COVERING that exploits the combinatorial properties of maximal matchings developed in the previous section. The correctness of our approach depends on the following generic results concerning independent sets, maximum matchings, and clique coverings.

In the following lemma, we call a clique “trivial” if it contains only a single vertex.

Lemma 2. *Let $G = (V, E)$ be a graph, and let I be an independent set of vertices in G . If M' is a maximum matching in $G[I, V - I]$, then in any clique covering C of G at most $|M'|$ vertices in I appear in non-trivial cliques in C .*

Proof. Assume that there are k vertices in I that appear in non-trivial cliques in C . Since I is an independent set, no two of these vertices can appear in the same clique. So, let C_1, \dots, C_k be the k cliques that contain these vertices. For the vertex $v_i \in I$ that appears in C_i , pick another vertex $u_i \in (V - I) \cap C_i$. Such a vertex must exist because C_i is non-trivial and I is an independent

set. Furthermore, the edges $\{u_1, v_1\}, \{u_2, v_2\}, \dots, \{u_k, v_k\}$ form a matching in $G[I, V - I]$ because the C_i 's are disjoint. Since M' is a maximum matching, it follows that $k \leq |M'|$.

Lemma 3. *Let $G = (V, E)$ be a graph, let I be an independent set in G , and let M' be a maximum matching in $G[I, V - I]$. For every clique covering C of G , there exists a clique covering C' of G such that $|C'| \leq |C|$ and each vertex in $M'[I]$ (those vertices in I that are covered by M') appears in a non-trivial clique in C' .*

Proof. Let $C = \{C_1, \dots, C_m\}$ be a clique covering of G . Examine each vertex $v_i \in M'[I]$. Let u_i be the other endpoint of the matched edge containing v_i . If v_i is contained in a trivial clique, delete u_i from its clique (it is still a clique, or we have deleted a clique), and add it to v_i 's clique. This does not increase the total number of cliques. Continue this process until all of the vertices $v_i \in M'[I]$ are contained in non-trivial cliques. Notice that this process may create trivial cliques of vertices in $M'[I]$. However, this process must eventually halt, since we increase the total number of vertices in $M'[I]$ that appear in non-trivial cliques with their paired vertex during each iteration. Moreover, these new cliques cannot be “destroyed.” This approach is given in Algorithm 1. To prove the correctness of this algorithm, it suffices to show that $c = |\{v_i \in M'[I] \mid \{u_i, v_i\} \in M' \text{ and } u_i, v_i \in C_j \text{ for some } C_j \in C\}|$ is a loop invariant for the algorithm. The algorithm will halt since c increases by one during each iteration of the loop. When the loop halts, the value of c must be $|M'[I]|$, and hence every vertex in $M'[I]$ appears in a non-trivial clique in C' . The algorithm creates no new cliques (though it may destroy some trivial cliques), hence the total number of cliques in C' is less than or equal to $|C|$.

Combining Lemma 2 and 3, we observe that if I is an independent set and M' is a maximum matching in the bipartite graph $G[I, V - I]$, then there is an optimal clique covering where all of the vertices in I that are not covered by M' appear in trivial cliques. Hence, these vertices can be deleted without affecting the relative size of the optimal clique covering. So, if we let G' be the graph

Algorithm 1 Creating Non-trivial Cliques

Input: A Clique Covering C of G

Output: A Clique Covering C' of G such that $|C'| \leq |C|$ and each vertex in $M'[I]$ is in a non-trivial clique.

set $c = |\{v_i \in M'[I] \mid \{u_i, v_i\} \in M' \text{ and } u_i, v_i \in C_j \text{ for some } C_j \in C\}|$;

while (there exists a $v_i \in M'[I]$ in a trivial clique C_i) **do**

 Find $\{u_i, v_i\} \in M'$.

 Find the C_k such that $u_i \in C_k$;

 Remove u_i from C_k ;

 add u_i to C_i ;

$c = c + 1$;

end while

return C' ;

formed by removing all of the vertices in I that are not covered by M' , then G has a clique covering of size $n - k$ if and only if G' has a clique covering of size $n' - k$. This approach forms the basis for the main technical contribution of this paper, the kernelization algorithm for $n - k$ CLIQUE COVERING given in the following theorem.

Theorem 2. *There exists an algorithm running in $O(k(m+n))$ steps that, when given an instance $\langle G, k \rangle$ of $n - k$ CLIQUE COVERING with n vertices and m edges either determines that G can be covered using $n - k$ cliques or produces an equivalent instance $\langle G', k \rangle$, where G' has $\leq 3k - 3$ vertices.*

Proof. We present Algorithm 2 below. To see that the algorithm runs in $O(k(m+n))$ steps, consider the running time of each step. Step 1 completes in $O(m+n)$ steps, as described at the end of section 2. Steps 2–6 can also be computed in $O(m+n)$ steps or faster. As explained at the end of section 2, step 7 takes $O(k(m+n))$ steps because $N(O_M)$ has at most $2k$ vertices, and hence the maximum bipartite matching algorithm of Hopcroft and Karp performs at most $4k$ passes. The final two steps can also be implemented in $O(m+n)$ steps.

Algorithm 2 Kernelization for $n - k$ CLIQUE COVERING.

Input: An instant $\langle G, k \rangle$ of $n - k$ CLIQUE COVERING

Output: Either “yes” if G has an $|V| - k$ clique covering or an equivalent instance $\langle G', k \rangle$ of $n - k$ CLIQUE COVERING

- 1: Compute a maximal matching M in G with no M augmenting path of length 3 as described in section 2.
 - 2: **if** $|M| \geq k$ **then**
 - 3: return “yes” since G has a $n - k$ clique covering consisting of k cliques of size 2 and $n - 2k$ trivial cliques.
 - 4: **end if**
 - 5: Partition V into covered (I_M) and exposed (O_M) vertices.
 - 6: Compute the bipartite graph $G^* = G[I_M, N(O_M)]$.
 - 7: Apply the algorithm of Hopcroft and Karp [19] to compute a maximum matching M' of G^* .
 - 8: Delete from G all vertices in O_M that are not covered by M' . Call this graph G' .
 - 9: Return $\langle G', k \rangle$.
-

To see that this algorithm produces the correct result, consider the following brief argument. Since O_M is an independent set and Algorithm 2 deletes from G all vertices in O_M that are not covered by M' , it is clear from Lemma 2 and 3 that G' has a clique covering of size $n' - k$ if and only if the original instance G has a clique covering of size $n - k$. To see that G' has the correct size, we employ results from the previous section.

Since $|O_M| = n - 2|M|$, we delete exactly $n - 2|M| - |M'|$ vertices from G to form G' in step 8 of the algorithm. Since $|M'| \leq |M|$ by Lemma 1, it follows that we delete at least $n - 3|M|$ vertices from G . Since G has n vertices to begin with, this leaves at most $3|M|$ vertices in G' . Since $|M| < k$, it follows that G' has at most $3k - 3$ vertices. This completes the proof.

5 The Crown Reduction Rule for Vertex Cover

To show the generality of the kernelization approach taken in the previous section, we show how to apply this technique to give a linear-time kernelization algorithm for instances of VERTEX COVER. The key to our kernelization for VERTEX COVER is the following observation. Let I be an independent set in G , and assume that there is a matching of size $|N(I)|$ in the bipartite graph $G[I, N(I)]$ formed by I , the neighborhood of I , and the edges between them in G . Under this assumption, at least $|N(I)|$ vertices from $I \cup N(I)$ must appear in any vertex cover of G . Hence, it suffices to include all of $N(I)$ in the vertex cover and delete all of $I \cup N(I)$ from G . This is the *crown reduction rule*. Here, we show how to efficiently find such an I via a maximum and a maximal matching.

Let I be an independent set of vertices from a graph $G = (V, E)$. Then, the *bipartite subgraph of G formed by the sets I and $V - I$* , denoted by $G[I, V - I]$, is the graph $G' = (V, E')$, where $E' = \{\{u, v\} \mid u \in I, v \in V - I, \text{ and } \{u, v\} \in E\}$. Let M be a maximum matching in $G[I, V - I]$. Then, we write $M[I]$ for the set of vertices in I that are covered by edges in M and $M[V - I]$ for the set of vertices in $V - I$ that are covered by edges in M . If S is a set of endpoints of edges in M , then we write $M_{-}[S]$ for the set of vertices matching the vertices in S , i.e.,

$$M_{-}[S] = \{u \mid \{u, v\} \in M \text{ and } v \in S\}.$$

As in section 4, we show that it is possible to eliminate vertices in I that do not appear in $M[I]$. As mentioned above, we can eliminate all of the vertices in I when M is an *upper perfect matching*, i.e., when $|M| = |N(I)|$, or, in other words, all of the vertices on one side of the bipartite graph are matched. While a maximum matching in $G[I, V - I]$ for an arbitrary independent set I may not be an upper perfect matching, we show below that there must exist an upper perfect matching for a set I' , where $I - M[I] \subseteq I' \subseteq I$. This requires some explanation.

Let I be an independent set in G , and let M be a maximum matching in the bipartite subgraph $G[I, V - I]$. Then, it is possible to select a subset of $M[V - I]$ that has an upper perfect matching with a subset of the vertices in I . To begin, we note the following easy result.

Lemma 4. $N(I - M[I]) \subseteq M[V - I]$, i.e., *vertices in I that are not covered by the matching must connect to only endpoints in the matching.*

Proof. Assume that a vertex $v \in I - M[I]$ connects to a vertex $u \in N(I) - M[V - I]$. Then, we can add the edge $\{u, v\}$ to the matching. Hence, the matching is not maximal (and hence not a maximum matching). This is a contradiction.

We next define a collection of subsets of $M[V - I]$.

$$\begin{aligned} C^0(M) &= N(I - M[I]) \\ C^i(M) &= N(M_{-}[C^{i-1}(M)]) \\ C(M) &= \bigcup_{i=0}^{\infty} C^i(M). \end{aligned}$$

The following results hold.

Lemma 5. 1. For each i , $C^i(M) \subseteq M[V - I]$.

2. For each $i > 0$, $C^{i-1}(M) \subseteq C^i(M)$.

3. For each i and for each vertex $v \in C^i(M)$, there exists an M -alternating path from a vertex $u \in I - M[I]$ to v .

Proof. (By induction on i) When $i = 0$, Lemma 4 tells us that $C^0(M) \subseteq M[V - I]$. Furthermore, each vertex $v \in C^0(M)$ is attached to a vertex $u \in I - M[I]$. This single edge is an M -alternating path from u to v .

Now, assume that conditions 1–3 are true for some $i \geq 0$. To see that condition (2) is true for $C^{i+1}(M)$, note that every vertex in $C^i(M)$ is an endpoint in the matching. Hence, $C^i(M) \subseteq N(M_-[C^i(M)]) = C^{i+1}(M)$. To see that condition (3) is true, assume that $v \in C^{i+1}(M) - C^i(M)$. Hence, there exists a vertex $u \in C^i(M)$ such that $\{u, v_1\} \in M$, and $v \in N(v_1)$. Since $v \in C^{i+1}(M) - C^i(M)$, it is clear that $\{v, v_1\} \notin M$. Moreover, since there exists an M alternating path from a vertex $v_2 \in I - M[I]$ to u (by the inductive hypothesis), adding the edges $\{u, v_1\}$, $\{v_1, v\}$ forms an M alternating path from v_2 to v . Finally, to see that condition (1) is true, notice that if $v \notin M[V - I]$, then the M alternating path from v_2 to v is an M augmenting path. Hence, the original matching M is not a maximum matching.

As we show above, the $C^i(M)$'s are ordered by the subset relation. As we show next, there must be a fixed point in these sets.

Lemma 6. If $C^i(M) = C^{i+1}(M)$, then $C^i(M) = C^{i+k}(M)$ for all $k \geq 1$.

Proof. (By induction on k .) When $k = 1$, the lemma holds trivially. Now, assume that the lemma holds when $k \geq 1$. Then,

$$C^{i+k+1}(M) = N(M_-[C^{i+k}(M)]) = N(M_-[C^i(M)]) = C^{i+1}(M) = C^i(M).$$

Lemma 5 and 6 imply a straightforward method for computing $C(M)$; compute $C^i(M)$ until $C^{i+1}(M) = C^i(M)$. The end result of this computation is $C(M)$. This computation must complete after $|M|$ passes, since $|C(M)| \leq |M|$. Moreover, notice that computing $C^{i+1}(M)$ from $C^i(M)$ can be done in $O(n+m)$ steps since it simply involves computing neighborhoods. Hence, the entire computation of $C(M)$ takes $O(|M|(n+m))$ steps, where $n = |V|$ and $m = |E|$.

Now, we can give a subset of $G[I, V - I]$ that has an upper perfect matching. Define

$$I' = M_-[C(M)] \cup (I - M[I]).$$

Lemma 7. $N(I') = C(M)$.

Proof. $N(I') = N(I - M[I]) \cup N(M_-[C(M)]) = C^0(M) \cup C(M) = C(M)$.

Lemma 8. The bipartite graph $G[I', V - I']$ has an upper perfect matching.

Proof. Lemma 7 tells us that $N(I') = C(M)$. Since I' contains all of the vertices in $M_-[C(M)]$, the edges from $C(M)$ to $M_-[C(M)]$ form an upper perfect matching of $G[I', V - I']$.

We now show that we can eliminate all of the vertices in I from G (and perhaps some others). The single reduction rule that we use is the following.

Crown Reduction Rule: Given an independent set I in G , if the bipartite graph $G[I, V - I]$ has an upper perfect matching (i.e., a matching of size $|N(I)|$), then delete I and $N(I)$ from G to form G' and set $k' = k - |N(I)|$.

The crown reduction rule is a generalization of the reduction rule that can be used to eliminate degree 1 vertices for VERTEX COVER.

Lemma 9. G' has a vertex cover of size k' iff G has a vertex cover of size k .

Proof. Assume that G' has a vertex cover V^* of size k' . Then, $V^* \cup N(I)$ is a vertex cover of G of size $k' + |N(I)| = k$. To see this, notice that the vertices in $N(I)$ covers the edges in the bipartite subgraph formed by I and $N(I)$. Moreover, the vertices in $N(I)$ also cover all edges from $N(I)$ to the rest of G .

Similarly, assume that G has a vertex cover V^* of size k and define $\hat{V}^* = V^* - (N(I) \cup I)$. This is a vertex cover of G' . Each matched edge connecting the vertices in $N(I)$ to the vertices in I must be covered by either a unique vertex in $N(I)$ or a unique vertex in I . Since I is an independent set, these sets are disjoint. Therefore, $|\hat{V}^*| \leq |V^*| - |N(I)| = k - |N(I)| = k'$.

The crown reduction rule leads to the following kernelization algorithm for VERTEX COVER.

Theorem 3. *There exists an algorithm running in time $O(k(n + m))$ steps that takes an instance $\langle G, k \rangle$ of VERTEX COVER with n vertices and m edges and either returns “no” (G has no vertex cover of size k) or produces an equivalent instance $\langle G', k' \rangle$, where $k' \leq k$ and $|V'| \leq 3k$.*

Proof. The algorithm proceeds as follows.

Algorithm 3 Linear-Time Kernelization Algorithm for VERTEX COVER.

- 1: Compute a maximal matching M in G with no M -augmenting path of length 3 using the algorithm described in section 2.
 - 2: **if** $|M| > k$ **then**
 - 3: return “no,” because G needs at least $k + 1$ vertices in any vertex cover.
 - 4: **end if**
 - 5: Otherwise, partition V into I_M and O_M .
 - 6: Compute the bipartite graph $G^* = G[O_M, N(O_M)]$.
 - 7: Compute a maximum matching M' in the bipartite graph G^* .
 - 8: Compute the crown $C(M')$, and let $I' = M'_\perp[C(M')] \cup (O_M - M'[O_M])$.
 - 9: Apply the crown reduction rule. This rule deletes $C(M')$ and I' from G .
 - 10: return $\langle G', k' \rangle$.
-

Notice that the correctness of this algorithm follows almost immediately from Lemma 9. To see that G' has $\leq 3k$ vertices, notice that (i) after step 3, we know that $|M| \leq k$, (ii) Lemma 1 tells us that $|M'| \leq |M|$, and (iii) the crown reduction rule removes at least $|O_M| - |M'| \geq |O_M| - |M|$ vertices from G . Since $n = |O_M| + 2|M|$ and $|M| \leq k$, $|V'| \leq n - |O_M| - |M| = 3|M| \leq 3k$.

To see that Algorithm 3 runs in time $O(k(m + n))$, notice that (i) step 1 takes $O(m + n)$ steps, (ii) steps 2–6 can be performed in $O(n + m)$ steps, (iii)

step 7 takes $O(k(n+m))$ steps since $|N(O_M)| \leq 2k$ because $|M| \leq k$, (iv) step 8 takes $O(k(n+m))$ steps by the argument prior to Lemma 7, and (v) steps 9 and 10 can be performed in $O(n+m)$ steps. This completes the proof.

6 Parameterized Algorithms for Graph Coloring and Clique Covering

In this section, we combine the results of sections 2, 3, and 4 to prove our main result.

Theorem 4. $n-k$ CLIQUE COVERING can be solved in time $O(k(n+m) + k^2 + 2^{3.8161k})$ steps on instances $\langle G, k \rangle$, where G has m edges and n vertices.

Proof. Given an instance $\langle G, k \rangle$ of $n-k$ CLIQUE COVERING, apply Algorithm 2 from section 4. The algorithm either determines that there is a clique covering of size $n-k$ of G , or it kernelizes $\langle G, k \rangle$ to an equivalent instance $\langle G', k \rangle$ such that G' has a clique covering of size $n'-k$ if and only if G has a clique covering of size $n-k$. Moreover, we have that $n' \leq 3k-3$. This takes $O(k(m+n))$ steps. Next, apply the relationship between graph coloring and clique covering, and convert $\langle G', k \rangle$ into an equivalent instance $\langle G', k \rangle$ of $n-k$ GRAPH COLORING. This takes $O(k^2)$ steps. Finally, apply the best-known exact algorithm from graph coloring, due to Eppstein [12]. This algorithm runs in $O(2.4150^n) = O(2.4150^{3k-3}) = O(2^{3.8161k})$ steps. This completes the algorithm and the proof.

Theorem 5. $n-k$ GRAPH COLORING can be solved in time $O(kn^2 + k^2 + 2^{3.8161k})$ on instances $\langle G, k \rangle$, where G has m edges and n vertices.

Proof. It suffices to employ the identity $\chi(G) = \bar{\chi}(\bar{G})$. Hence, G can be colored using $n-k$ colors if and only if \bar{G} can be covered using $n-k$ cliques. Therefore, it suffices to first compute \bar{G} and to then employ the algorithm given in Theorem 4. The result follows.

7 Concluding Remarks

This paper provides an alternate approach to exact algorithms for graph coloring. The approach provided here can be used to design faster exact algorithms for graphs with high chromatic numbers ($\chi(G) \geq 2/3n$) because the parameterized algorithm given here runs faster than the $O(2.4150^n)$ exact algorithm when $k \leq 1/3n$. Perhaps this new parameterized algorithm can be used in some inventive way to improve the running-time of the exact algorithm for graph coloring for all graphs.

Acknowledgments

The authors would like to thank Christian Sloper, Dan Xiao, and David Fleeman for their helpful comments on earlier drafts of this paper.

References

1. Garey, M.R., Johnson, D.S.: Computers and Intractability: a guide to the theory of NP-completeness. W.H. Freeman (1979)
2. Halldórsson, M.M.: A still better performance guarantee for approximate graph coloring. *Information Processing Letters* **45** (1993) 19–23
3. Lund, C., Yannakakis, M.: On the hardness of approximating minimization problems. *Journal of the ACM* **41** (1994) 960–981
4. Bellare, M., Goldreich, O., Sudan, M.: Free bits, PCPs and non-approximability – towards tight results. *SIAM Journal on Computing* **27** (1998) 804–915
5. Feige, U., Kilian, J.: Zero knowledge and the chromatic number. In: *Proceedings of the Eleventh Annual IEEE Conference on Computational Complexity*, IEEE Computer Society Press (1996) 278–289
6. Demange, M., Grisoni, P., Paschos, V.T.: Approximation results for the minimum graph coloring problem. *Information Processing Letters* **50** (1994) 19–23
7. Hassin, R., Lahav, S.: Maximizing the number of unused colors in the vertex coloring problem. *Information Processing Letters* **52** (1994) 87–90
8. Halldórsson, M.M.: Approximating discrete collections via local improvement. In: *Proceedings of the Sixth ACM-SIAM Symposium on Discrete Algorithms*, ACM Press (1995) 160–169
9. Halldórsson, M.M.: Approximating k -set cover and complementary graph coloring. In: *Proceedings of the 5th IPCO Conference on Integer Programming and Combinatorial Optimization*. Volume 1084 of LNCS., Springer-Verlag (1996) 118–131
10. Duh, R., Fürer, M.: Approximation of k -set cover by semi-local optimization. In: *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, ACM Press (1997) 256–264
11. Downey, R., Fellows, M.R.: *Parameterized Complexity*. Springer-Verlag (1999)
12. Eppstein, D.: Small maximal independent sets and faster exact graph coloring. In: *Algorithms and Data Structures: 7th International Workshop, WADS 2001*. Volume 2125 of *Lecture Notes in Computer Science*., Springer-Verlag (2001) 462–470
13. Papadimitriou, C.H., Yannakakis, M.: On limited nondeterminism and the complexity of the V-C dimension. *Journal of Computer and System Sciences* **53** (1996)
14. Chen, J., Kanj, I., Jia, W.: Vertex cover: further observations and further improvements. *Journal of Algorithms* **41** (2001) 280–301
15. Nemhauser, G.L., Trotter, L.E.: Vertex packings: Structural properties and algorithms. *Mathematical Programming* **8** (1975) 232–248
16. Galil, Z.: Efficient algorithms for finding maximum matching in graphs. *ACM Computing Surveys* **18** (1986) 23–38
17. Lovász, L., Plummer, M.D.: *Matching Theory*. Volume 29 of *Annals of Discrete Mathematics*. North Holland, Amsterdam (1986)
18. Berge, C.: Two theorems in graph theory. *Proceedings of the National Academy of Sciences U.S.A.* **43** (1957) 842–844
19. Hopcroft, J.E., Karp, R.M.: An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on Computing* **2** (1973) 225–231
20. Micali, S., Vazirani, V.V.: An $O(\sqrt{|v|} \cdot |E|)$ algorithm for finding maximum matching in general graphs. In: *21st Annual Symposium on Foundations of Computer Science*, Syracuse, New York, IEEE (1980) 17–27