Parameterized Local Search for Max c-Cut

Jaroslav Garvardt[1] , **Niels Grüttemeier**[2] , **Christian Komusiewicz**[1] and **Nils Morawietz**[1]

[1]Friedrich Schiller University Jena, Institute of Computer Science, Germany
[2]Fraunhofer Institute of Optronics, System Technologies and Image Exploitation, Germany

[1]{c.komusiewicz, jaroslav.garvardt, nils.morawietz}@uni-jena.de
[2]niels.gruettemeier@iosb-ina.fraunhofer.de

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 distinctly 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 also 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, for all $c \geq 2$, LS MAX c-CUT presumably cannot be solved in $g(k) \cdot n^{\mathcal{O}(1)}$ time even on bipartite graphs. We then show an algorithm for LS MAX c-CUT 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. Finally, we evaluate the practical performance of this algorithm in a hill-climbing 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.

1 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 data clustering [Chatziafratis *et al.*, 2021; Felzenszwalb *et al.*, 2022], computation of rankings [Chatziafratis *et al.*, 2021], design of experimental studies [Arbour *et al.*, 2021], sampling of public opinions in social networks [Huang *et al.*, 2017], channel assignment in wireless networks [Subramanian *et al.*, 2008], module detection in genetic interaction data [Leiserson *et al.*, 2011], and scheduling of TV commercials [Gaur *et al.*, 2009]. In addition, MAX c-CUT is closely related to the energy minimization problem in Hopfield neural networks [Šíma *et al.*, 1999; Kleinberg and Tardos, 2006; Wang, 2006]. 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 an algorithmic viewpoint, even restricted cases of MAX c-CUT are 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], even on graphs with maximum degree 3 [Berman and Karpinski, 1999]. 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. While MAX c-CUT admits polynomial-time constant factor approximation algorithms [Frieze and Jerrum, 1997], there are no polynomial-time approximation schemes unless P=NP [Papadimitriou and Yannakakis, 1991], even on graphs with bounded maximum degree [Berman and Karpinski, 1999]. 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 *et al.*, 2013].

A popular heuristic approach for MAX c-CUT 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 one 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: Finding 1-optimal solutions for MAX CUT is PLS-complete on edge-weighted graphs [Schäffer and Yannakakis, 1991] and thus 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 edge weights are constant, however, a simple hill climb-

ing 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. As noted by Kleinberg and Tardos [2006], a standard algorithm for searching the k-flip neighborhood takes $\Theta(n^k \cdot m)$ time where n is the number of vertices. This led Kleinberg and Tardos to conclude that the k-flip neighborhood is impractical. In this work, we ask whether we can do better than the brute-force $\Theta(n^k \cdot m)$-time algorithm or, in other words, whether the dismissal of k-flip neighborhood may have been premature.

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 $\mathcal{O}(n^k \cdot m)$ 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 toolkit for negative results 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 *et al.*, 2013; Guo *et al.*, 2014; Gaspers *et al.*, 2012; Komusiewicz *et al.*, 2023; Marx, 2008; Szeider, 2011]. In contrast to these many, mostly negative, theoretical results, there are so far only few encouraging experimental studies [Gaspers *et al.*, 2019; Grüttemeier *et al.*, 2021; Hartung and Niedermeier, 2013; Katzmann and Komusiewicz, 2017]. The maybe most extensive positive results so far were obtained for LS VERTEX COVER where 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 parameterization by k *and* 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. 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 for the parameter k. 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. To put this running time bound into context, two aspects should be discussed. First, the NP-hardness of the special case of MAX c-CUT with $\Delta = 3$ implies that a running time of $g(\Delta) \cdot n^{\mathcal{O}(1)}$ is impossible unless P=NP. Second, only the parameter k occurs in the exponent; we say that the running time grows mildly with respect to Δ and strongly with respect to k. This is desirable as k is a user-determined parameter whereas Δ depends on the input; a broader discussion of this type of running times is given by Komusiewicz and Morawietz [2022].

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 an optimal 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 finally 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. 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).

Due to lack of space, several proofs are deferred to a full version.

2 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. 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 with $E_G(S) := E_G(S, S)$ the edges between vertices of 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 $N_G^i[x]$ of x* is defined via $N_G^0[x] := \{x\}$, and $N_G^i[x] := N_G[N_G^{i-1}[x]]$ for $i > 0$. We say that *vertices v and w have distance at least $i + 1$* if $w \notin N_G^i[v]$. 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 $E(f) := \{\{u, v\} \in$

$E \mid f(u) \neq f(v)\}$. 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'. Let f and f' be c-colorings of G. We say that f and f' are *k-neighbors* if $d(f, f') \leq k$. 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'. The problem 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, and $k \in \mathbb{N}$.
Question: Is there a c-coloring f' such that $d(f, f') \leq k$ and $\omega(E(f')) > \omega(E(f))$?

The special case of LS MAX c-CUT where $c = 2$ is denoted as LS MAX CUT.

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 $D(f, \tilde{f}) \subsetneq D(f, f')$.

For details on parameterized complexity we refer to the standard monographs [Cygan *et al.*, 2015; Downey and Fellows, 2013].

3 W[1]-hardness of LS Max Cut

We first show an intractability result. 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 any optimal k-neighbor of the current solution just by considering the graph 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 $|\{w \in N(v) : f(w) = i\}| \geq |\{w \in N(v) : f(w) = f(v)\}| + 2k - 1$.*

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 $f'(v) = i$.*

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 for 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 for the size k of the sought clique [Downey and Fellows, 2013].

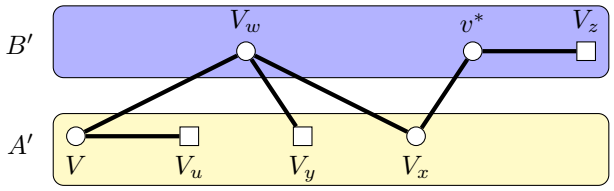

Figure 1: The connections between the different vertex sets in G'. Two vertex sets X and Y are non-adjacent in the figure if $E(X, Y) = \emptyset$. Each vertex v in a vertex set with a rectangular node is k'-blocked from the opposite part of the partition. The vertex set Γ is not shown.

Let $I := (G = (V, E), k)$ be an instance of CLIQUE. In the following, we construct an equivalent instance $I' := (G' = (V', E'), \omega', A', B', k')$ of LS MAX CUT with $\omega' : E' \to \{1\}$. Here, (A', B') is a partition of G' and describes the initial 2-coloring of the instance. 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 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 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 $B' := V_w \cup \{v^*\} \cup V_z$, $A' := V' \setminus B'$, and $k' := 2 \cdot \binom{k}{2} + k + 1$.

To ensure that some vertices are blocked in the final instance, we add the following further vertices to A' and B': 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 to v'. Let Γ be the set of those additional vertices. Figure 1 shows a sketch of the vertex sets and their connections in G'. Note that G' is bipartite and 2-degenerate.

A formal correctness proof is deferred to the full version. Instead, we provide some intuition. Note that each vertex in $V_u \cup V_y$ is contained in A', has at most two neighbors in A', and at least $2k' + 2$ neighbors in B'. Morever, 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. In other words, only vertices in V, V_w, V_x, and the vertex v^* can flip their colors. A clique S in the graph G then corresponds to a flip of v^*, the vertices corresponding to S, and the vertices w_e and x_e for each edge e of the clique. The key mechanism is that each inclusion-minimal improving flip has to contain v^*, so that edges between v^* and V_z become properly colored. To compensate for the edges between V_x and v^* that are not properly colored after flipping v^*, for some edges e of G, the corre-

sponding vertices of V_x and V_w and both endpoints of e have to flip their color. The size of V_z ensures that this has to be done for at least $\binom{k}{2}$ such edges of G. Since we only allow a flip of size k', this then ensures that the edges of G whose corresponding vertices flip their color belong to a clique of size k in G. □

The above reduction can be adapted to prove hardness of LS MAX c-CUT for each fixed $c \geq 2$.

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.*

4 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 know that the colors of the vertices of S change, but we do not yet know 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 $\theta_v^i := \omega(\{\{v, w\} \in E \mid w \in N(v) \setminus S \wedge f(w) \neq i\})$ of edges between v and vertices of $V \setminus S$ that do not receive color i under f. 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 $c' \in [1, c]$. 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 $T[S, c] - \omega_S$: this term 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 $\mathcal{O}(2^{|S'|} \cdot k^2)$ time. Consequently, all entries can be computed in $\mathcal{O}(\sum_{i=0}^{k} \binom{k}{i} \cdot 2^i \cdot c \cdot k^2) = \mathcal{O}(3^k \cdot c \cdot k^2)$ time in total. Hence, the total running time is $\mathcal{O}(3^k \cdot c \cdot k^2 + k \cdot \Delta(G))$. □

For LS MAX CUT, if we interpret a candidate set S as vertices that must all flip their colors, 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.*

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.*

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-mentioned 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.

5 Speedup Strategies

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

5.1 Upper Bounds

To prevent the algorithm 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 algorithm 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 $b(I, S')$ is an* upper bound *if for every c-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 not larger 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 obtained by 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

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

Intuitively, $\alpha_v - \beta_v$ is an upper bound for the improvement obtained 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'. In the definition of β_v, we take into account the edges between v and S' that are falsely counted twice, once when extending $f_{S'}$ with v and a second time in the term $\omega_v^{\overline{f(v)}} - \omega_v^{f(v)}$. Hence, α_v is the improvement over the coloring $f_{S'}$ obtained by changing only the color of v. Let $Y \subseteq V \setminus S'$ be a set containing the $k - |S'|$ vertices from $V \setminus S'$ with largest α_v-values. 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.*

Upper Bound for $c \geq 3$. We next present an upper bound $b_{c \geq 3}$ 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 $b_{c \geq 3}$ consequently contains a summand $b(S')$ that overestimates the edge weights when only the vertices in S' change their colors.

To specify $b(S')$, 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

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

As mentioned above, $b_{c \geq 3}$ the summand $b(S')$ replaces $\omega(E(f_{S'}))$ which was 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

$$b_{c\geq 3}(I, S') := b(S') + \sum_{v \in Y} \alpha_v + 2\binom{k - |S'|}{2} \cdot \omega_{\max}$$

and show that it is in fact an upper bound.

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

5.2 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 for each considered value of k an *auxiliary vertex set* V_k containing all remaining vertices that are potentially part of an improving flip of a k-neighbor of f. For each value of k, the set V_k is initialized once with V, when we search for the first time for an improving k-neighbor.

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 has verified that a vertex v is in no improving k-flip $D(f, f')$, then we may remove v from V_k.

Recall that we set the initial value of k to one and increment k if the current coloring f is k-optimal. If at any time our algorithm replaces the current coloring f by a better coloring f', we set k back to one and continue by searching for an improving k-neighbor of the new coloring f', where k again is incremented if necessary. Now, for each value of k' that was already considered for a previous coloring, we only consider the remaining vertices of $V_{k'}$ together with vertices that have a small distance to the flip between f' and the last previously encountered $(k' - 1)$-optimal coloring. This idea is formalized by 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 within 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 $v \in D(f', \tilde{f})$.*

In our implementation, we use Lemma 3 as follows: 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 idea behind this technique can be explained as follows: if a vertex can be excluded from V_k, then all equivalent vertices can also be excluded, where equivalence is defined as follows.

data	$\|V\|$	$\|E\|$	MOH	LS	ILP	UB
g11	800	1 600	669	—	**671**	**671**
g12	800	1 600	660	661	**663**	**663**
g13	800	1 600	686	687	**688**	**688**
g15	800	4 661	3 984	**3 985**	**3 985**	4 442
g24	2 000	19 990	17 162	**17 163**	—	19 989
g25	2 000	19 990	17 163	**17 164**	—	19 989
g26	2 000	19 990	17 154	**17 155**	—	19 989
g27	2 000	19 990	4 020	**4 021**	—	9 840
g28	2 000	19 990	3 973	**3 975**	—	9 822
g31	2 000	19 990	4 003	**4 005**	—	9 776
g32	2 000	4 000	1 653	1658	**1 666**	1 668
g33	2 000	4 000	1 625	1628	**1 636**	1 640
g34	2 000	4 000	1 607	1609	**1 616**	1 617
g35	2 000	11 778	10 046	**10 048**	—	11 711
g37	2 000	11 785	10 052	**10 053**	**10 053**	11 691
g40	2 000	11 766	2 870	**2 871**	—	5 471
g41	2 000	11 785	2 887	**2 888**	—	5 452
g48	3 000	6 000	**6 000**	—	—	**6 000**
g49	3 000	6 000	**6 000**	—	—	**6 000**
g50	3 000	6 000	**6 000**	—	—	**6 000**
g55	5 000	12 498	12 427	12 429	**12 432**	12 498
g56	5 000	12 498	4 755	**4 757**	—	6 157
g57	5 000	10 000	4 080	4092	**4 103**	4 154
g59	5 000	29 570	7 274	**7 276**	—	14 673
g61	7 000	17 148	6 858	**6 861**	—	8 728
g62	7 000	14 000	5 686	**5 710**	5 706	5 981
g63	7 000	41 459	35 315	**35 318**	—	41 420
g64	7 000	41 459	10 429	**10 437**	—	20 713
g65	8 000	16 000	6 489	6 512	**6 535**	6 711
g66	9 000	18 000	7 414	7 442	**7 443**	7 843
g67	10 000	20 000	8 088	8 116	**8 141**	9 080
g70	10 000	9 999	**9 999**	—	—	**9 999**
g72	10 000	20 000	8 190	8 224	**8 244**	9 166
g77	14 000	28 000	11 579	**11 632**	11 619	13 101
g81	20 000	40 000	16 326	**16 392**	16 374	18 337

Table 1: The graphs from the G-set for which LS or ILP found an improved coloring, or for which we verified that MOH colorings are globally optimal (for $c = 3$). MOH shows the value of the published solutions of [Ma and Hao, 2017], LS and ILP show the best solution of our hill-climbing algorithm and any of the two ILP-runs, respectively. The best coloring is bold. Finally, UB shows the better upper bound computed during the two ILP-runs. For empty entries, no improved coloring was found. For bold UB entries, some found solution matches this upper bound, verifying its optimality.

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 $x \in N(v) \setminus \{w\}$.*

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 $w \in D(f, \tilde{f})$.*

Consequently, when our algorithm removes a vertex v from V_k for some k because no improving k-neighbor f' of f contains v, then it also removes all weighted twins of v with the same color as v from V_k.

6 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 *et al.*, 2015; Wang *et al.*, 2013; Zhu *et al.*, 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 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. MOH is designed to quickly improve substantially on starting solutions but after a while progress stalls (we provide more details on this below). In contrast, our approach makes steady progress but is not as fast as MOH concerning the initial improvements. Hence, we focus on evaluating the performance of LS as a post-processing for MOH by trying to improve their solutions as fast as possible.

We excluded the graph (g23) from our evaluation since there is a large gap between the value of the published colorings and the stated value of the corresponding colorings and we did not want to exploit this gap in our evaluation. The remaining 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 weight 1 or -1). For each of these graphs, we ran experiments for each $c \in \{2, 3, 4\}$ with a time limit of 30 minutes and the published MOH solution as initial solution. In addition to LS, for each instance we ran standard ILP-formulations[1] for MAX c-CUT (again with 30 minute time limit) using the Gurobi solver version 9.5, 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 improved 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, LS was able to improve on 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.

Over all $c \in \{2, 3, 4\}$, on 35 instances, both LS and the ILP found improved colorings compared to the MOH 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 new record coloring, and for 38 instances only LS found a new record coloring. Thus, LS finds improvements also for very hard instances on which MOH provided the best known solutions so far.

	improvable	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

Table 2: The number of instances where LS or ILP found improved solutions. Column 'improvable' shows how many best known MOH colorings [Ma and Hao, 2017] might be suboptimal (as they do not meet the ILP upper bounds). 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.

The MOH solutions were obtained within a time limit of 30, 120, and 240 minutes for small, medium, and large instances, respectively. Each such run was repeated at least 10 times. The average time MOH took to find the best solution was 33% of the respective time limit. Hence, on average, after MOH found their best solution, in the remaining time (at least 20 minutes), MOH did not find any better solution. For all instances where LS was able to improve on 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. Hence, it is indeed helpful to consider larger values of k than the commonly used values of 1 or 2.

7 Conclusion

We summarize our main experimental 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 the ILP approaches. Third, to find improved solutions, it is frequently 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 speed-up strategies. Altogether, these findings indicate that parameterized local search is a promising technique in the design of local search algorithms and that its usefulness should be explored for further hard problems.

[1] Details on the ILP-formulation are provided in a full version of this work.

Acknowledgements

Nils Morawietz was supported by the Deutsche Forschungsgemeinschaft (DFG), project OPERAH, KO 3669/5-1.

Contribution Statement

The research for this article was done while all authors were members of Philipps-Universität Marburg, Department of Mathematics and Computer Science.

References

[Arbour *et al.*, 2021] David Arbour, Drew Dimmery, and Anup B. Rao. Efficient balanced treatment assignments for experimentation. In *Proceedings of the 24th International Conference on Artificial Intelligence and Statistics (AISTATS '21)*, volume 130 of *Proceedings of Machine Learning Research*, pages 3070–3078. PMLR, 2021.

[Benlic and Hao, 2013] Una Benlic and Jin-Kao Hao. Breakout local search for the Max-Cut problem. *Eng. Appl. Artif. Intell.*, 26(3):1162–1173, 2013.

[Berman and Karpinski, 1999] Piotr Berman and Marek Karpinski. On some tighter inapproximability results (extended abstract). In *Proceedings of the 26th International Colloquium on Automata, Languages and Programming (ICALP '99)*, volume 1644 of *Lecture Notes in Computer Science*, pages 200–209. Springer, 1999.

[Bonnet *et al.*, 2019] Édouard Bonnet, Yoichi Iwata, Bart M. P. Jansen, and Lukasz Kowalik. 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*, pages 23:1–23:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

[Chatziafratis *et al.*, 2021] Vaggos Chatziafratis, Mohammad Mahdian, and Sara Ahmadian. Maximizing agreements for ranking, clustering and hierarchical clustering via MAX-CUT. In *Proceedings of the 24th International Conference on Artificial Intelligence and Statistics (AISTATS '21)*, volume 130 of *Proceedings of Machine Learning Research*, pages 1657–1665. PMLR, 2021.

[Cygan *et al.*, 2015] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.

[Dörnfelder *et al.*, 2014] Martin Dörnfelder, Jiong Guo, Christian Komusiewicz, and Mathias Weller. On the parameterized complexity of consensus clustering. *Theoretical Computer Science*, 542:71–82, 2014.

[Downey and Fellows, 2013] Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013.

[Fellows *et al.*, 2012] Michael R. Fellows, Fedor V. Fomin, Daniel Lokshtanov, Frances A. Rosamond, Saket Saurabh, and Yngve Villanger. Local search: Is brute-force avoidable? *Journal of Computer and System Sciences*, 78(3):707–719, 2012.

[Felzenszwalb *et al.*, 2022] Pedro Felzenszwalb, Caroline Klivans, and Alice Paul. Clustering with semidefinite programming and fixed point iteration. *Journal of Machine Learning Research*, 23(190):1–23, 2022.

[Festa *et al.*, 2002] Paola Festa, Panos M. Pardalos, Mauricio G. C. Resende, and Celso C. Ribeiro. Randomized heuristics for the max-cut problem. *Optimization Methods and Software*, 17(6):1033–1058, 2002.

[Frieze and Jerrum, 1997] Alan M. Frieze and Mark Jerrum. Improved approximation algorithms for MAX k-CUT and MAX BISECTION. *Algorithmica*, 18(1):67–81, 1997.

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

[Gaspers *et al.*, 2012] Serge Gaspers, Eun Jung Kim, Sebastian Ordyniak, Saket Saurabh, and Stefan Szeider. Don't be strict in local search! In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence (AAAI '12)*. AAAI Press, 2012.

[Gaspers *et al.*, 2019] Serge Gaspers, Joachim Gudmundsson, Mitchell Jones, Julián Mestre, and Stefan Rümmele. Turbocharging treewidth heuristics. *Algorithmica*, 81(2):439–475, 2019.

[Gaur *et al.*, 2009] Daya Ram Gaur, Ramesh Krishnamurti, and Rajeev Kohli. Conflict resolution in the scheduling of television commercials. *Operations Research*, 57(5):1098–1105, 2009.

[Grüttemeier *et al.*, 2021] Niels Grüttemeier, Christian Komusiewicz, and Nils Morawietz. Efficient Bayesian network structure learning via parameterized local search on topological orderings. In *Proceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI '21)*, pages 12328–12335. AAAI Press, 2021. Full version available at https://doi.org/10.48550/arXiv.2204.02902.

[Guo *et al.*, 2013] Jiong Guo, Sepp Hartung, Rolf Niedermeier, and Ondrej Suchý. The parameterized complexity of local search for TSP, more refined. *Algorithmica*, 67(1):89–110, 2013.

[Guo *et al.*, 2014] Jiong Guo, Danny Hermelin, and Christian Komusiewicz. Local search for string problems: Brute-force is essentially optimal. *Theoretical Computer Science*, 525:30–41, 2014.

[Hartung and Niedermeier, 2013] Sepp Hartung and Rolf Niedermeier. Incremental list coloring of graphs, parameterized by conservation. *Theoretical Computer Science*, 494:86–98, 2013.

[Huang *et al.*, 2017] Weiran Huang, Liang Li, and Wei Chen. Partitioned sampling of public opinions based on their social dynamics. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI '17)*, pages 24–30. AAAI Press, 2017.

[Jensen and Toft, 2011] Tommy R Jensen and Bjarne Toft. *Graph coloring problems*, volume 39. John Wiley & Sons, 2011.

[Kann *et al.*, 1997] Viggo Kann, Sanjeev Khanna, Jens Lagergren, and Alessandro Panconesi. On the hardness of approximating Max k-Cut and its dual. *Chicago Journal of Theoretical Computer Science*, 1997(2), 1997.

[Karp, 1972] Richard M. Karp. Reducibility among combinatorial problems. In *Proceedings of a Symposium on the Complexity of Computer Computations*, The IBM Research Symposia Series, pages 85–103. Plenum Press, New York, 1972.

[Katzmann and Komusiewicz, 2017] Maximilian Katzmann and Christian Komusiewicz. 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)*, pages 846–852. AAAI Press, 2017.

[Kleinberg and Tardos, 2006] Jon M. Kleinberg and Éva Tardos. *Algorithm design*. Addison-Wesley, 2006.

[Komusiewicz and Morawietz, 2022] Christian Komusiewicz and Nils Morawietz. Parameterized Local Search for Vertex Cover: When Only the Search Radius Is Crucial. In *17th International Symposium on Parameterized and Exact Computation (IPEC 2022)*, volume 249 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 20:1–20:18, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[Komusiewicz and Sommer, 2021] Christian Komusiewicz and Frank Sommer. Enumerating connected induced subgraphs: Improved delay and experimental comparison. *Discrete Applied Mathematics*, 303:262–282, 2021.

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

[Komusiewicz *et al.*, 2023] Christian Komusiewicz, Simone Linz, Nils Morawietz, and Jannik Schestag. On the complexity of parameterized local search for the maximum parsimony problem. In *34th Annual Symposium on Combinatorial Pattern Matching, CPM 2023*, LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. Accepted for publication.

[Leiserson *et al.*, 2011] Mark D. M. Leiserson, Diana Tatar, Lenore J. Cowen, and Benjamin J. Hescott. Inferring mechanisms of compensation from E-MAP and SGA data using local search algorithms for max cut. *Journal of Computational Biology*, 18(11):1399–1409, 2011.

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

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

[Papadimitriou and Yannakakis, 1991] Christos H. Papadimitriou and Mihalis Yannakakis. Optimization, approximation, and complexity classes. *Journal of Computer and System Sciences*, 43(3):425–440, 1991.

[Schäffer and Yannakakis, 1991] Alejandro A. Schäffer and Mihalis Yannakakis. Simple local search problems that are hard to solve. *SIAM Journal on Computing*, 20(1):56–87, 1991.

[Shylo *et al.*, 2015] VP Shylo, F Glover, and IV Sergienko. Teams of global equilibrium search algorithms for solving the weighted maximum cut problem in parallel. *Cybernetics and Systems Analysis*, 51(1):16–24, 2015.

[Subramanian *et al.*, 2008] Anand Prabhu Subramanian, Himanshu Gupta, Samir R. Das, and Jing Cao. Minimum interference channel assignment in multiradio wireless mesh networks. *IEEE Transactions on Mobile Computing*, 7(12):1459–1473, 2008.

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

[Vredeveld and Lenstra, 2003] Tjark Vredeveld and Jan Karel Lenstra. On local search for the generalized graph coloring problem. *Operations Research Letters*, 31(1):28–34, 2003.

[Šíma *et al.*, 1999] Jiří Šíma, Pekka Orponen, and Teemu Antti-Poika. Some afterthoughts on Hopfield networks. In *Proceedings of the 26th Conference Theory and Practice of Informatics on Current Trends in Theory and Practice of Informatics (SOFSEM '99)*, volume 1725 of *Lecture Notes in Computer Science*, pages 459–469. Springer, 1999.

[Wang *et al.*, 2013] Yang Wang, Zhipeng Lü, Fred Glover, and Jin-Kao Hao. Probabilistic GRASP-Tabu search algorithms for the UBQP problem. *Computers & Operations Research*, 40(12):3100–3107, 2013.

[Wang, 2006] Jiahai Wang. An improved discrete Hopfield neural network for Max-Cut problems. *Neurocomputing*, 69(13-15):1665–1669, 2006.

[Zhu *et al.*, 2013] Wenxing Zhu, Geng Lin, and M. Montaz Ali. Max-k-cut by the discrete dynamic convexized method. *INFORMS Journal on Computing*, 25(1):27–40, 2013.