Parameterized Local Search for Max c-Cut

Anonymous submission

Abstract

In the NP-hard MAX c-CUT problem, one is given an undirected edge-weighted graph G and wants to color the vertices of G with c colors such that the total weight of edges with distinct colored endpoints is maximal. The case with $c = 2$ is the famous MAX CUT problem. To deal with the NP-hardness of this problem, we study parameterized local search algorithms. More precisely, we study LS-MAX c-CUT where we are additionally given a vertex coloring f and an integer k and the task is to find a better coloring f' that differs from f in at most k entries, if such a coloring exists; otherwise, f is k-optimal. We show that LS-MAX c-CUT presumably cannot be solved in $g(k) \cdot n^{\mathcal{O}(1)}$ time even on bipartite graphs, for all $c \geq 2$. We then show an algorithm for LS-MAX c-CUT with running time $\mathcal{O}((3e\Delta)^k \cdot k^3 \cdot \Delta \cdot n)$, where Δ is the maximum degree of the input graph. Finally, we evaluate the practical performance of this algorithm in a hill-climbing search approach as a post-processing for state-of-the-art heuristics for MAX c-CUT. We show that using parameterized local search, the results of this heuristic can be further improved on a set of standard benchmark instances.

Introduction

Graph coloring and its generalizations are among the most famous NP-hard optimization problems (Jensen and Toft 2011) with numerous practical applications. In one prominent problem variant, we want to color the vertices of an edge-weighted graph with c colors so that the sum of the weights of all edges that have endpoints with different colors is maximized. This problem is known as MAX c-CUT (Frieze and Jerrum 1997; Kann et al. 1997) or MAXIMUM COLORABLE SUBGRAPH (Papadimitriou and Yannakakis 1991). Applications of MAX c-CUT include channel assignment in wireless networks (Subramanian et al. 2008), module detection in genetic interaction data (Leiserson et al. 2011), and scheduling of TV commercials where advertisers may specify that certain ads should be aired in different commercial breaks (Gaur, Krishnamurti, and Kohli 2009).

An equivalent formulation of the problem is to ask for a coloring that minimizes the weight sum of the edges whose endpoints receive the same color; this formulation is known as GENERALIZED GRAPH COLORING (Vredeveld and Lenstra 2003).

From a theoretical viewpoint, even restricted cases of MAX c-CUT are algorithmically hard: The special case $c = 2$ is the MAX CUT problem which is NP-hard even for positive unit weights (Karp 1972; Garey and Johnson 1979). Moreover, for all $c \geq 3$, the GRAPH COLORING problem where we ask for a coloring of the vertices with c colors such that the endpoints of every edge receive different colors is NP-hard (Karp 1972). As a consequence, MAX c-CUT is NP-hard for all $c \geq 3$, again even when all edges have positive unit weight. Due to these hardness results, MAX c-CUT is mostly solved using heuristic approaches (Festa et al. 2002; Leiserson et al. 2011; Ma and Hao 2017; Zhu, Lin, and Ali 2013). One of the techniques that is used is hill-climbing local search (Festa et al. 2002; Leiserson et al. 2011) with the 1-flip neighborhood. Here, an initial solution (usually computed by a greedy algorithm) is replaced by a better one in the 1-flip neighborhood as long as such a better solution exists. Herein, the 1-flip neighborhood of a coloring f is the set of all colorings that can be obtained by changing the color of 1 vertex. A coloring f that has no improving 1-flip is called 1-optimal and the problem of computing 1-optimal solutions has also received interest from a theoretical standpoint: It is presumably hard to find 1-optimal solutions for MAX CUT since it is PLS-complete on edge-weighted graphs (Schäffer and Yannakakis 1991) and thus, it is presumably not efficiently solvable in the worst case. This PLS-completeness result for the 1-flip neighborhood was later extended to GENERALIZED GRAPH COLORING, and thus to MAX c-CUT, for all c (Vredeveld and Lenstra 2003). For graphs where the absolute values of all edges are constant, however, a simple hill climbing algorithm terminates after $\mathcal{O}(m)$ improvements, where m is the number of edges in the input graph. Here, a different question arises: Can we replace the 1-flip neighborhood with a larger efficiently searchable neighborhood, to avoid being stuck in a bad local optimum? A natural candidate is the k-flip neighborhood where we are allowed to change the color of at most k vertices. Clearly, the k-flip neighborhood can be searched in $\mathcal{O}(n^k \cdot m)$ time where n is the number of vertices, but can we do better?

The ideal framework to answer this question is parameterized local search, where the ultimate goal would be to design an algorithm that in $g(k) \cdot n^{\mathcal{O}(1)}$ time either finds a better solution in the k-flip neighborhood or correctly answers that the current solution is k-optimal. Such a running time is preferable to the $\mathcal{O}(n^k \cdot m)$ since the degree of the polynomial running time part does not depend on k and thus the running time scales better with n. The framework also provides a negative toolkit that allows to conclude that an algorithm with such a running time is unlikely by showing W[1]-hardness. In fact, most parameterized local search problems turn out to be W[1]-hard with respect to the parameter k (Bonnet et al. 2019; Dörnfelder et al. 2014; Fellows et al. 2012; Guo, Hermelin, and Komusiewicz 2014; Guo et al. 2013; Gaspers et al. 2012; Marx 2008; Szeider 2011). In spite of these negative results, practical parameterized local search algorithms are possible, as evidenced by the LS-VERTEX COVER problem. In this problem, the input is an undirected graph G with a vertex cover S and the question is whether the k-swap neighborhood of S contains a smaller vertex cover. The key to obtain practical parameterized local search algorithms is to consider additional structural parameters such as the maximum degree Δ of the input graph. As shown by (Katzmann and Komusiewicz 2017) LS-VERTEX COVER can be solved in $(2\Delta)^k \cdot n^{\mathcal{O}(1)}$ time. More importantly, an experimental evaluation of this algorithm showed that it can be tuned to solve the problem for $k \approx 20$ on large sparse graphs, and that k-optimal solutions for $k \geq 9$ turned out to be optimal for almost all graphs considered in the experiments.

Our Results. We study LS-MAX c-CUT, where we want to decide whether a given coloring has a better one in its k-flip neighborhood. We first show that LS-MAX c-CUT is presumably not solvable in $g(k) \cdot n^{\mathcal{O}(1)}$ time by showing W[1]-hardness of the problem. We then show an algorithm with running time $\mathcal{O}((3e\Delta)^k \cdot c \cdot k^3 \cdot \Delta \cdot n)$, where Δ is the maximum degree of the input graph. The algorithm is based on two main facts: First, we show that minimal improving flips are connected in the input graph. This allows to enumerate candidate flips in $\mathcal{O}((e\Delta)^k \cdot k \cdot n)$ time. Second, we show that, given a set of k vertices to flip, we can determine the best way to flip their colors in $\mathcal{O}(3^k \cdot c \cdot k^2 + k \cdot \Delta)$ time. We then discuss several ways to speed up the algorithm, for example by computing upper bounds for the improvement of partial flips. We then evaluate our algorithm experimentally when it is applied as post-processing for a state-of-the-art MAX c-CUT heuristic (Ma and Hao 2017). In this application, we take the solutions computed by the heuristic and improve them by hill-climbing with the k-flip neighborhood for increasing values of k. More precisely, we try to find an improving 1-flip as long as it exists. If this is not the case, we try to find an improving 2-flip, if no improving 2-flip exists, we try to find an improving 3-flip and so on. We show that, for a standard benchmark data set, a large fraction of the previously best solutions can be improved by our algorithm, leading to new record solutions for these instances. The post-processing is particularly successful for the harder instances of the data set (with $c > 2$ and both positive and negative edge weights).

Preliminaries

Notation. For integers i and j with $i \leq j$, we define $[i,j] := \{k \in \mathbb{N} \mid i \leq k \leq j\}$. For a set A, we denote with $\binom{A}{2} := \{\{a,b\} \mid a \in A, b \in A\}$ the collection of all size-two subsets of A. For two sets A and B, we denote with $A \oplus B := (A \setminus B) \cup (B \setminus A)$ the *symmetric difference* of A and B. A tuple (A, B) is a *partition* of C, if $A \cup B = C$ and $A \cap B = \emptyset$.

An (undirected) graph $G = (V, E)$ consists of a vertex set V and an edge set $E \subseteq \binom{V}{2}$. For vertex sets $S \subseteq V$ and $T \subseteq V$ we denote with $E_G(S, T) := \{\{s,t\} \in E \mid s \in S, t \in T\}$ the edges between S and T and we denote with $E_G(S) := E_G(S, S)$ the edges between vertices of S. For a collection $\mathcal{S}$ of pairwise disjoint subsets of V, we define $E_G(\mathcal{S}) := \bigcup_{\{S,T\} \in \binom{\mathcal{S}}{2}} E_G(S, T)$ as the edges between any pair of subsets of V in $\mathcal{S}$. Moreover, we define $G[S] := (S, E_G(S))$ as the *subgraph of G induced by S*. A vertex set S is *connnected* if $G[S]$ is a connected graph. For a vertex $v \in V$, we denote with $N_G(v) := \{w \in V \mid \{v, w\} \in E\}$ the *open neighborhood* of v in G and with $N_G[v] := N_G(v) \cup \{v\}$ the *closed neighborhood* of v in G. Analogously, for a vertex set $S \subseteq V$, we define $N_G[S] := \bigcup_{v \in S} N_G[v]$ and $N_G(S) := \bigcup_{v \in S} N_G(v) \setminus S$. Moreover, the *closed i-neighborhood $N^i_G[x]$ of x* is defined via $N^0_G[x] := \{x\}$, and $N^i_G[x] := N_G[N^{i-1}_G[x]]$ for $i > 0$. We also say that *vertices v and w have distance at least $i+1$* if $w \notin N^i_G[v]$. If G is clear from context, we may omit the subscript.

Problem Formulation. Let X and Y be sets and let $f, f' : X \to Y$. The *flip* between f and f' is defined as $D(f, f') := \{x \in X \mid f(x) \neq f'(x)\}$ and the *flip distance* between f and f' is defined as $d(f, f') := |D(f, f')|$. For an integer c and a graph $G = (V, E)$, a function $f : V \to [1, c]$ is a *c-coloring* of G. Let f be a c-coloring of G, we define the set $E(f)$ of *properly colored edges* as $E(f) := \{\{u, v\} \in E \mid f(u) \neq f(v)\}$. Let f and f' be c-colorings of G. We say that f and f' are *k-neighbors* if $d(f, f') \leq k$. For an edge-weight function $\omega : E \to \mathbb{Q}$ and an edge set $E' \subseteq E$, we let $\omega(E')$ denote the sum of the weights of all edges in E'. If $\omega(E(f)) > \omega(E(f'))$, we say that f is *improving* over f'. Finally, a c-coloring f is *k-optimal* if f has no improving k-neighbor f'. Let $i \in [1, c]$, then $f^{-1}(i) := \{v \in V \mid f(v) = i\}$ is the set of vertices of color i. The problems of finding an improving neighbor of a given coloring can now be formalized as follows.

LS-MAX c-CUT
Input: A graph $G = (V, E)$, $c \in \mathbb{N}$, a weight function $\omega : E \to \mathbb{Q}$, a c-coloring $f : V \to [1, c]$, and $k \in \mathbb{N}$.
Question: Is there a c-coloring $f' : V \to [1, c]$ with $d(f, f') \leq k$ such that $\omega(E(f')) > \omega(E(f))$?

The special case of LS-MAX c-CUT where $c = 2$ can alternatively be defined as follows.

LS-MAX CUT
Input: A graph $G = (V, E)$, a weight function $\omega : E \to \mathbb{Q}$, a partition (A, B) of V, and $k \in \mathbb{N}$.
Question: Is there a set $S \subseteq V$ of size at most k such that $\omega(E(A, B)) < \omega(E(A \oplus S, B \oplus S))$?

While these problems are defined as decision problems, our algorithms solve the search problem that returns an improving k-flip if it exists.

Let f and f' be c-colorings of a graph G. We say that f' is an *inclusion-minimal improving k-flip for* f, if f' is an improving k-neighbor of f and if there is no improving k-neighbor $\tilde{f}$ of f with $D(f,\tilde{f}) \subsetneq D(f,f')$. Let (A,B) be a partition of G. In the context of LS-MAX CUT, we call a vertex set S *inclusion-minimal improving k-flip for* (A,B), if $|S| \leq k$, $\omega(E(A \oplus S, B \oplus S)) > \omega(E(A,B))$, and if there is no $S' \subsetneq S$ such that $\omega(E(A \oplus S', B \oplus S')) > \omega(E(A,B))$.

W[1]-hardness of LS Max Cut

We first show an intractability result for LS-MAX CUT. More precisely, we show that LS-MAX CUT is W[1]-hard for parameterization by k even on bipartite graphs with unit weights. This implies that LS-MAX CUT presumably cannot be solved within $g(k) \cdot n^{\mathcal{O}(1)}$ time for any computable function g.

To prove the hardness, we introduce the term of *blocked vertices* in instances with unit weights. Intuitively, a vertex v is blocked for a color class i if we can conclude that v does not move to i in a solution just by considering the neighborhood of v. This concept is formalized as follows.

Definition 1. *Let $G = (V,E)$ be a graph, let f be a c-coloring of G, and let k be an integer. Moreover, let v be a vertex of V and let $i \in [1,c] \setminus \{f(v)\}$ be a color. The vertex v is (i,k)-*blocked *in G with respect to f if $|\{w \in N(v) \mid f(w) = i\}| \geq |\{w \in N(v) \mid f(w) = f(v)\}| + 2k - 1$.*

Lemma 1. *Let $G = (V,E)$ be a graph, let f be a c-coloring of G, let k be an integer. Moreover, let v be a vertex in V which is (i,k)-blocked in G with respect to f. Then, there is no inclusion-minimal improving k-neighbor f' of f with $f'(v) = i$.*

Proof. Let f' be a c-coloring of G with $d(f,f') \leq k$ and $f'(v) = i$. Hence, $v \in D(f,f')$ and thus $|D(f,f') \cap N(v)| \leq k-1$. Consequently $|\{w \in N(v) \mid f'(w) = f(v)\}| \leq |\{w \in N(v) \mid f(w) = f(v)\}| + k - 1$ and $|\{w \in N(v) \mid f'(w) = i\}| \geq |\{w \in N(v) \mid f(w) = i\}| - k + 1$. Since v is (i,k)-blocked in G with respect to f, $|\{w \in N(v) \mid f'(w) = f(v)\}| < |\{w \in N(v) \mid f'(w) = i\}|$. Consequently, for the c-coloring f^* of G that agrees with f' on $V \setminus \{v\}$ and where $f^*(v) := f(v)$, we have $|E(f^*)| > |E(f')|$. Hence, f' is not an inclusion minimal improving k-neighbor of f. □

The idea of blocking a vertex by its neighbors finds application in the construction for the W[1]-hardness from the next theorem.

Theorem 1. LS-MAX CUT *is* W[1]*-hard when parameterized by k on bipartite* 2*-degenerate graphs with unit weights.*

Proof. We reduce from CLIQUE, which is given an undirected graph G and $k \in \mathbb{N}$ and asks whether G contains a clique of size k. CLIQUE is W[1]-hard when parameterized by the size k of the sought clique (Downey and Fellows 2013; Cygan et al. 2015).

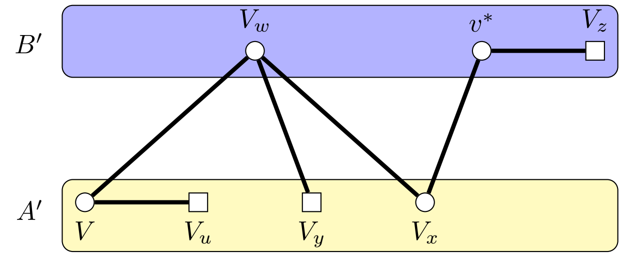

Figure 1: The connections between the different vertex sets in G'. Two vertex sets V' and V'' are non-adjacent in the figure if $E(V',V'') = \emptyset$. Each vertex v in a vertex set with a rectangular node is k'-blocked from the opposite part of the partition. The vertex set Γ is not shown.

Let $I := (G = (V,E), k)$ be an instance of CLIQUE. We construct an equivalent instance $I' := (G' = (V',E'), \omega', A', B', k')$ of LS-MAX CUT with $\omega' : E' \to \{1\}$ as follows. We start with an empty graph G' and add each vertex of V to G'. Next, for each edge $e \in E$, we add two vertices u_e and w_e to G' and add for each such vertex an edge to each endpoint of e to G'. Afterwards, we add a vertex v^* to G' and for each edge $e \in E$, we add vertices x_e and y_e and add edges $\{w_e, x_e\}, \{w_e, y_e\}$, and $\{x_e, v^*\}$ to G'. Finally, we add a set V_z of $|E| - 2 \cdot \binom{k}{2} + 1$ vertices to G' and connect each of vertex of V_z to v^*.

In the following, let $V_\alpha := \{\alpha_e \mid e \in E\}$ for any $\alpha \in \{u, w, x, y\}$. We set

$$B' := V_w \cup \{v^*\} \cup V_z,\ A' := V(G') \setminus B', \text{ and}$$
$$k' := 2 \cdot \binom{k}{2} + k + 1.$$

To finish the construction, we add further vertices to A' and B'. Intuitively, these vertices cause that some vertices are blocked in the final instance.

For each vertex $v' \in V_u \cup V_y$, we add a set of $2k' + 2$ vertices to B' that are only adjacent to v' and for each vertex $v' \in V_z$, we add a set of $2k' + 2$ vertices to A' that are only adjacent v'. Let Γ be the set of those additional vertices.

Note that each vertex in $V_u \cup V_y$ is contained in A', has at most two neighbors in A' and $2k' + 2$ neighbors in B' and each vertex in V_z is contained in B', has one neighbor in B' and $2k' + 2$ neighbors in A'. Hence, each vertex in $V_u \cup V_y$ is (B', k')-blocked and each vertex in V_z is (A', k')-blocked. Consequently, due to Lemma 1, no inclusion minimal improving k'-flip for (A', B') contains any vertex of $V_u \cup V_y \cup V_z$. As a consequence, no inclusion minimal improving k'-flip for (A', B') contains any vertex of Γ. Figure 1 shows a sketch of the vertex sets and their connections in G'. Note that G' is bipartite and 2-degenerate.

Before we show the correctness, we provide some intuition. The additional vertices guarantee that only vertices in V, V_w, V_x, and the vertex v^* can flip their colors. A clique S in the original instance then corresponds to a flip of v^*, the vertices corresponding to S, and the vertices w_e and x_e corresponding to the edge of the clique. Swapping these vertices transforms edges between v^* and V_z into properly colored edges but causes that edges between V_x and v^* receive the same color on both endpoints. Since a clique has $\binom{|S|}{2}$ edges, the swaps of the vertices corresponding to the clique edges guarantee that the total flip is improving.

Next, we show that I is a yes-instance of CLIQUE if and only if I' is a yes-instance of LS-MAX CUT.

($\Rightarrow$) Let $S \subseteq V$ be a clique of size k in G. Hence, $\binom{S}{2} \subseteq E$. We set $S' := S \cup \{w_e, x_e \mid e \in \binom{S}{2}\} \cup \{v^*\}$. Note that $|S'| = k + 2 \cdot \binom{k}{2} + 1 = k'$. Let $C := E(A', B')$ and let $C' := E(A' \oplus S', B' \oplus S')$. It remains to show that $|C'| > |C|$.

For each $v \in S$ and each $v' \in N_G(v) \setminus S$, we have $\{v, w_{\{v,v'\}}\} \in C \setminus C'$ and $\{v, u_{\{v,v'\}}\} \in C' \setminus C$. For each $v' \in N_G(v) \cap S$, we have $\{v, w_{\{v,v'\}}\} \in C \cap C'$ and $\{v, u_{\{v,v'\}}\} \in C' \setminus C$. Moreover, for each edge $e \in \binom{S}{2} \subseteq E$, we have $\{w_e, x_e\} \in C \cap C'$, $\{x_e, v^*\} \in C \cap C'$, and $\{w_e, y_e\} \in C \setminus C'$. Finally, for each edge $e \in E \setminus \binom{S}{2}$, we have $\{v^*, x_e\} \in C \setminus C'$ and for each $z \in V_z$, we have $\{v^*, z\} \in C' \setminus C$. All remaining edges of E' are either contained in $C \cap C'$ or contained in neither C nor C'. Hence,

$$\begin{aligned} C \setminus C' = \; & \{\{v, w_{\{v,v'\}}\} \mid v \in S, v' \in N_G(v) \setminus S\} \\ & \cup \{\{w_e, y_e\} \mid e \in \binom{S}{2}\} \\ & \cup \{\{v^*, x_e\} \mid e \in E \setminus \binom{S}{2}\}. \end{aligned}$$

Furthermore, we have

$$\begin{aligned} C' \setminus C = & \{\{v, u_{\{v,v'\}}\} \mid v \in S, v' \in N_G(v)\} \\ & \cup \{\{v^*, z\} \mid z \in V_z\}. \end{aligned}$$

Since $|V_z| = |E| - 2 \cdot \binom{k}{2} + 1$, we get

$$\begin{aligned} & |C' \setminus C| - |C \setminus C'| \\ = & |\{\{v, u_{\{v,v'\}}\} \mid v \in S, v' \in N_G(v)\}| \\ & - |\{\{v, w_{\{v,v'\}}\} \mid v \in S, v' \in N_G(v) \setminus S\}| \\ & + |\{\{v^*, z\} \mid z \in V_z\}| \\ & - |\{\{w_e, y_e\} \mid e \in \binom{S}{2}\} \cup \{\{v^*, x_e\} \mid e \in E \setminus \binom{S}{2}\}| \\ = & 2 \cdot \binom{k}{2} + |E| - 2 \cdot \binom{k}{2} + 1 - |E| = 1 \end{aligned}$$

Hence, I' is a yes-instance of LS-MAX CUT.

($\Leftarrow$) Let $S' \subseteq V'$ be an inclusion minimal improving k'-flip for (A', B'). Due to Lemma 1, we can assume without loss of generality that $S' \subseteq V \cup V_w \cup V_x \cup \{v^*\}$. Moreover, assume that S' is inclusion minimal. By construction, each vertex $v \in V$ is adjacent to $|N_G(v)|$ vertices in A' and adjacent to $|N_G(v)|$ vertices in B' in G'. Since no vertex of $\{u_e \mid e \in E\}$ is contained in S' and S' is inclusion minimal, for each vertex $v \in S' \cap V$, there is at least one $w_e \in S'$ with $v \in e$, as otherwise,

$$\begin{aligned} & |E(A' \oplus (S' \setminus \{v\}), B' \oplus (S' \setminus \{v\}))| \\ \geq & |E(A' \oplus S', B' \oplus S')| \\ > & |E(A', B')|. \end{aligned}$$

Moreover, each vertex $w_e \in V_w$ is adjacent to the four vertices $e \cup \{x_e, y_e\}$ in A' and adjacent to no vertex in B'.

Since no vertex of V_y is contained in S' and S' is inclusion minimal, for each vertex $w_e \in S' \cap V_w$, all three vertices $e \cup \{x_e\}$ are contained in S', as otherwise,

$$\begin{aligned} & |E(A' \oplus (S' \setminus \{w_e\}), B' \oplus (S' \setminus \{w_e\}))| \\ \geq & |E(A' \oplus S', B' \oplus S')| \\ > & |E(A', B')|. \end{aligned}$$

Furthermore, each vertex $x_e \in V_x$ is adjacent to the vertices w_e and v^* in B' and adjacent to no vertex in A'. Since S' is inclusion minimal, for each vertex $x_e \in S' \cap V_x$, both w_e and v^* are contained in S', as otherwise,

$$\begin{aligned} & |E(A' \oplus (S' \setminus \{x_e\}), B' \oplus (S' \setminus \{x_e\}))| \\ \geq & |E(A' \oplus S', B' \oplus S')| \\ > & |E(A', B')| \end{aligned}$$

Finally, recall that v^* is adjacent to the $|E|$ vertices V_x in A' and to the $|E| - 2 \cdot \binom{k}{2} + 2$ vertices V_z in B'. Hence, since S' is inclusion minimal and no vertex of V_z is contained in S', if $v^* \in S'$, then $|V_x \cap S'| \geq \binom{k}{2}$, as otherwise,

$$\begin{aligned} & |E(A' \oplus (S' \setminus \{v^*\}), B' \oplus (S' \setminus \{v^*\}))| \\ \geq & |E(A' \oplus S', B' \oplus S')| \\ > & |E(A', B')|. \end{aligned}$$

Since $S' \subseteq V \cup V_w \cup V_x \cup \{v^*\}$ and $S' \neq \emptyset$, S' contains v^*, at least $\binom{k}{2}$ vertices of V_x, for each vertex x_e of V_x the vertex w_e and the endpoints of e. Let $S := S' \cap V$. By the fact that $|S'| \leq k' = 2 \cdot \binom{k}{2} + k + 1$, $|V_x \cap S'| = |V_w \cap S'| \geq \binom{k}{2}$, and $v^* \in S'$, S has size at most k. Moreover, since for each $w_e \in S'$, both endpoints of the edge e are contained in S and $|V_w \cap S'| \geq \binom{k}{2}$, S is a clique of size k in G. Hence, I is a yes-instance of CLIQUE. □

The above reduction can be adapted to prove W[1]-hardness for k of LS-MAX c-CUT for each fixed $c \geq 2$. Consider the instance $I := (G, \omega, (A, B), k)$ of LS-MAX CUT that has been constructed in the proof of Theorem 1 and let $c > 2$. For every vertex v of G, we add further degree-one neighbors. More precisely, for every color $i \in [3, c]$, the vertex v receives additional neighbors of color i such that v is (i, k)-blocked. Let G' be the resulting graph.

Then, for any inclusion minimal improving k-flip f' for f of G', we have $S := D(f, f') \subseteq A \cup B$, $f'(a) = 2$ for each $a \in A \cap S$, and $f'(b) = 1$ for each $b \in B \cap S$. Hence, I is a yes-instance of LS-MAX CUT if and only if the instance I' is a yes-instance of LS-MAX c-CUT. Since we only added degree-one vertices, the graph is still bipartite and 2-degenerate.

Corollary 1. *For every $c \geq 2$, LS-MAX c-CUT is W[1]-hard for k on bipartite 2-degenerate graphs with unit weights.*

Algorithms

Our algorithm for LS-MAX c-CUT follows a simple framework: Generate a collection of candidate sets S that may improve the coloring if the vertices in S flip their colors. For each such candidate set S, we only now that the colors of the vertices of S change, but we do not yet now which new color the vertices receive. To answer this question, that is, to find whether there is any coloring of S that leads to an improved global coloring, we use dynamic programming.

We first describe the subroutine that we use to check for the existence of a good coloring of a given candidate set S.

Theorem 2. *Let $G = (V, E)$ be a graph, let $\omega : E \to \mathbb{Q}$ be an edge weight function, let f be a c-coloring of G, and let $S \subseteq V$ be a set of size at most k. One can compute in $\mathcal{O}(3^k \cdot c \cdot k^2 + k \cdot \Delta(G))$ time a c-coloring f' of G such that $D(f, f') \subseteq S$ and $\omega(E(f'))$ is maximal.*

Proof. We use dynamic programming. Initially, we compute for each $v \in S$ and each $i \in [1, c]$ the weight $\theta_v^i := \omega(\{\{v, w\} \in E \mid w \in N(v) \setminus S, f(w) \neq i\})$ of edges between v and vertices not contained in $f^{-1}(i) \cup S$. Moreover, we compute the weight ω_S of all properly colored edges of $E(S, N[S])$ as $\omega_S := \omega(\{\{u, v\} \in E(S, N[S]) \mid f(u) \neq f(v)\})$. This can be done in $\mathcal{O}(c \cdot k + k \cdot \Delta(G))$ time.

The table T has entries of type $T[S', c']$ where $S' \subseteq S$ and $c' \in [1, c]$. Intuitively, each entry $T[S', c']$ stores the maximum sum of weights of properly colored edges with at least one endpoint in S' and no endpoint in $S \setminus S'$ such that the following holds:

1. the vertices in S' have some color in $[1, c']$, and
2. every vertex $v \in V \setminus S$ has color $f(v)$.

We start to fill the dynamic programming table by setting $T[S', 1] := \sum_{v \in S'} \theta_v^1$ for each $S' \subseteq S$.

For $S' \subseteq S$ and $c' \in [2, c]$, we set:

$$T[S', c'] := \max_{S'' \subseteq S'} T[S' \setminus S'', c' - 1] + \omega(E(S'', S' \setminus S'')) + \sum_{v \in S''} \theta_v^{c'}.$$

The maximal improvement $\omega(E(f')) - \omega(E(f))$ for any c-coloring f' with $D(f, f') \subseteq S$ can then be found by evaluating $T[S, c] - \omega_S$. Intuitively, the term $T[S, c] - \omega_S$ corresponds to the maximum sum of weights of properly colored edges we get when distributing the vertices of S among all color classes minus the original weights when every vertex of S sticks with its color under f. The corresponding c-coloring can be found via traceback.

The formal correctness proof is straightforward and thus omitted. Hence, it remains to show the running time. The dynamic programming table T has $2^k \cdot c$ entries. Each of these entries can be computed in $2^{|S'|} \cdot k^2$ time. Consequently, all entries can be computed in $\mathcal{O}(\sum_{i=0}^{n} \binom{k}{i} \cdot 2^i \cdot c \cdot k^2) = \mathcal{O}(3^k \cdot c \cdot k^2)$ time in total. Hence, the total running time is $\mathcal{O}(3^k \cdot c \cdot k^2 + k \cdot \Delta(G))$ time. ☐

For LS-MAX CUT, the situation is even simpler: When given a set $S \subseteq V$ of k vertices that must flip their colors, the best possible improvement can be computed in $\mathcal{O}(k \cdot \Delta(G))$ time, since every vertex of S must replace its color with the unique other color.

Recall that the idea of our algorithms for LS-MAX CUT and LS-MAX c-CUT is to iterate over possible candidate sets of vertices that may flip their colors. With the next lemma we show that it suffices to consider those vertex sets that are connected in the input graph.

Lemma 2. *Let $I := (G = (V, E), c, \omega, f, k)$ be an instance of LS-MAX c-CUT, then for every improving k-neighbor f' of f where $d(f, f')$ is minimal, the flip $D(f, f')$ is connected in G.*

Proof. Let f' be an improving k-neighbor of f. Let $S' := D(f, f')$ be the flip of f and f' and let $\mathcal{C}$ denote the connected components in $G[S']$. We show that if there are at least two connected components in $\mathcal{C}$, then there is an improving k-neighbor $\tilde{f}$ of f with $d(f, \tilde{f}) < d(f, f')$. For each connected component $C \in \mathcal{C}$, let $E_C^+ := (E(C, V) \cap E(f')) \setminus E(f)$ denote the set of properly colored edges in $E(f') \setminus E(f)$ that have at least one endpoint in C and let $E_C^- := (E(C, V) \cap E(f)) \setminus E(f')$ denote the set of properly colored edges in $E(f) \setminus E(f')$ that have at least one endpoint in C. Since $0 < \omega(E(f')) - \omega(E(f)) = \sum_{C \in \mathcal{C}} (\omega(E_C^+) - \omega(E_C^-))$, there is at least one connected component $S \in \mathcal{C}$ with $\omega(E_S^+) - \omega(E_S^-) > 0$. Let $\tilde{f}$ be the c-coloring of G that agrees on $V \setminus S$ with f and agrees on S with f'. Hence, $\tilde{f}$ is an improving k-neighbor of f with $d(f, \tilde{f}) = |S| < d(f, f')$. ☐

We next combine Theorem 2 and Lemma 2.

Theorem 3. LS-MAX c-CUT *can be solved in $\mathcal{O}((3 \cdot e)^k \cdot (\Delta(G) - 1)^{k+1} \cdot c \cdot k^3 \cdot n)$ time.*

Proof. Let $I = (G, c, \omega, f, k)$ be an instance of LS-MAX c-CUT. By Lemma 2, I is a yes-instance of LS-MAX c-CUT if and only if f has an improving k-neighbor f' where $S := D(f, f')$ is connected. Since we can enumerate all connected vertex sets S of size at most k in G in $\mathcal{O}(e^k \cdot (\Delta(G) - 1)^k \cdot k \cdot n)$ time (Komusiewicz and Sorge 2015; Komusiewicz and Sommer 2021) and we can compute for each such set S the c-coloring f' with $D(f, f') \subseteq S$ that maximizes $\omega(E(f'))$ in $\mathcal{O}(3^k \cdot c \cdot k^2 + \Delta(G) \cdot k)$ time due to Theorem 2, we obtain the stated running time. ☐

For LS-MAX CUT, we can improve the running time since computing the unique flip for a given set can be done in $\mathcal{O}(\Delta(G) \cdot k)$ time.

Theorem 4. LS-MAX CUT *can be solved in $\mathcal{O}(e^k \cdot (\Delta(G) - 1)^{k+1} \cdot k^2 \cdot n)$ time.*

Proof. Let $I = (G, \omega, (A, B), k)$ be an instance of LS-MAX CUT. Due to Lemma 2, I is a yes-instance of LS-MAX CUT if and only if there is a connected vertex set S of size at most k such that $\omega(E(A \oplus S, B \oplus S)) > \omega(E(A, B))$. Since we can enumerate all connected vertex sets S of size at most k in G in $\mathcal{O}(e^k \cdot (\Delta(G) - 1)^k \cdot k \cdot n)$ time (Komusiewicz and Sorge 2015; Komusiewicz and Sommer 2021) and we can compute for each such set S the weight $\omega(E(A\oplus S, B\oplus S))$ in $\mathcal{O}(\Delta(G)\cdot k)$ time, we can solve I in the stated running time. □

Hill-Climbing Algorithm To obtain not only a single improvement of a given coloring but a c-coloring with a total weight of properly colored edges as high as possible, we introduce the following hill-climbing algorithm.

Given an initial coloring f, we set the initial value of k to one. In each step, we use the above-ementioned algorithm for LS-MAX c-CUT to search for an improving coloring in the k-flip neighborhood of the current coloring. Whenever the algorithm finds an improving k-neighbor f' for the current coloring f, the current coloring gets replaced by f' and k gets set back to one. If the current coloring is k-optimal, the value of k is incremented and the algorithm continues to search for an improvement in the new k-flip neighborhood. This is done until a given time limit is reached.

ILP Formulation. In our experiments, we also use the following ILP formulation for LS-MAX c-CUT. For each vertex $v \in V$ and each color $i \in [1, c]$, we use a binary variable $x_{v,i}$ which is equal to one if and only if $f'(v) = i$. We further use for each edge $e \in E$ a binary variable y_e to indicate if e is properly colored with respect to f'. Thus, for each edge $\{u, v\} \in E$, the variable y_e is set to one if and only if for each color $i \in [1, c]$, $x_{u,i} = 0$ or $x_{v,i} = 0$.

maximize $\sum_{e\in E} y_e \cdot \omega(e)$ subject to

$$\begin{aligned}
\sum_{i\in[1,c]} x_{v,i} = 1 &\quad \text{for each } v \in V\\
x_{u,i} + x_{v,i} + y_{\{u,v\}} \leq 2 &\quad \text{for each } \{u,v\} \in E\\
&\quad \text{with } \omega(\{u,v\}) > 0\\
&\quad \text{and each } i \in [1,c]\\
x_{u,i} + \sum_{j\in[1,c]\setminus\{i\}} x_{v,j} - y_{\{u,v\}} \leq 1 &\quad \text{for each } \{u,v\} \in E\\
&\quad \text{with } \omega(\{u,v\}) < 0\\
&\quad \text{and each } i \in [1,c]\\
x_{v,i} \in \{0,1\} &\quad \text{for each } v \in V\\
&\quad \text{and each } i \in [1,c]\\
y_e \in \{0,1\} &\quad \text{for each } e \in E
\end{aligned}$$

Speedup Strategies

We now introduce several speedup strategies that we use in our implementation to avoid enumerating all possible subsets of vertices. First we describe how to speed up the algorithm for LS-MAX c-CUT.

Upper Bounds

To prevent the program from enumerating all possible connected subsets of size at most k, we use upper bounds to determine for a given connected subset S' of size smaller than k if S' can possibly be extended to a set S of size k such that there is an improving c-coloring f' for G where the flip of f and f' is exactly S. If there is no such possibility, then we prevent our program from enumerating supersets of S'. With the next definition we formalize this concept.

Definition 2. *Let $I := (G, c, \omega, f, k)$ be an instance of* LS-MAX c-CUT *and let S' with $|S'| < k$ be a subset of vertices of G. A value $b(I, S') \in \mathbb{Q}$ is called an* upper bound *if for every coloring f' of G, with $S' \subset D(f, f')$ and $d(f, f') = k$, we have*

$$b(I, S') \geq \omega(E(f')).$$

In our implementation, we use upper bounds as follows: Given a set S' we compute the value $b(I, S')$ and check if it is smaller than $\omega(E(f))$ for the current coloring f. If this is the case, we abort the enumeration of supersets of S', otherwise, we continue.

We introduce two upper bounds; one for $c = 2$ and one for $c \geq 3$. To describe these upper bounds, we introduce the following notation: Given a vertex v and a color i, we let $\omega_v^i := \omega(\{\{v, w\} \mid w \in N(v) \wedge f(w) \neq i\})$ denote the total weight of properly colored edges incident with v if we change the color of v to i in a given coloring f. Thus, the term $\omega_v^i - \omega_v^{f(v)}$ describes the improvement we obtain for changing only the color of v to i. Furthermore, let $\omega_{\max} := \max_{e\in E} |\omega(e)|$ denote the maximum absolute edge weight.

Upper Bound for $c = 2$. Let I be an instance of LS-MAX c-CUT with $c = 2$ and let S' be a vertex set of size less than k. Since $c = 2$, we let $\overline{f(v)}$ denote the unique color distinct from $f(v)$ for each vertex v. For a vertex set $A \subseteq V$, let f_A denote the coloring where $f_A(v) := f(v)$ for all $v \notin A$ and $f_A(v) = \overline{f(v)}$, otherwise. Intuitively, f_A is the coloring resulting from f when exactly the vertices in A change their colors. For each vertex $v \in V \setminus S'$, we define $\alpha_v := \omega_v^{\overline{f(v)}} - \omega_v^{f(v)} + \beta_v$, where

$$\beta_v := \sum_{\substack{e\in E(v,S')\\ e\in E(f)}} 2\cdot\omega(e) - \sum_{\substack{e\in E(v,S')\\ e\notin E(f)}} 2\cdot\omega(e).$$

Intuitively, $\alpha_v - \beta_v$ is an upper bound for the improvement we obtain, when we choose to change only the color of v to $\overline{f(v)}$. The term β_v corresponds to the contribution of the edges between v and the vertices of S'. Hence, α_v is the improvement over the coloring $f_{S'}$ we obtain by changing only the color of v. Let $Y \subseteq V \setminus S'$ be a set containing the $k - |S'|$ vertices with biggest α_v-values from $V \setminus S'$. We define the upper bound by

$$b_{c=2}(I, S') := \omega(E(f_{S'})) + \underbrace{\sum_{v\in Y} \alpha_v}_{(1)} + \underbrace{2\binom{k-|S'|}{2}\omega_{\max}}_{(2)}.$$

Recall that the overall goal is to find a set X such that changing the colors of $S' \cup X$ results in a better coloring. The summand (1) corresponds to an overestimation of all weights of edges incident with exactly one vertex of X by fixing the falsely counted edges between X and S' due to the included β_v summands. The summand (2) corresponds to an overestimation of the weight of properly colored edges with both endpoints in X. We next show that $b_{c=2}$ is in fact an upper bound.

Proposition 1. *If $c = 2$, then $b_{c=2}(I, S')$ is an upper bound.*

Proof. Let f' be a coloring with $S' \subset D(f, f')$ and $d(f, f') = k$ and let $X := D(f, f') \setminus S'$. We show $\omega(E(f')) \leq b_{c=2}(I, S')$. To this end, we consider the coloring $f_{S'}$ that results from f when exactly the vertices in S' change their colors and analyze how $\omega(E(f'))$ differs from $\omega(E(f_{S'}))$.

$$\begin{aligned}
&\omega(E(f')) \\
&= \omega(E(f_{S'})) + \underbrace{\sum_{v\in X}(\omega_v^{\overline{f(v)}} - \omega_v^{f(v)})}_{(1)} \\
&\quad + \underbrace{\sum_{\substack{e\in E(X,S')\\ e\in E(f)}} 2\cdot\omega(e) - \sum_{\substack{e\in E(X,S')\\ e\notin E(f)}} 2\cdot\omega(e)}_{(2)} \\
&\quad + \underbrace{\sum_{\substack{e\in E(X)\\ e\in E(f)}} 2\cdot\omega(e) - \sum_{\substack{e\in E(X)\\ e\notin E(f)}} 2\cdot\omega(e)}_{(3)}
\end{aligned}$$

By adding (1) to $\omega(E(f_{S'}))$, we added the weight of all properly colored edges if only v changes its color for every vertex $v \in X$. The difference between $\omega(E(f'))$ and $\omega(E(f_{S'}))$ + (1) then consists of all edge-weights that were falsely counted in (1) since both endpoints were moved. To compensate this, the summands (2) and (3) need to be added. Summand (2) corresponds to falsely counted edges with one endpoint in X and one endpoint in S', while (3) corresponds to falsely counted edges with both endpoints in X. Observe that every falsely counted edge weight was counted for both of its endpoints within $\omega(E(f_{S'}))$ + (1). Thus, each edge weight in (2) and (3) needs to be multiplied by 2.

Note that (3) is upper bounded by $2\binom{k-|S'|}{2}\cdot\omega_{\max}$. Furthermore, note that (1) + (2) $= \sum_{v\in X}\alpha_v$. Thus, by the construction of the set Y we have (1) + (2) $\leq \sum_{v\in Y}\alpha_v$. Consequently, we have $\omega(E(f')) \leq b_{c=2}(I, S')$. □

Upper Bound for $c \geq 3$. We next present an upper bound $b_{c\geq3}$ that works for the case where $c \geq 3$. Recall that the upper bound $b_{c=2}$ relies on computing $\omega(E(f_{S'}))$, where $f_{S'}$ is the coloring resulting from f when exactly the vertices in S' change their colors. This was possible since for $c = 2$, there is only one coloring for which the flip with f is exactly S'. In case of $c \geq 3$, each vertex in S' has $c-1 \geq 2$ options to change its color. Our upper bound $b_{c\geq3}$ consists of a summand that is at least as big as the largest sum of edge weights when only the vertices in S' change their colors.

To specify the summand in $b_{c\geq3}$ corresponding to this contribution of $f_{S'}$, we introduce the following notation: Given a vertex $v \in S'$ and a color i, we let $\theta_v^i := \omega(\{\{v, w\} \mid w \in N(v) \setminus S' \wedge f(w) \neq i\})$. Analogously to ω_v^i, the value θ_v^i describes the weight of properly colored edges when changing the color of v to i, but excludes all edges inside S'. We define the term

$$b(S') := \omega(E(f)) + \binom{|S'|}{2}\cdot\omega_{\max} - \sum_{\substack{e\in E(S')\\ e\in E(f)}}\omega(e) + \sum_{v\in S'}\left(\max_{i\neq f(v)}\theta_v^i - \theta_v^{f(v)}\right).$$

For $b_{c\geq3}$ we will use the summand $b(S')$ instead of $\omega(E(f_{S'}))$ that we used for $b_{c=2}$. Intuitively, the sum $\sum_{v\in S'}(\max_{i\neq f(v)}\theta_v^i - \theta_v^{f(v)})$ is an overestimation of the improvement for properly colored edges with exactly one endpoint in S', the term $\binom{|S'|}{2}\cdot\omega_{\max}$ overestimates the properly colored edges inside S', and the remaining terms overestimate the properly colored edges outside S'.

Analogously to $b_{c=2}$, for every $v \in V \setminus S'$, we define a value $\alpha_v := \max_{i\neq f(v)}(\omega_v^i - \omega_v^{f(v)}) + \beta_v$ with

$$\beta_v := \sum_{e\in E(v,S')} 2\cdot|\omega(e)|,$$

and let $Y \subseteq V \setminus S'$ be a set containing the $k - |S'|$ vertices with biggest α_v-values from $V \setminus S'$. We define the upper bound by

$$b_{c\geq3}(I, S') := b(S') + \sum_{v\in Y}\alpha_v + 2\binom{k-|S'|}{2}\cdot\omega_{\max}$$

and show that it is in fact an upper bound.

Proposition 2. *If $c \geq 3$, then $b_{c\geq3}(I, S')$ is an upper bound.*

Proof. Let f' be a coloring with $S' \subset D(f, f')$ and $d(f, f') = k$ and let $X := D(f, f') \setminus S'$. We show $\omega(E(f')) \leq b_{c\geq3}(I, S')$. To this end, we let $f|_A$ for each $A \subseteq S' \cup X$ denote the coloring defined by

$$f|_A(v) := \begin{cases} f'(v) & \text{if } v \in A\text{, and} \\ f(v) & \text{if } v \notin A. \end{cases}$$

To show $\omega(E(f')) \leq b_{c\geq3}(I, S')$ we analyze how $\omega(E(f'))$ differs from $\omega(E(f|_{S'}))$.

$$\begin{aligned}
&\omega(E(f')) \\
&\leq \omega(E(f|_{S'})) + \underbrace{\sum_{v\in X}(\omega_v^{f'(v)} - \omega_v^{f(v)})}_{(1)} \\
&\quad + \underbrace{\sum_{e\in E(S',\,X)} 2\cdot|\omega(e)|}_{(2)} + \underbrace{\sum_{e\in E(X)} 2\cdot|\omega(e)|}_{(3)}
\end{aligned}$$

By adding (1) to $\omega(E(f|_{S'}))$, we added the weight of all properly colored edges if only v changes its color for every vertex $v \in X$. The difference between $\omega(E(f'))$

and $\omega(E(f|_{S'}))$ + (1) then consists of all edge-weights that were falsely counted in (1) since both endpoints were moved. To compensate this, the summands (2) and (3) were added. Summand (2) overestimates the weight of falsely counted edges with one endpoint in X and one endpoint in S', while (3) overestimates the weight of falsely counted edges with both endpoints in X. Observe that every falsely counted edge weight may be counted for both of its endpoints within $\omega(E(f|_{S'}))$ + (1). Thus, each edge weight in (2) and (3) needs to be multiplied by 2.

Note that (1) + (2) $\leq \sum_{v\in X}\alpha_v \leq \sum_{v\in Y}\alpha_v$ and that (3) $\leq 2\binom{k-|S'|}{2}\cdot\omega_{\max}$. Therefore, it remains to show that $\omega(E(f|_{S'}))\leq b(S')$. To this end, note that $\omega(E(f|_{S'}))$ can be expressed by the sum of $\omega(E(f))$, improvement of the weight of properly colored edges inside S' between f and $f|_{S'}$, and $\sum_{v\in S'}(\theta_v^{f'(v)}-\theta_v^{f(v)})$:

$$\omega(E(f|_{S'})) = \omega(E(f)) + \sum_{\substack{e\in E(S')\\ e\in E(f|_{S'})}}\omega(e) - \sum_{\substack{e\in E(S')\\ e\in E(f)}}\omega(e) + \sum_{v\in S'}(\theta_v^{f'(v)}-\theta_v^{f(v)}).$$

Since $\binom{|S'|}{2}\cdot\omega_{\max}$ is at least as big as the sum of the weights of properly edges inside S' under $f|_{S'}$, we conclude $\omega(E(f|_{S'}))\leq b(S')$. Hence, $b_{c\geq 3}$ is an upper bound. □

Prevention of Redundant Flips

We introduce further speed-up techniques that we used in our implementation of the hill-climbing algorithm. Roughly speaking, the idea behind these speed-up techniques is to exclude vertices that are not contained in an improving flip $D(f,f')$ of any k-neighbor f' of f. To this end, we introduce the *auxiliary vertex set* V_k containing all remaining vertices that are potentially part of an improving flip of a k-neighbor of f.

Initially, V_k contains all vertices of G. It is easy to see that all vertices x that are (i,k)-blocked for all $i\neq f(x)$ can be removed from V_k if each edge of G has weight 1. This also holds for general instances when considering an extension of the definition of (i,k)-blocked vertices for arbitrary weight functions. Moreover, whenever our algorithm is able to verify that a vertex v is in no improving flip $D(f,f')$, we may remove v from V_k.

If at any time our algorithm replaces the current coloring f by a better coloring f', we continue by searching for an improving k-neighbor of the new coloring f'. Instead of initializing a new auxiliary vertex set V_k with all of V, we only consider the remaining vertices of V_k and the vertices that have a small distance to $D(f,f')$. This idea is formalized within the next lemma.

Lemma 3. *Let $G=(V,E)$ be a graph, let $\omega: E\to\mathbb{Q}$ be an edge weight function, and let k be an integer. Moreover, let f and f' be $(k-1)$-optimal c-colorings of G and let v be a vertex of distance at least $k+1$ to each vertex of $D(f,f')$. If there is no improving k-neighbor $\hat{f}$ of f with $v\in D(f,\hat{f})$, then there is no improving k-neighbor $\tilde{f}$ of f' with $v\in D(f',\tilde{f})$.*

Proof. We prove the lemma by contraposition. Let $\widetilde{f}$ be an improving k-neighbor of f' with $v\in D(f',\widetilde{f})$. We show that there is an improving k-neighbor $\widehat{f}$ of f with $v\in D(f,\widehat{f})$.

For a vertex $w\in V$ we set

$$\widehat{f}(w) := \begin{cases}\widetilde{f}(w) & \text{if } w\in D(f',\widetilde{f}),\text{ and}\\ f(w) & \text{if } w\notin D(f',\widetilde{f}),\text{ otherwise.}\end{cases}$$

Observe that $v\in d(f,\widehat{f})$. Moreover, $\widehat{f}$ and f differ on at most $d(f',\widetilde{f})\leq k$ positions and therefore, $\widehat{f}$ is a k-neighbor of f. It remains to show that $\widehat{f}$ is improving. To this end, we define the edge set $X\subseteq E$ as the set of all edges with at least one endpoint in $D(f',\widetilde{f})$. Consider the following claim about properly colored edges.

Claim 1. *It holds that*

a) $E(\widetilde{f})\setminus X = E(f')\setminus X$ *and* $E(\widehat{f})\setminus X = E(f)\setminus X$ *,*
b) $E(\widetilde{f})\cap X = E(\widehat{f})\cap X$ *and* $E(f)\cap X = E(f')\cap X$.

Proof. a) Let $e\in E\setminus X$. Note that both endpoints of e are elements of $V\setminus D(f',\widetilde{f})$. Thus, the endpoints of e have distinct colors under $\widetilde{f}$ if and only if they have distinct colors under f'. Consequently, we have $E(\widetilde{f})\setminus X = E(f')\setminus X$. Furthermore, by to the construction of $\widehat{f}$ we have $D(f,\widehat{f}) = D(f',\widetilde{f})$, which implies $E(\widehat{f})\setminus X = E(f)\setminus X$.

$b)$ Since f' is $(k-1)$-optimal and $\widetilde{f}$ is an improving k-neighbor of f', the set $D(f',\widetilde{f})$ contains exactly k vertices. Thus, we may assume by Lemma 2 that $D(f',\widetilde{f})$ is connected. This implies that $D(f',\widetilde{f})\subseteq N^{k-1}[v]$, since $v\in D(f',\widetilde{f})$.

Let $e\in X$. Since $D(f',\widetilde{f})\subseteq N^{k-1}[v]$, both endpoints of e have distance at most k from v. Together with the fact that v has distance at least $k+1$ from $D(f,f')$, this implies that both endpoints of e do not belong to $D(f,f')$. Therefore, we have $f(u)=f'(u)$ for each endpoint u of e. Consequently, the endpoints of e have different colors under f if and only if they have different colors underand f', which implies $E(f)\cap X = E(f')\cap X$.

By the definition of $\widehat{f}$, the fact that $f(u)=f'(u)$ for each endpoint u of e also implies that $\widehat{f}(u)=\widetilde{f}(u)$, which implies $E(\widetilde{f})\cap X = E(\widehat{f})\cap X$. ◇

We next use Claim 1 to show that $\widehat{f}$ is an improving neighbor of f. Since $\widetilde{f}$ is an improving neighbor of f' we have $\omega(E(\widetilde{f}))>\omega(E(f'))$, which implies

$$\omega(E(\widetilde{f})\cap X)+\omega(E(\widetilde{f})\setminus X) > \omega(E(f')\cap X)+\omega(E(f')\setminus X).$$

Together with Claim 1 $a)$ we then have

$$\omega(E(\widetilde{f})\cap X) > \omega(E(f')\cap X).$$

By applying the equations form Claim 1 b) we have

$$\omega(E(\widehat{f}) \cap X) > \omega(E(f) \cap X).$$

Finally, since $E(\widehat{f}) \setminus X = E(f) \setminus X$ by Claim 1 a), we may add the weights of all edges in $E(\widehat{f}) \setminus X$ to the left side of the inequality and the weight of all edges in $E(f) \setminus X$ to the right side. We end up with the inequality $\omega(E(\widehat{f})) > \omega(E(f))$ and therefore, $\widehat{f}$ is an improving k-neighbor of f. □

We next describe how we exploit Lemma 3 in our implementation: We start with a coloring f and search for improving k-neighbors of f for increasing values of k starting with $k = 1$. Whenever we find an improving neighbor f' of f we continue by searching for an improving neighbor f'' of f' starting with $k = 1$ again. We use Lemma 3 so that we do not have to initialize the set V_k by $V_k := V$ again if we already encountered a $(k-1)$-optimal coloring previously. More precisely, if we want to find an improving k-neighbor for a $(k-1)$-optimal coloring f', we take the last previously encountered $(k-1)$-optimal coloring f and add only the vertices of distance at most k to $D(f, f')$ to the current vertices in V_k instead of setting V_k back to V. This is correct since every vertex, which is not in V_k, is not part of any improving k-flip of f and therefore according to Lemma 3 the only vertices outside of V_k that can possibly be in an improving k-flip of f' are those with distance at most k to $D(f, f')$.

Next, we provide a further technique to identify vertices that can be removed from V_k. The intuitive idea behind this technique can be explained as follows: if a vertex can be excluded from V_k, then all 'copies' of this vertex can also be excluded. To specify the term 'copies', we provide the following definition.

Definition 3. *Let $G = (V, E)$ be a graph, let $\omega : E \to \mathbb{Q}$ be an edge weight function. Two vertices v and w of G are* weighted twins *if $N(v) \setminus \{w\} = N(w) \setminus \{v\}$ and $\omega(\{v, x\}) = \omega(\{w, x\})$ for each $x \in N(v) \setminus \{w\}$.*

The next lemma shows that if our algorithm excludes a vertex v from the enumeration, then we can also exclude w if v and w are weighted twins with $f(v) = f(w)$.

Lemma 4. *Let $G = (V, E)$ be a graph, let $\omega : E \to \mathbb{Q}$ be an edge weight function, and let k be an integer. Moreover, let f be a c-coloring of G and let v and w be weighted twins in G with $f(v) = f(w)$. If there is no improving k-neighbor f' of f with $v \in D(f, f')$, then there is no improving k-neighbor $\tilde{f}$ of f with $w \in D(f, \tilde{f})$.*

Proof. Assume towards a contradiction that there is an improving k-neighbor $\tilde{f}$ of f with $w \in D(f, \tilde{f})$. By assumption, $v \notin D(f, \tilde{f})$. Let f' be the c-coloring that agrees with $\tilde{f}$ on $V \setminus \{v, w\}$ and where $f'(v) := \tilde{f}(w)$ and $f'(w) := \tilde{f}(v) = f(v)$. Recall that $\omega(\{v, x\}) = \omega(\{w, x\})$ for each $x \in N(v) \cap N(w)$ and that $N(v) \setminus \{w\} = N(w) \setminus \{v\}$. For each $x \in N(v) \cap N(w)$, let $E_x := \{\{v, x\}, \{w, x\}\}$. Note that since $\tilde{f}(v) \neq \tilde{f}(w)$, at least one edge of E_x is contained in $E(\tilde{f})$. If both edges of E_x are contained in $E(\tilde{f})$, then $f'(x) = \tilde{f}(x) \notin \{f'(v), f'(w)\}$ and thus both edges of E_x are contained in $E(f')$. If only one edge of E_x is contained in $E(\tilde{f})$, then $E(f')$ contains exactly the other edge of E_x since $f'(v) = \tilde{f}(w)$, $f'(w) = \tilde{f}(v)$, and $f'(x) = \tilde{f}(x)$. Since the weight of both edges of E_x are the same for each $x \in N(v) \cap N(w)$, we have $\omega(E(f')) = \omega(E(\tilde{f}))$. Hence, f' is an improving k-neighbor of f with $v \in D(f, f')$, a contradiction. □

Consequently, when our algorithm removes a vertex v from the set V_k for some k based on the knowledge that there is no improving k-neighbor f' of f with $v \in D(f, f')$, then we also remove all weighted twins of v with the same color as v from V_k.

Implementation and Experimental Results

Our hill-climbing algorithm (LS) is implemented in JAVA/Kotlin and uses the JGraphT library. The enumeration algorithm for enumerating the candidate sets is a JAVA implementation of a polynomial-delay algorithm for enumerating all connected induced subgraphs of a given size (Komusiewicz and Sommer 2021).

We used the graphs from the G-set benchmark (https://web.stanford.edu/~yyye/yyye/Gset/), an established benchmark data set for MAX c-CUT with $c \in \{2, 3, 4\}$ (and thus also for MAX CUT) (Benlic and Hao 2013; Festa et al. 2002; Ma and Hao 2017; Shylo, Glover, and Sergienko 2015; Wang et al. 2013; Zhu, Lin, and Ali 2013). The data set consists of 71 graphs with vertex-count between 800 and 20,000 and a density between 0.02% and 6%.

As starting solutions, we used the solutions computed by the MOH algorithm of Ma and Hao (2017) for each each graph of the G-set and each $c \in \{2, 3, 4\}$. For $c = 3$ and $c = 4$, these are the best known solutions for all graphs of the G-set.

For one graph (g23) and each $c \in \{2, 3, 4\}$, there is a large gap between the value of the published coloring and the stated value of the corresponding coloring(for example, for $c = 3$, the published coloring has a value of $13,275$ whereas it is stated that the coloring has a value of $17,168$). To not exploit these colorings in our evaluation, we only considered the remaining 70 graphs. These 70 graphs are of two types: 34 graphs are unit graphs (where each edge has weight 1) and 36 graphs are signed graphs (where each edge has either weight 1 or -1). For each of these graphs, we ran experiments of LS for each $c \in \{2, 3, 4\}$ and a time limit of 30 minutes with the published MOH solution as a starting solution. Additionally, for each instance we ran standard ILP-formulations for 30 minutes, once without starting solution and once with the MOH solution as starting solution. Each run of an ILP provides both a best found solution and an upper bound on the maximum value of any c-coloring for the given instance. Each experiment was performed on a single thread of an Intel(R) Xeon(R) Silver 4116 CPU with 2.1 GHz, 24 CPUs and 128 GB RAM.

The ILP-upper bounds verified the optimality of 22 MOH solutions. Thus, of the 210 instances, only 188 instances are interesting in the sense that LS or the ILP can find an im-

Table 1: The graphs from the G-set for which LS or ILP found an improved coloring, or for which we verified that the MOH colorings are globally optimal (for $c = 3$). MOH shows the value of the published solutions of (Ma and Hao 2017), LS shows the best solution our hill-climbing algorithm found. ILP shows the best solution of any of the two ILP-runs. The better coloring is bold. Finally, UB shows the better upper bound computed during the two ILP-runs. If there is no number in the LS or ILP column for an instance, then the corresponding algorithm was not able to find an improved coloring. If a UB entry is bold, then some found solution matches this upper bound, verifying the optimality of the best found solution.

data	MOH	LS	ILP	UB
g11	669	—	**671**	**671**
g12	660	661	**663**	**663**
g13	686	687	**688**	**688**
g15	3984	**3985**	**3985**	4442
g24	17162	**17163**	—	19989
g25	17163	**17164**	—	19989
g26	17154	**17155**	—	19989
g27	4020	**4021**	—	9840
g28	3973	**3975**	—	9822
g31	4003	**4005**	—	9776
g32	1653	1658	**1666**	1668
g33	1625	1628	**1636**	1640
g34	1607	1609	**1616**	1617
g35	10046	**10048**	—	11711
g37	10052	**10053**	**10053**	11691
g40	2870	**2871**	—	5471
g41	2887	**2888**	—	5452
g48	**6000**	—	—	**6000**
g49	**6000**	—	—	**6000**
g50	**6000**	—	—	**6000**
g55	12427	12429	**12432**	12498
g56	4755	**4757**	—	6157
g57	4080	4092	**4103**	4154
g59	7274	**7276**	—	14673
g61	6858	**6861**	—	8728
g62	5686	**5710**	5706	5981
g63	35315	**35318**	—	41420
g64	10429	**10437**	—	20713
g65	6489	6512	**6535**	6711
g66	7414	7442	**7443**	7843
g67	8088	8116	**8141**	9080
g70	**9999**	—	—	**9999**
g72	8190	8224	**8244**	9166
g77	11579	**11632**	11619	13101
g81	16326	**16392**	16374	18337

Table 2: The number of instances where LS and ILP found improved solutions. Column nonOpt shows for how many instances the best known MOH colorings (Ma and Hao 2017) might be supoptimal (because they). Columns LS and ILP show how many of these solutions where improved by the respective approaches. Columns I_1, I_2, and I_3 show for how many instances the first improvement was found by LS within 10 seconds, between 10 and 60 seconds, and after more than 60 seconds, respectively.

	nonOpt	I_1	I_2	I_3	LS	ILP
unit $c = 2$	31	2	1	0	3	2
unit $c = 3$	30	8	0	0	8	3
unit $c = 4$	28	5	3	1	9	4
signed $c = 2$	29	1	1	0	2	6
signed $c = 3$	36	19	2	1	22	14
signed $c = 4$	34	20	5	0	25	14
sum	188	55	12	2	69	43

proved solution. The upper bounds also verified the optimality of 8 further improved solutions found by LS or ILP.

In total, the ILP found better colorings than the MOH coloring for 43 of the 188 instances. In comparison, our hill-climbing algorithm was able to find better solutions than the MOH solutions for 69 instances of the 188 instances. Table 1 gives the results for $c = 3$, showing those instances where the MOH coloring was verified to be optimal by the ILP or where LS or the ILP found an improved coloring. The full overview for $c \in \{2, 3, 4\}$ is shown in Table 3, Table 4, and Table 5.

Overall, on 35 instances, both LS and the ILP found improved colorings compared to the MOH coloring. On 7 of these instances, LS and the ILP find colorings of equal value, on 22 the ILP finds a better solution than LS, and on 6 LS finds a better solution than the ILP. Thus, the ILP usually finds better colorings, whenever both approaches found an improved coloring. For $c > 2$, both approaches find new record colorings. More precisely, for 23 instances, only the ILP found a new record coloring; for 6 instances, both approaches found a record coloring, and for 38 instances only LS found a record coloring. This shows that LS finds improvements also for very hard instances on which MOH provided the best known solutions so far.

For all instances where LS was able to find a better solution than the MOH solution, the average time to find the first improving flip was 15.17 seconds. For an overview on the number of improved instances and the time when LS found the first improvement, see Table 2. It is also interesting to see for which value of k, the first improvement was found (in other words, the smallest value k such that the MOH solutions are not k-flip optimal). On average, this value was 3.39.

We summarize our experimantal findings as follows. First, parameterized local search can be used successfully as a post-processing for state-of-the-art heuristics for MAX c-CUT, in many cases leading to new record solutions for $c > 2$. Second, the number of instances where an improvement was found is larger for LS than for ILP approaches. Third, to find improved solutions, it is sometimes necessary to explore

k-flip neighborhoods for larger values of k. Finally, this can be done within an acceptable amount of time by using our algorithm for LS-MAX c-CUT and our speedup strategies.

References

Benlic, U.; and Hao, J. 2013. Breakout Local Search for the Max-Cutproblem. *Eng. Appl. Artif. Intell.*, 26(3): 1162–1173.

Bonnet, É.; Iwata, Y.; Jansen, B. M. P.; and Kowalik, L. 2019. Fine-Grained Complexity of k-OPT in Bounded-Degree Graphs for Solving TSP. In *Proceedings of the 27th Annual European Symposium on Algorithms (ESA '19)*, volume 144 of *LIPIcs*, 23:1–23:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Cygan, M.; Fomin, F. V.; Kowalik, L.; Lokshtanov, D.; Marx, D.; Pilipczuk, M.; Pilipczuk, M.; and Saurabh, S. 2015. *Parameterized Algorithms*. Springer.

Dörnfelder, M.; Guo, J.; Komusiewicz, C.; and Weller, M. 2014. On the parameterized complexity of consensus clustering. *Theoretical Computer Science*, 542: 71–82.

Downey, R. G.; and Fellows, M. R. 2013. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer.

Fellows, M. R.; Fomin, F. V.; Lokshtanov, D.; Rosamond, F. A.; Saurabh, S.; and Villanger, Y. 2012. Local search: Is brute-force avoidable? *Journal of Computer and System Sciences*, 78(3): 707–719.

Festa, P.; Pardalos, P. M.; Resende, M. G. C.; and Ribeiro, C. C. 2002. Randomized heuristics for the Max-Cut problem. *Optimization Methods and Software*, 17(6): 1033–1058.

Frieze, A. M.; and Jerrum, M. 1997. Improved Approximation Algorithms for MAX k-CUT and MAX BISECTION. *Algorithmica*, 18(1): 67–81.

Garey, M. R.; and Johnson, D. S. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman.

Gaspers, S.; Kim, E. J.; Ordyniak, S.; Saurabh, S.; and Szeider, S. 2012. Don't Be Strict in Local Search! In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence (AAAI '12)*. AAAI Press.

Gaur, D. R.; Krishnamurti, R.; and Kohli, R. 2009. Conflict Resolution in the Scheduling of Television Commercials. *Operations Research*, 57(5): 1098–1105.

Guo, J.; Hartung, S.; Niedermeier, R.; and Suchý, O. 2013. The Parameterized Complexity of Local Search for TSP, More Refined. *Algorithmica*, 67(1): 89–110.

Guo, J.; Hermelin, D.; and Komusiewicz, C. 2014. Local search for string problems: Brute-force is essentially optimal. *Theoretical Computer Science*, 525: 30–41.

Jensen, T. R.; and Toft, B. 2011. *Graph coloring problems*, volume 39. John Wiley & Sons.

Kann, V.; Khanna, S.; Lagergren, J.; and Panconesi, A. 1997. On the Hardness of Approximating Max k-Cut and its Dual. *Chicago Journal of Theoretical Computer Science*, 1997(2).

Karp, R. M. 1972. Reducibility Among Combinatorial Problems. In *Proceedings of a Symposium on the Complexity of Computer Computations*, The IBM Research Symposia Series, 85–103. Plenum Press, New York.

Katzmann, M.; and Komusiewicz, C. 2017. Systematic Exploration of Larger Local Search Neighborhoods for the Minimum Vertex Cover Problem. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI '17)*, 846–852. AAAI Press.

Komusiewicz, C.; and Sommer, F. 2021. Enumerating connected induced subgraphs: Improved delay and experimental comparison. *Discret. Appl. Math.*, 303: 262–282.

Komusiewicz, C.; and Sorge, M. 2015. An algorithmic framework for fixed-cardinality optimization in sparse graphs applied to dense subgraph problems. *Discrete Applied Mathematics*, 193: 145–161.

Leiserson, M. D. M.; Tatar, D.; Cowen, L. J.; and Hescott, B. J. 2011. Inferring Mechanisms of Compensation from E-MAP and SGA Data Using Local Search Algorithms for Max Cut. *J. Comput. Biol.*, 18(11): 1399–1409.

Ma, F.; and Hao, J. 2017. A multiple search operator heuristic for the max-k-cut problem. *Annals of Operations Research*, 248(1-2): 365–403.

Marx, D. 2008. Searching the k-change neighborhood for TSP is W[1]-hard. *Operations Research Letters*, 36(1): 31–36.

Papadimitriou, C. H.; and Yannakakis, M. 1991. Optimization, Approximation, and Complexity Classes. *Journal of Computer and System Sciences*, 43(3): 425–440.

Schäffer, A. A.; and Yannakakis, M. 1991. Simple Local Search Problems That are Hard to Solve. *SIAM Journal on Computing*, 20(1): 56–87.

Shylo, V.; Glover, F.; and Sergienko, I. 2015. Teams of global equilibrium search algorithms for solving the weighted maximum cut problem in parallel. *Cybernetics and Systems Analysis*, 51(1): 16–24.

Subramanian, A. P.; Gupta, H.; Das, S. R.; and Cao, J. 2008. Minimum Interference Channel Assignment in Multiradio Wireless Mesh Networks. *IEEE Transactions on Mobile Computing*, 7(12): 1459–1473.

Szeider, S. 2011. The parameterized complexity of k-flip local search for SAT and MAX SAT. *Discrete Optimization*, 8(1): 139–145.

Vredeveld, T.; and Lenstra, J. K. 2003. On local search for the generalized graph coloring problem. *Operations Research Letters*, 31(1): 28–34.

Wang, Y.; Lü, Z.; Glover, F.; and Hao, J.-K. 2013. Probabilistic GRASP-tabu search algorithms for the UBQP problem. *Computers & Operations Research*, 40(12): 3100–3107.

Zhu, W.; Lin, G.; and Ali, M. M. 2013. Max-k-Cut by the Discrete Dynamic Convexized Method. *INFORMS Journal on Computing*, 25(1): 27–40.

Table 3: The solutions of the best found c-coloring for any of the G-set graphs for $c = 2$. The column MOH shows the value of the published solutions of (Ma and Hao 2017). The column LS shows the best solution our hill-climbing algorithm found. The column ILP shows the best solution of any of the two ILP-runs. Finally, column UB shows the best upper bound for any solution of the corresponding graph our ILP was able to find. If there is no number in the LS or ILP column for an instance, then the corresponding algorithm was not able to find a better solution than the published solutions of (Ma and Hao 2017). If the UB entry is bold for some instance, then some found solution matches this upper bound, that is, the optimality of the best found solution was verified.

data	n	m	MOH	LS	ILP	UB	ILP_1	UB_1	ILP_2	UB_2
g1	800	19176	**11624**	—	—	16188	—	16188	—	16225
g2	800	19176	**11620**	—	—	15915	—	15915	—	16254
g3	800	19176	**11622**	—	—	15766	—	15766	—	16058
g4	800	19176	**11646**	—	—	16059	—	16217	—	16059
g5	800	19176	**11631**	—	—	16182	—	16182	—	16220
g6	800	19176	**2178**	—	—	6387	—	6627	—	6387
g7	800	19176	**2006**	—	—	6225	—	6339	—	6225
g8	800	19176	**2005**	—	—	6209	—	6209	—	6215
g9	800	19176	**2054**	—	—	6106	—	6106	—	6379
g10	800	19176	**2000**	—	—	6315	—	6315	—	6322
g11	800	1600	**564**	—	—	**564**	—	**564**	—	**564**
g12	800	1600	**556**	—	—	**556**	—	**556**	—	**556**
g13	800	1600	**582**	—	—	**582**	—	**582**	—	**582**
g14	800	4694	**3064**	—	—	3158	—	3158	—	3158
g15	800	4661	**3050**	—	—	3139	—	3148	—	3139
g16	800	4672	**3052**	—	—	3144	—	3144	—	3148
g17	800	4667	**3047**	—	—	3144	—	3144	—	3153
g18	800	4694	**992**	—	—	1137	—	1140	—	1137
g19	800	4661	**906**	—	—	1044	—	1046	—	1044
g20	800	4672	**941**	—	—	1069	—	1074	—	1069
g21	800	4667	**931**	—	—	1072	—	1074	—	1072
g22	2000	19990	**13359**	—	—	17677	—	17888	—	17677
g24	2000	19990	**13337**	—	—	17905	—	17905	—	18070
g25	2000	19990	**13340**	—	—	17993	—	17993	—	18049
g26	2000	19990	**13328**	—	—	17744	—	18236	—	17744
g27	2000	19990	**3341**	—	—	8273	—	8394	—	8273
g28	2000	19990	**3298**	—	—	7505	—	8194	—	7505
g29	2000	19990	**3405**	—	—	7594	—	7594	—	8382
g30	2000	19990	**3413**	—	—	8033	—	8033	—	8354
g31	2000	19990	**3310**	—	—	7688	—	8253	—	7688
g32	2000	4000	**1410**	—	—	**1410**	—	**1410**	—	**1410**
g33	2000	4000	**1382**	—	—	**1382**	—	**1382**	—	**1382**
g34	2000	4000	**1384**	—	—	**1384**	—	**1384**	—	**1384**
g35	2000	11778	**7687**	—	—	8985	—	9242	—	8985
g36	2000	11766	**7680**	—	—	9125	—	9199	—	9125
g37	2000	11785	**7691**	—	—	9056	—	9056	—	9265
g38	2000	11779	**7688**	—	—	8923	—	8923	—	9130
g39	2000	11778	**2408**	—	—	3214	—	3254	—	3214
g40	2000	11766	**2400**	—	—	3157	—	3157	—	3218
g41	2000	11785	**2405**	—	—	3196	—	3196	—	3199
g42	2000	11779	**2481**	—	—	3228	—	3228	—	3229
g43	1000	9990	**6660**	—	—	8055	—	8264	—	8055
g44	1000	9990	**6650**	—	—	8228	—	8228	—	8287
g45	1000	9990	**6654**	—	—	8146	—	8182	—	8146
g46	1000	9990	**6649**	—	—	8148	—	8168	—	8148
g47	1000	9990	**6657**	—	—	8075	—	8146	—	8075
g48	3000	6000	**6000**	—	—	**6000**	—	**6000**	—	**6000**
g49	3000	6000	**6000**	—	—	**6000**	—	**6000**	—	**6000**
g50	3000	6000	**5880**	—	—	**5880**	—	**5880**	—	**5880**
g51	1000	5909	**3848**	—	—	3992	—	4160	—	3992
g52	1000	5916	**3851**	—	—	4095	—	4111	—	4095
g53	1000	5914	**3850**	—	—	3971	—	3971	—	4090
g54	1000	5916	**3852**	—	—	4073	—	4073	—	4082
g55	5000	12498	**10299**	—	—	11692	—	11692	—	11755
g56	5000	12498	**4016**	—	—	5378	—	5378	—	5418
g57	5000	10000	**3494**	—	—	**3494**	—	**3494**	—	**3494**
g58	5000	29570	19289	**19290**	**19290**	24833	—	24833	**19290**	25197
g59	5000	29570	**6086**	—	—	10372	—	10372	—	10577
g60	7000	17148	**14190**	—	—	16528	—	16547	—	16528
g61	7000	17148	**5797**	—	—	8144	—	8144	—	8199
g62	7000	14000	4868	4870	**4872**	**4872**	**4872**	**4872**	**4872**	**4872**
g63	7000	41459	27033	**27037**	—	35046	—	35046	—	35703
g64	7000	41459	8747	**8748**	—	15137	—	15137	—	15929
g65	8000	16000	5560	—	**5562**	5568	—	5568	**5562**	5568
g66	9000	18000	6360	—	**6364**	6368	**6364**	6368	**6364**	6369
g67	10000	20000	6942	—	**6948**	6952	—	6957	**6948**	6952
g70	10000	9999	9544	9551	**9575**	9714	—	9714	**9575**	9723
g72	10000	20000	6998	—	**7004**	7013	**7004**	7013	7002	7014
g77	14000	28000	**9928**	—	—	9948	—	9948	—	9951
g81	20000	40000	14036	—	**14044**	14078	—	14078	**14044**	14080

Table 4: The solutions of the best found c-coloring for any of the G-set graphs for $c = 3$.

data	n	m	MOH	LS	ILP	UB	ILP$_1$	UB$_1$	ILP$_2$	UB$_2$
g1	800	19176	**15165**	—	—	19148	—	19148	—	19159
g2	800	19176	**15172**	—	—	19160	—	19160	—	19160
g3	800	19176	**15173**	—	—	19134	—	19134	—	19158
g4	800	19176	**15184**	—	—	19117	—	19135	—	19117
g5	800	19176	**15193**	—	—	19146	—	19146	—	19164
g6	800	19176	**2632**	—	—	9441	—	9453	—	9441
g7	800	19176	**2409**	—	—	9242	—	9253	—	9242
g8	800	19176	**2428**	—	—	9228	—	9228	—	9230
g9	800	19176	**2478**	—	—	9261	—	9275	—	9261
g10	800	19176	**2407**	—	—	9233	—	9233	—	9267
g11	800	1600	669	—	**671**	**671**	**671**	**671**	**671**	674
g12	800	1600	660	661	**663**	**663**	**663**	**663**	**663**	**663**
g13	800	1600	686	687	**688**	**688**	**688**	**688**	**688**	**688**
g14	800	4694	**4012**	—	—	4497	—	4497	—	4510
g15	800	4661	3984	**3985**	**3985**	4442	—	4442	**3985**	4474
g16	800	4672	**3990**	—	—	4458	—	4465	—	4458
g17	800	4667	**3983**	—	—	4413	—	4413	—	4457
g18	800	4694	**1207**	—	—	1962	—	1962	—	1991
g19	800	4661	**1081**	—	—	1833	—	1859	—	1833
g20	800	4672	**1122**	—	—	1888	—	1888	—	1889
g21	800	4667	**1109**	—	—	1827	—	1827	—	1875
g22	2000	19990	**17167**	—	—	19989	—	19989	—	19989
g24	2000	19990	17162	**17163**	—	19989	—	19989	—	19989
g25	2000	19990	17163	**17164**	—	19989	—	19989	—	19989
g26	2000	19990	17154	**17155**	—	19989	—	19990	—	19989
g27	2000	19990	4020	**4021**	—	9840	—	9841	—	9840
g28	2000	19990	3973	**3975**	—	9822	—	9827	—	9822
g29	2000	19990	**4106**	—	—	9947	—	9948	—	9947
g30	2000	19990	**4117**	—	—	9929	—	9929	—	9933
g31	2000	19990	4003	**4005**	—	9776	—	9861	—	9776
g32	2000	4000	1653	1658	**1666**	1668	**1666**	1668	1664	1670
g33	2000	4000	1625	1628	**1636**	1640	**1636**	1640	**1636**	1640
g34	2000	4000	1607	1609	**1616**	1617	**1616**	1617	1615	1618
g35	2000	11778	10046	**10048**	—	11711	—	11711	—	11714
g36	2000	11766	**10039**	—	—	11702	—	11702	—	11703
g37	2000	11785	10052	**10053**	**10053**	11691	—	11753	**10053**	11691
g38	2000	11779	**10040**	—	—	11703	—	11745	—	11703
g39	2000	11778	**2903**	—	—	5457	—	5457	—	5551
g40	2000	11766	2870	**2871**	—	5471	—	5471	—	5471
g41	2000	11785	2887	**2888**	—	5452	—	5472	—	5452
g42	2000	11779	**2980**	—	—	5551	—	5567	—	5551
g43	1000	9990	**8573**	—	—	9985	—	9985	—	9988
g44	1000	9990	**8571**	—	—	9957	—	9957	—	9980
g45	1000	9990	**8566**	—	—	9983	—	9983	—	9986
g46	1000	9990	**8568**	—	—	9983	—	9985	—	9983
g47	1000	9990	**8572**	—	—	9966	—	9966	—	9983
g48	3000	6000	**6000**	—	—	**6000**	—	**6000**	—	**6000**
g49	3000	6000	**6000**	—	—	**6000**	—	**6000**	—	**6000**
g50	3000	6000	**6000**	—	—	**6000**	—	**6000**	—	**6000**
g51	1000	5909	**5037**	—	—	5708	—	5712	—	5708
g52	1000	5916	**5040**	—	—	5703	—	5703	—	5726
g53	1000	5914	**5039**	—	—	5694	—	5694	—	5746
g54	1000	5916	**5036**	—	—	5667	—	5667	—	5682
g55	5000	12498	12427	12429	**12432**	12498	—	12498	**12432**	12498
g56	5000	12498	4755	**4757**	—	6157	—	6157	—	6176
g57	5000	10000	4080	4092	**4103**	4154	—	4154	**4103**	4176
g58	5000	29570	**25195**	—	—	29556	—	29556	—	29560
g59	5000	29570	7274	**7276**	—	14673	—	14673	—	14678
g60	7000	17148	**17075**	—	—	17148	—	17148	—	17148
g61	7000	17148	6858	**6861**	—	8728	—	8728	—	8735
g62	7000	14000	5686	**5710**	5706	5981	—	6033	5706	5981
g63	7000	41459	35315	**35318**	—	41420	—	41420	—	41435
g64	7000	41459	10429	**10437**	—	20713	—	20747	—	20713
g65	8000	16000	6489	6512	**6535**	6711	—	6711	**6535**	6970
g66	9000	18000	7414	7442	**7443**	7843	—	7843	**7443**	8246
g67	10000	20000	8088	8116	**8141**	9080	—	9089	**8141**	9080
g70	10000	9999	**9999**	—	—	**9999**	—	**9999**	—	**9999**
g72	10000	20000	8190	8224	**8244**	9166	—	9243	**8244**	9166
g77	14000	28000	11579	**11632**	11619	13101	—	13101	11619	13104
g81	20000	40000	16326	**16392**	16374	18337	—	18515	16374	18337

Table 5: The solutions of the best found c-coloring for any of the G-set graphs for $c = 4$.

data	n	m	MOH	LS	ILP	UB	ILP_1	UB_1	ILP_2	UB_2
g1	800	19176	**16803**	—	—	19176	—	19176	—	19176
g2	800	19176	**16809**	—	—	19176	—	19176	—	19176
g3	800	19176	**16806**	—	—	19176	—	19176	—	19176
g4	800	19176	**16814**	—	—	19176	—	19176	—	19176
g5	800	19176	**16816**	—	—	19176	—	19176	—	19176
g6	800	19176	**2751**	—	—	9544	—	9544	—	9583
g7	800	19176	**2515**	—	—	9343	—	9430	—	9343
g8	800	19176	**2525**	—	—	9397	—	9423	—	9397
g9	800	19176	**2585**	—	—	9410	—	9410	—	9477
g10	800	19176	**2510**	—	—	9380	—	9429	—	9380
g11	800	1600	**677**	—	—	**677**	—	**677**	—	**677**
g12	800	1600	664	—	**665**	**665**	**665**	**665**	**665**	**665**
g13	800	1600	**690**	—	—	**690**	—	**690**	—	**690**
g14	800	4694	**4440**	—	—	4670	—	4671	—	4670
g15	800	4661	**4406**	—	—	4622	—	4622	—	4644
g16	800	4672	**4415**	—	—	4630	—	4635	—	4630
g17	800	4667	**4411**	—	—	4625	—	4636	—	4625
g18	800	4694	1261	**1262**	**1262**	2122	—	2140	**1262**	2122
g19	800	4661	**1121**	—	—	2045	—	2050	—	2045
g20	800	4672	**1168**	—	—	2049	—	2049	—	2074
g21	800	4667	**1155**	—	—	2023	—	2052	—	2023
g22	2000	19990	**18776**	—	—	19990	—	19990	—	19990
g24	2000	19990	18769	**18772**	—	19990	—	19990	—	19990
g25	2000	19990	18775	**18776**	—	19990	—	19990	—	19990
g26	2000	19990	18767	**18770**	—	19990	—	19990	—	19990
g27	2000	19990	4201	**4202**	—	9928	—	9951	—	9928
g28	2000	19990	4150	**4157**	—	9888	—	9888	—	9919
g29	2000	19990	4293	**4294**	—	10010	—	10022	—	10010
g30	2000	19990	4305	**4308**	—	10019	—	10019	—	10024
g31	2000	19990	4171	**4176**	—	9910	—	9914	—	9910
g32	2000	4000	1669	1671	**1679**	**1679**	**1679**	**1679**	**1679**	**1679**
g33	2000	4000	1638	1640	**1644**	**1644**	**1644**	**1644**	**1644**	**1644**
g34	2000	4000	1616	1617	**1623**	**1623**	**1623**	**1623**	**1623**	1625
g35	2000	11778	**11111**	—	—	11775	—	11776	—	11775
g36	2000	11766	11108	—	**11109**	11763	—	11763	**11109**	11764
g37	2000	11785	11117	**11118**	—	11785	—	11785	—	11785
g38	2000	11779	11108	**11109**	—	11778	—	11778	—	11778
g39	2000	11778	3006	**3007**	—	5736	—	5794	—	5736
g40	2000	11766	2976	**2978**	—	5665	—	5669	—	5665
g41	2000	11785	2983	**2986**	2984	5751	—	5751	2984	5758
g42	2000	11779	3092	**3095**	—	5787	—	5800	—	5787
g43	1000	9990	9376	**9377**	**9377**	9990	—	9990	**9377**	9990
g44	1000	9990	**9379**	—	—	9990	—	9990	—	9990
g45	1000	9990	9376	**9377**	**9377**	9990	—	9990	**9377**	9990
g46	1000	9990	**9378**	—	—	9990	—	9990	—	9990
g47	1000	9990	**9381**	—	—	9990	—	9990	—	9990
g48	3000	6000	**6000**	—	—	**6000**	—	**6000**	—	**6000**
g49	3000	6000	**6000**	—	—	**6000**	—	**6000**	—	**6000**
g50	3000	6000	**6000**	—	—	**6000**	—	**6000**	—	**6000**
g51	1000	5909	5571	**5572**	**5572**	5871	—	5881	**5572**	5871
g52	1000	5916	**5584**	—	—	5891	—	5891	—	5891
g53	1000	5914	**5574**	—	—	5887	—	5887	—	5888
g54	1000	5916	**5579**	—	—	5889	—	5889	—	5889
g55	5000	12498	**12498**	—	—	**12498**	—	**12498**	—	**12498**
g56	5000	12498	4931	**4935**	—	6213	—	6213	—	6213
g57	5000	10000	4112	4132	**4145**	4220	4141	4305	**4145**	4220
g58	5000	29570	**27885**	—	—	29569	—	29569	—	29570
g59	5000	29570	7539	**7546**	—	14731	—	14731	—	14731
g60	7000	17148	**17148**	—	—	**17148**	—	**17148**	—	**17148**
g61	7000	17148	7110	**7114**	—	8748	—	8751	—	8748
g62	7000	14000	5743	5758	**5788**	6534	5774	6534	**5788**	6541
g63	7000	41459	39083	**39089**	—	41459	—	41459	—	41459
g64	7000	41459	10814	**10819**	—	20775	—	20775	—	20792
g65	8000	16000	6534	6561	**6579**	7256	6573	7256	**6579**	7349
g66	9000	18000	7474	7495	**7522**	8497	7505	8497	**7522**	8500
g67	10000	20000	8155	8185	**8220**	9299	—	9299	**8220**	9303
g70	10000	9999	**9999**	—	—	**9999**	—	**9999**	—	**9999**
g72	10000	20000	8264	8296	**8337**	9357	—	9357	**8337**	9376
g77	14000	28000	11674	**11712**	11691	13296	—	13296	11691	13455
g81	20000	40000	16470	**16525**	16485	19088	—	19088	16485	19580

Table 6: For each instance, the largest values of k for which an improving flip was found.

	2	3	4	5	6	7	8	9	10	11	12	13
unit $c = 2$	0	0	0	2	0	0	0	0	0	0	0	1
unit $c = 3$	3	1	3	0	0	1	0	0	0	0	0	0
unit $c = 4$	2	0	3	3	1	0	0	0	0	0	0	0
signed $c = 2$	0	0	1	0	0	1	0	0	0	0	0	0
signed $c = 3$	0	4	0	2	3	1	1	8	3	0	0	0
signed $c = 4$	1	2	4	3	6	5	4	0	0	0	0	0
sum	6	7	11	10	10	8	5	8	3	0	0	1

Table 7: The values of k for which the first improving flip was found.

	2	3	4	5	6	7	8	9	10
unit $c = 2$	1	0	0	2	0	0	0	0	0
unit $c = 3$	4	3	1	0	0	0	0	0	0
unit $c = 4$	5	0	1	2	1	0	0	0	0
signed $c = 2$	0	0	1	0	0	1	0	0	0
signed $c = 3$	7	10	1	3	0	0	0	0	1
signed $c = 4$	3	13	5	2	2	0	0	0	0
sum	20	26	9	9	3	1	0	0	1