Local Search: Is brute-force avoidable?[*]

Michael R. Fellows[†] Fedor V. Fomin[‡] Daniel Lokshtanov[‡]
Frances A. Rosamond[†] Saket Saurabh[‡] Yngve Villanger[‡]

August 7, 2009

Abstract

Many local search algorithms are based on searching in the k-exchange neighborhood. This is the set of solutions that can be obtained from the current solution by exchanging at most k elements. As a rule of thumb, the larger k is, the better are the chances of finding an improved solution. However, for inputs of size n, a naïve brute-force search of the k-exchange neighborhood requires $n^{\mathcal{O}(k)}$ time, which is not practical even for very small values of k. We show that for several classes of sparse graphs, like planar graphs, graphs of bounded vertex degree and graphs excluding some fixed graph as a minor, an improved solution in the k-exchange neighborhood for many problems can be found much more efficiently. Our algorithms run in time $\mathcal{O}(\tau(k) \cdot n^c)$, where τ is a function depending on k only and c is a constant independent of k. We demonstrate the applicability of this approach on different problems like r-Center, Vertex Cover, Odd Cycle Transversal, Max-Cut, and Min-Bisection. In particular, on planar graphs, all our algorithms searching for a k-local improvement run in time $\mathcal{O}(2^{\mathcal{O}(k)} \cdot n^2)$, which is polynomial for $k = \mathcal{O}(\log n)$. We also complement the algorithms with complexity results indicating that—brute force search is unavoidable—in more general classes of sparse graphs.

1 Introduction

Local search is one of the most common approaches applied in practice for solving hard optimization problems. Very often it is used as a subroutine in iterated algorithms and evolutionary algorithms. The history of employing local search in combinatorial optimization and operations research dates

[*]The results of this paper were presented in [17].

[†]University of Newcastle, Newcastle, Australia. {michael.fellows—frances.rosamond}@newcastle.edu.au

[‡]Department of Informatics, University of Bergen, Norway. {fomin—daniello—saket.saurabh—yngvev}@ii.uib.no. Supported by the Norwegian Research Council.

back to the 1950s with the first edge-exchange algorithms for the traveling salesperson [5, 8]. The idea of local search is to improve a solution by searching for a better solution in a neighborhood which is defined in a problem specific way. For example, for the classical Traveling Salesperson problem, the neighborhood of a tour can be defined as the set of all tours that differ from it in at most k edges, or so-called the k-exchange neighborhood [27, 32]. Another classical example is the Min-Bisection problem [22], where for a given graph with $2n$ vertices and weights on the edges, the task is to partition the vertices into two sets of n vertices such that the sum of the weights of the edges between the sets is minimized. A natural k-exchange neighborhood for this problem would be the set of partitions that can be obtained from each other by swapping at most k pairs of vertices.

Most of the literature on local search is primarily devoted to experimental studies of different heuristics. The theoretical study of local search has been developing mainly in three directions. The first direction is the study of performance guarantees of local search, i.e. the quality of the solution [3, 4, 20, 23, 33]. The second direction of the theoretical work is on the asymptotic convergence of local search in probabilistic settings, such as simulated annealing [1, 2, 15]. The third direction, which is the most relevant to our work, concerns the time required to reach a local optimum. An illustrative example here is the simplex method, which can be seen as a local search algorithm with feasible solutions corresponding to "vertices" of a polytope. The neighbors of a solution are the solutions that can be reached from it by a single "pivot". There can be different rules of choosing a pivot when several neighbors improve a solution. However, for each of the known rules there are examples requiring an exponential number of iterations for reaching the local optimum. Motivated by the fact that many local search algorithms are based on neighborhood structures for which locally optimal solutions are not known to be computable in polynomial time, Johnson, Papadimitriou and Yannakakis [22] defined a complexity class PLS which can be seen as an analogue of the class NP for local search problems. Many natural local search problems appear to be PLS-complete. We refer to books [2, 30] for more information on different aspects of local search.

In this paper we take a different twist in the study of local search and endeavor to answer the following natural question. Is there a faster way of searching the k-exchange neighborhood than brute-force? This question is important because the typical running time of a brute-force algorithm is $n^{\mathcal{O}(k)}$, where n is the input length, which becomes a real obstacle in using k-exchange neighborhoods in practice even for sufficiently small values of k. It has been taken by default for many years that finding an improved solution in the k-exchange neighborhood cannot be done significantly faster than the brute-force search. But is there mathematical evidence for this common belief? Or maybe for some problems it is possible to search k-exchange neighborhoods in time $\mathcal{O}(\tau(k)n^c)$, where c is a small constant, which can make local search much more powerful?

An appropriate tool to answer all these questions is parameterized complexity. In the parameterized framework, for decision problems with input size n, and a parameter k, the goal is to design an algorithm with runtime $\tau(k) \cdot n^{\mathcal{O}(1)}$, where τ is a function of k alone. Problems having such an algorithm are said to be fixed parameter tractable (FPT). There is also a theory of hardness that allows to identify parameterized problems that are not amenable to such algorithms. The hardness hierarchy is represented by $W[i]$ for $i \geq 1$. For an introduction and more recent developments see books [13, 18, 31].

Relevant results. The parameterized complexity of local search remains unexplored with exceptions that are few and far between. As it was mentioned by Marx in a recent survey on local search problems [28]:

> *"So far, there are only a handful of parameterized complexity results in the literature, but they show that this is a fruitful research direction. The fixed-parameter tractability results are somewhat unexpected and this suggests that there are many other such results waiting to be discovered."*

The first breakthrough in the area is due to Marx [29] who proved that finding a local improvement in k-exchange neighborhood for TSP is $W[1]$-hard. It also appears, that finding k-local optimum is not $W[1]$-hard for every NP-hard problem. An example of this can be found in the work of Khuller, Bhatia, and Pless [25] who investigated the NP-hard problem of finding a feedback edge set that is incident to the minimum number of vertices. One of the results obtained in [25] is that checking whether it is possible to improve a solution by replacing at most k edges can be done in time $\mathcal{O}(n^2 + n\tau(k))$, i.e., it is FPT parameterized by k. Very recently, Krokhin and Marx [26] investigated the local search problem for finding a minimum weight assignment for a Boolean constraint satisfaction instance. Szeider [37] studied the parameterized complexity of k-exchange problems for SAT.

Prior to our paper no systematic research was done on the complexity of k-optimum on graph problems.

Organization of the paper In this work, we initiate the systematic study of parameterized complexity of local search for different graph problems. In Section 2 we provide all technical definitions. In Section 3 we formally define the parameterized version of the local search problem and set up a framework of study. In Section 4 we give a generic hardness proof, which show that local search versions of most graph problems are $W[1]$ or $W[2]$-hard on general graphs. Thus in general situation brute-force is unavoidable and this is why we turn our attention to classes of sparse graphs. In Section 5 we investigate local search for problems on graphs of bounded local treewidth, a wide class of graphs containing planar graphs, graphs embeddable on a surface of bounded genus and graphs of bounded vertex degree. We show that many local search problems become FPT when the input graph is of bounded local treewidth. In particular, we show that finding k-local improvement on graphs of bounded local treewidth is FPT for many natural problems including Vertex Cover, Odd Cycle Transversal, Dominating Set, and r-Center. With a modification of this technique (Section 6) we also show that finding k-local improvement for Max-Cut, and Min-Bisection is FPT for apex-minor-free graphs. All these results are based on the idea of reducing the search in k-exchange neighborhood to searching for an improvement in a ball of small diameter around some vertex in the metric of the input graph. For example, on planar graphs, this approach leads to algorithms with running time $\mathcal{O}(2^{\mathcal{O}(k)} \cdot n^2)$ for many of the problems mentioned above. In Section 7 extend these results to more general classes of graphs, namely, graphs excluding a fixed graph as a minor. Here we strongly use Structure Theorem of Robertson and Seymour from Graph Minors. We end up with the proof that existence of FPT algorithms is highly unlikely for larger classes of sparse graphs. We prove that most of the local search problems are $W[1]$-hard on 3-degenerated graphs. These results can be found in Section 8. We conclude with open problems and directions for further research.

2 Preliminaries

Let $G = (V, E)$ be an undirected graph where V is the set of vertices and E is the set of edges. We denote the number of vertices by n and the number of edges by m. We also use $|G|$ to denote the number of vertices in G. For a subset $V' \subseteq V$, by $G[V']$ we mean the subgraph of G induced by V'. By $N(u)$ we denote (open) neighborhood of u that is the set of all vertices adjacent to u and by $N[u] = N(u) \cup \{u\}$. Similarly, for a subset $D \subseteq V$, we define $N[D] = \cup_{v \in D} N[v]$. The *distance* $d_G(u, v)$ between two vertices u and v of G is the length of the shortest path in G from u to v. The *diameter* of a graph G, denoted by $diam(G)$, is defined to be the maximum length of a shortest path between any pair of vertices of $V(G)$. By an abuse of notation, we define diameter of a graph as the maximum of the diameters of its connected components. For $r \geq 0$, the *r-neighborhood* of a vertex $v \in V$ is defined as $N_G^r[v] = \{u \mid d_G(v, u) \leq r\}$. We also let $B(r, v) = N_G^r[v]$ and call it a ball of radius r around v. Similarly $B(r, A) = \cup_{v \in A} N_G^r[v]$ for $A \subseteq V$. Given a weight function $w : V \to \mathbb{R}^+ \cup \{0\}$ and $A \subseteq V$, $w(A) = \sum_{u \in A} w(u)$. Given two sets S_1 and S_2, the *symmetric difference* between S_1 and S_2 is defined as $S_1 \triangle S_2 = (S_1 \setminus S_2) \cup (S_2 \setminus S_1)$.

2.1 Classes of sparse graphs

The *genus* of a graph is the minimal integer g such that the graph can be embedded in a surface of genus g. In particular, a planar graph has genus 0. Planar graphs and graphs of bounded genus are the classes of graphs that can be characterized by a set of graphs that they do not contain as a minor.

For an edge $e = (u, v)$ of a graph G, the graph G/e is obtained by contracting (u, v); that is, G/e is obtained from G by identifying the vertices u and v and removing all the loops and duplicate edges. A *minor* of a graph G is a graph H that can be obtained from a subgraph of G by contracting edges. A graph class $\mathcal{C}$ is *minor closed* if any minor of any graph in $\mathcal{C}$ is also an element of $\mathcal{C}$. A minor closed graph class $\mathcal{C}$ is *H-minor-free* or simply *H-free* if $H \notin \mathcal{C}$. A graph H is called an apex graph if for some vertex v of H the removal of v turns H into a planar graph. A minor closed graph class $\mathcal{C}$ is *apex-minor-free* if there is an apex graph H such that $H \notin \mathcal{C}$.

Another class of sparse graphs we are interested in are graphs of bounded local treewdith. A *tree decomposition* of a (undirected) graph $G = (V, E)$ is a pair (X, T) where $T = (V_T, E_T)$ is a tree whose vertices we will call *nodes* and $X = (\{X_i \mid i \in V_T\})$ is a collection of subsets of V such that

1. $\bigcup_{i \in V_T} X_i = V$,
2. for each edge $(v, w) \in E$, there is an $i \in V_T$ such that $v, w \in X_i$, and
3. for each $v \in V$ the set of nodes $\{i \mid v \in X_i\}$ forms a subtree of T.

The *width* of a tree decomposition $(\{X_i \mid i \in V_T\}, T)$ is $\max_{i \in V_T}\{|X_i| - 1\}$. The *treewidth* of a graph G is the minimum width over all tree decompositions of G. We use notation $\mathbf{tw}(G)$ to denote the treewidth of a graph G. The definition of treewidth can be generalized to take into account the local properties of G and is called *local treewidth* [14].

Definition 1 (Local tree-width). *The* local tree-width *of a graph G is a function $\mathbf{ltw}^G : \mathbb{N} \to \mathbb{N}$ which associates to every integer $r \in \mathbb{N}$ the maximum tree-width of an r-neighborhood of vertices of G, i.e. $\mathbf{ltw}^G(r) = \max_{v \in V}\{tw(G[N_G^r(v)])\}$.*

For a function $h : \mathbb{N} \to \mathbb{N}$ we define the graph class $\mathscr{G}_h$ such that for each graph $G \in \mathscr{G}$, and for each integer $r \in \mathbb{N}$, we have $\mathbf{ltw}^G(r) \leq h(r)$. We say that a class of graphs $\mathscr{G}$ is of *bounded local treewidth* if $\mathscr{G} \subseteq \mathscr{G}_h$ for some $h : \mathbb{N} \to \mathbb{N}$. A graph class $\mathscr{G}_h$ has *bounded local treewidth* if there exists a function $h : \mathbb{N} \to \mathbb{N}$ such that for each graph $G \in \mathscr{G}$, and for each integer $r \in \mathbb{N}$, we have $\mathbf{ltw}^G(r) \leq h(r)$. Well known graph classes of bounded local treewidth are planar graphs, graphs of bounded genus, and graphs of bounded maximum vertex degree (see [14] and [19] for more details and algorithmic properties of these graphs). By the result of Robertson and Seymour [34] (see also [6]), $h(r)$ can be chosen as $3r$ for planar graphs. Similarly, Eppstein have shown in [14] that $h(r)$ can be chosen as $c_g g(\Sigma)r$ for graphs embeddable in a surface Σ, where $g(\Sigma)$ is the genus of the surface Σ and c_g is a constant depending only on $g(\Sigma)$. Demaine and Hajiaghayi [11] extended this result and showed that for minor closed families of graphs which do not contain some fixed apex graph as a minor, $h(r) = \mathcal{O}(r)$.

A graph G is d-degenerated if every induced subgraph of G has a vertex of degree at most d. It is well known, for example that planar graphs are 5-degenerated and that H-minor-free graphs are h-degenerated for some constant h. See Fig. 1 for the relationship between all mentioned classes of graphs.

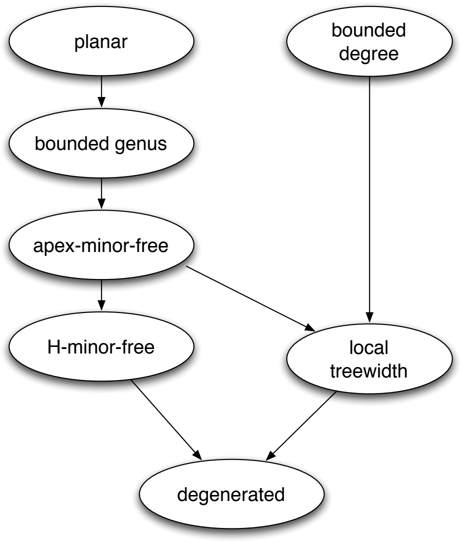

Figure 1: Classes of sparse graphs. Arrows points from narrow to broader classes.

2.2 Parameterized complexity

TO DISCUSS. SHALL WE GIVE HERE THE DEFINITION OF FPT AND W[1] AND W[2]? OR AT LEAST GIVE SOME REFERENCES AND SAY WHAT IS FPT REDUCTION?

3 Framework of Study

We begin our study of local search variants of optimization problems by introducing notation and concepts that will be used throughout the paper.

A combinatorial optimization problem is specified by a set of problem instances and it is either a *minimization* or *maximization* problem.

Definition 2. *An instance of a combinatorial optimization problem P is a pair $(\mathscr{S}, c)$, where the solution set $\mathscr{S}$ is the set of feasible solution and the cost function c is a mapping $c : \mathscr{S} \to \mathbb{R}$. The problem is to find a globally optimum solution $s^* \in \mathscr{S}$ such that $c(s^*) \leq c(s)$ $(c(s^*) \geq c(s))$ for all $s \in \mathscr{S}$ if P is minimization (maximization) problem.*

For a given feasible point $s \in \mathscr{S}$ in a particular problem, it is useful to define a set of points that are “close” to s in some sense.

Definition 3. *Let $(\mathscr{S}, f)$ be an instance of a combinatorial optimization problem P. A neighborhood function is a mapping $\mathscr{N} : \mathscr{S} \to 2^{\mathscr{S}}$, which defines for each solution $s \in \mathscr{S}$, a set $\mathscr{N}(s) \subseteq \mathscr{S}$ of solutions. The set $\mathscr{N}(s)$ is the neighborhood of solution s and each $s' \in \mathscr{N}(s)$ is a neighbor of s.*

In general a *local search* algorithm starts off with an initial solution and then tries to find better solutions by searching neighborhoods. The most commonly used version of local search algorithm iteratively improves the solution by replacing the current solution with lower/higher cost. If the algorithm can not improve the current solution then the current solution is *locally optimum.*

Definition 4. *Let $(\mathscr{S}, c)$ be an instance of a combinatorial optimization problem P and let $\mathscr{N}$ be a neighborhood function. A solution $\hat{s}$ is locally optimum (minimal/maximal) with respect to $\mathscr{N}$ if $c(\hat{s}) \leq c(s)$ $(c(\hat{s}) \geq c(s))$ for all $s \in \mathscr{N}(\hat{s})$ if P is a minimization (maximization) problem. We denote the set of locally optimum solution by $\mathscr{S}$.*

There are various neighborhood structure known in the literature for different problems. For many optimization problems defined on graphs, the solution is a subset of vertices or edges of the graph. This is the case for the problems Vertex Cover, Independent Set, and Dominating Set, Edge Dominating Set, and Minimum Maximal Matching, to name a few. A problem P is a *vertex subset* problem (or an *edge subset* problem) if a feasible solution to P is $V' \subseteq V$ $(E' \subseteq E)$. We use $\mathscr{S}$ to denote the set of feasible solutions to a problem P.

For vertex subset (or edge subset) problems, a natural neighborhood function is obtained by exchanging k elements of the current solution. The neighborhood function in which we are interested is called *k-exchange neighborhoods* (k-**ExN**). We elaborate this further for minimization vertex-subset problems. Let $w : V \to \mathbb{R}^+$ be a weight function. Then the cost function $c : \mathscr{S} \to \mathbb{R}^+$ is defined as $\sum_{v \in s} w(v)$ for all $s \in \mathscr{S}$. For a pair of elements $s_1, s_2 \in \mathscr{S}$, let $H(s_1, s_2)$ denote the Hamming distance that is the size of the set $|s_1 \setminus s_2|$. We say that s' is neighbor of s with respect to k-**ExN** if

$H(s, s') \leq k$. Let $\mathcal{N}_k^{en}(s)$ denote the set of neighbors of s with respect to k-**ExN**. Then the generic problem of local search for a vertex subset graph optimization problem P with respect to k-**ExN** is defined as follows.

k-LOCAL SEARCH P (k-LS-P)

Input: A graph $G = (V, E)$, a weight function $w : V \to \mathbb{R}^+$, a solution S and an integer $k \geq 0$.
Parameter: A positive integer k.
Question: Does there exist a solution $T \in \mathcal{N}_k^{en}(S)$ such that $c(T) < c(S)$ if P is minimization problem (and $c(S) > c(T)$ if P is maximization problem)?

For example, if P is VERTEX COVER or ODD CYCLE TRANSVERSAL, we use k-LS-VERTEX COVER and k-LS-ODD CYCLE TRANSVERSAL, respectively.

In general, an input to parameterized local search problem for a graph optimization problem P is a 4-tuple $(G = (V, E), w, S, k)$, where S is an initial solution, $w : V \to \mathbb{R}^+$, and $k \in \mathbb{N}$ is a positive integer. If w is a unit weight function then we ignore w and then input becomes 3-tuple $(G = (V, E), S, k)$.

4 Hardness Results for Local Search in General Graphs

In this section we prove a general result showing that local search variants of many vertex subset problems in general graphs are hard for different levels of the parameterized hierarchy. To do this we need to introduce some notation. A graph property Π is a collection of graphs. The property Π is said to be *hereditary* if $G \in \Pi$ implies that every induced subgraph of G is also in Π. For a property Π, we define the Π-SUBSET problem and its parametric dual, the Π-DELETION problem as follows. Π-SUBSET: Given a graph $G = (V, E)$ and a positive integer t, determine whether there exists a set of t vertices $V' \subseteq V$ such that $G[V']$ is in Π. Π-DELETION: Given a graph $G = (V, E)$ and a positive integer t, determine whether there exists a set of t vertices $V' \subseteq V$ such that $G[V \setminus V']$ is in Π. Khot and Raman [24] studied the parameterized complexity of Π-SUBSET problems and showed the following result.

Theorem 1 (Khot and Raman [24]). *Let Π be a hereditary property that includes all independent sets but not all cliques (or vice versa). Then Π-SUBSET is $W[1]$ hard, parameterized by t, the size of the subsets.*

We study the local search variants of Π-SUBSET and Π-DELETION, k-LS-Π-SUBSET and k-LS-Π-DELETION. An instance to k-LS-Π-SUBSET is a tuple $(G = (V, E), S, k)$ with S a subset of V satisfying that $G[S] \in \Pi$. The question is whether there is another set $S' \subseteq V$ such that $|S' \setminus S| \leq k$ and $|S'| > |S|$. The k-LS-Π-DELETION problem is derived from the Π-DELETION problem in a similar fashion.

Theorem 2. *Let* Π *be a hereditary property. For a graph class* $\mathcal{G}$*, the* k*-LS-*Π-SUBSET *problem is FPT on* $\mathcal{G}$ *if and only if the* k*-LS-*Π-DELETION *problem is FPT on* $\mathcal{G}$*. Furthermore, if the* Π-SUBSET *problem or the* Π-DELETION *problem is* $W[1]$ *hard then so is both the the* k*-LS-*Π-SUBSET *problem and the* k*-LS-*Π-DELETION *problem.*

Proof. For a subset S of V, (G, S, k) is an YES instance to k-LS-Π-SUBSET if and only if $(G, V \setminus S, k)$ is an YES instance to the k-LS-Π-DELETION problem. This shows the equivalence of the two local search problems on the graph class $\mathcal{G}$.

We now show that the existence of an FPT algorithm for k-LS-Π-SUBSET implies that Π-SUBSET is FPT. Let $(G = (V, E), k)$ be an instance of Π-SUBSET and $\mathcal{A}$ be an FPT algorithm for k-LS-Π-SUBSET. Since $\emptyset \in \Pi$, we have that $\emptyset$ is a feasible solution for Π-SUBSET. Also, any two sets of size at most k have hamming distance at most $2k$ between each other. Running $\mathcal{A}$ at most k times (with parameter $2k$) we can either find a subset $U \subset V$ of size at least k such that $G[U]$ is in Π, or conclude that no such subset exists. Hence if the running time of $\mathcal{A}$ is $f(k) \cdot |V|^{\mathcal{O}(1)}$, for some function f, then Π-SUBSET is solvable in time $kf(2k) \cdot |V|^{\mathcal{O}(1)}$, making the Π-SUBSET problem FPT and thus proving k-LS-Π-SUBSET $W[1]$-hard. A similar reduction can be obtained from the Π-DELETION problem to the k-LS-Π-DELETION problem. This concludes the proof. □

Theorems 1 and 2 together yield hardness results for the local search variants of many Π-SUBSET problems. Examples include local search variants of INDEPENDENT SET (whose dual is VERTEX COVER), INDUCED FOREST (its dual is FEEDBACK VERTEX SET), and INDUCED BIPARTITE SUBGRAPH (with dual ODD CYCLE TRANSVERSAL). Similarly to Theorem 2, one can show the hardness of local search variants of $W[2]$-hard problems such as DOMINATING SET or INDEPENDENT DOMINATING SET.

5 FPT Algorithms for k-LS in Graphs of Bounded Local Treewidth

In this section we show that many local search problems become fixed parameter tractable in graphs of bounded local treewidth.

5.1 Domination Problems

In this section we give an algorithm for the local search variant of a generalization of Dominating Set, called the r-Center problem. A subset $S \subseteq V$ is called an r-center if $V \subseteq B(r, S)$.

In k-LS-r-Center, we are given an undirected graph $G = (V, E)$, with weight function $w : V \to \mathbb{R}^+$, an r-center $S \subseteq V$ and integers k, r. The problems asks whether there exists a $S' \in N_k^{en}(S)$ such that $c(S') < c(S)$. Here k and r are the parameters.

When all the vertices have weight 1 and $r = 1$ this is the k-LS-Dominating Set problem. Our result is based on a combinatorial characterization of changed solutions. We prove that if there is an improved solution that is close to the current solution in the solution space, then there is another improved solution close to the current one in the solution space with the additional property that all changes are concentrated locally in the input graph.

Lemma 1. *Let S_1 and S_2 be r-centers of a weighted graph $G = (V, E)$ with weight function $w : V \to \mathbb{R}^+$, such that $|S_1 \triangle S_2| \leq p$ and $c(S_1) > c(S_2)$. Then there are sets $F_1, F_2 \subseteq V$ such that*

(*a*) *the set $S = (S_1 \setminus F_1) \cup F_2$ is an r-center of G and $c(S) < c(S_1)$;*

(*b*) *$|F_1 \cup F_2| \leq p$; and*

(*c*) *there is a vertex $z \in V$ such that $F_1 \cup F_2 \subseteq B(z, 2pr)$.*

Proof. Let $X = S_1 \setminus S_2$ and $Y = S_2 \setminus S_1$. Furthermore, $X \cap Y = \emptyset$ and $S_1 \setminus X = S_2 \setminus Y = S_1 \cap S_2$. This implies that $c(S_1) = w(S_1 \setminus X) + w(X) > c(S_2) = w(S_2 \setminus Y) + w(Y)$, and hence $w(X) > w(Y)$. Consider the auxiliary graph $G^* = (X \cup Y = S_1 \triangle S_2, E')$, where two vertices $u, v \in (X \cup Y)$ are adjacent if the shortest path distance in G between u and v is at most $2r$. Let $C_1, \ldots, C_p$ be the connected components of G^*. For every connected component C_i, we assign a tuple $\mu(C_i) = (x, y)$ where $x = w(X \cap C_i)$ and $y = w(Y \cap C_i)$. Since $w(X) > w(Y)$, we have that there is a connected component C_j such that $x > y$ in $\mu(C_j)$. We put $F_1 = X \cap C_j$ and $F_2 = Y \cap C_j$ and claim that these sets satisfy the conditions of the lemma.

Indeed, $S = (S_1 \setminus F_1) \cup F_2 = S_1 \setminus (X \cap C_j) \cup (Y \cap C_j)$. Since $x > y$, we have that $c(S) = w(S_1) - w(X \cap C_j) + w(Y \cap C_j) < c(S_1)$. We show first that S is a r-center of G. We know that S_1 is a r-center of G and hence only vertices which could be at distance more than r from the vertices of S are those which were distance at most r from the vertices in $(X \cap C_j)$. Let u be a vertex which is at distance more than r from any vertex in S. Then there exists a vertex, say $u_1 \in (X \cap C_j)$, such that $d(u, u_1) \leq r$. Furthermore, S_2 is an r-center not containing any of the vertices of $(X \cap C_j)$. Hence, there exists $v \in Y$ such that $d(u, v) \leq r$. But this implies that $d(u_1, v) \leq 2r$

and hence $v \in (Y \cap C_j)$. This proves that S is an r-center of G. We can select any vertex in $X \cap C_j$ for z. The size of C_j is at most p and any pair of adjacent (in this component) vertices are at distance at most $2r$ in G. Hence, the ball around z of radius $2pr$ contains all the vertices of $F_1 \cup F_2$; that is, $(F_1 \cup F_2) \subseteq B(z, 2pr)$. □

We define a generalized form of k-LS-r-Center which we call k-LS-Generalized r-Center where apart from S, we are also given a subset $T \subseteq S$, and we need to find a solution $S' \in \mathcal{N}_k^{en}(S)$ such that $c(S') < c(S)$ and $T \subseteq S'$. So an instance of k-LS-Generalized r-Center looks like (G, w, S, T, k).

Lemma 2. *Let $\mathcal{G}$ be the class of graphs which is closed under taking induced subgraphs and for which we can solve* k-LS-Generalized r-Center *in time $f(\ell) \cdot |G|^{\mathcal{O}(1)}$ whenever $G \in \mathcal{G}$ and of diameter at most ℓ. Then* k-LS-r-Center *is FPT for $\mathcal{G}$.*

Proof. Let (G, w, S, k) be an instance of k-LS-r-Center, where $G \in \mathcal{G}$. By Lemma 1, we know that if there is a local improvement in the k-exchange neighborhood of S, then there exists a solution $S' \in \mathcal{N}_k^{en}(S)$ with $c(S') < c(S)$ and $z \in S \setminus S'$ such that $S \triangle S' \subseteq B(z, 4rk)$.

For every vertex $v \in S$ we try v on the role of the desired vertex z and do as follows. We do BFS starting at v. Let the layers created by doing BFS on v be $L_0^v, L_1^v, \ldots, L_t^v$. We have two cases: either (a) $t \leq 4rk + r$ or (b) $t > 4rk + r$. In case (a) we set $T_v = \emptyset$ and get (G, w, S, T_v, k) as an instance for k-LS-Generalized r-Center. In the other case, we first take $4rk + r$ layers; that is,

$$B(v, 4rk + r) = \bigcup_{j=0}^{4rk+r} L_j^v.$$

We know that all the changed vertices, those that go out and those that will come in (that is, vertices in the set $S \triangle S'$) are in $B(v, 4rk)$. Let

$$T_v := S \cap \bigcup_{j=4rk+1}^{4rk+r} L_j^v.$$

Furthermore, for $v \in S$, let $S_v = S \cap B(v, 4rk + r)$. For every $v \in S$, we get the following instance $(B(v, 4rk+r), w, S_v, T_v, k)$ for k-LS-Generalized r-Center. An instance (G, w, S, k) is an Yes instance for k-LS-r-Center if and only if there exists $v \in S$ for which $(B(v, 4rk+r), w, S_v, T_v, k)$ is an Yes instance for k-LS-Generalized r-Center. Since k-LS-Generalized r-Center is solvable in time $f(4rk+r) \cdot |B(v, 4rk+r)|^{\mathcal{O}(1)}$ for $G[B(v, 4rk+r)]$ for $v \in S$, we have that k-LS-r-Center for G is solvable in time $f(4rk + r) \cdot |G|^{\mathcal{O}(1)}$. This concludes the proof of the lemma. □

Theorem 4.1 in [9] give an algorithm that takes a graph G with m edges, a branch decomposition of G of width ℓ, and integers k, r as input, and check in time $\mathcal{O}((2r+1)^{\frac{3}{2}\ell} \cdot m)$ for the existence of a (k,r)-center in G. This result is obtained by dynamic programming over the branch decomposition, using $2r+1$ different colors. In the proof of the following lemma we define an extension to the proof of Theorem 4.1 in [9].

MAYBE WE SHOULD GIVE HERE JUST FEW LINES OF WHAT WE DO BEFORE LEMMA. OR TO GIVE THE FULL PROOF. YV: suggestion added.

Lemma 3. *Let $\mathcal{G}$ be the class of graphs of treewidth at most t, then k-*LS-Generalized *r-*Center *can be solved in time $\mathcal{O}((k(2r+1))^{\frac{3}{2}t+1} \cdot |G|^{\mathcal{O}(1)})$ for graphs $G \in \mathcal{G}$, where a tree decomposition of width t is provided.*

Proof. By [35] we know that any graph of treewidth t has branchwidth at most $t+1$. We can now use the treedecomposition of width t, to obtain a branch decomposition of width $t+1$. (YV: IS THERE SOMEWONE WE CAN CITE ON THIS?) Let (G, w, S, T, k) be an instance of k-LS-Generalized r-Center. By the algorithm of [9] the problem can be solved to optimality in $\mathcal{O}((2r+1)^{\frac{3}{2}t+1} \cdot |G|^{\mathcal{O}(1)})$ time by dynamic programming. In our case we would like to find the optimal solution having at most k changes compared to S. We use the same dynamic programming and the same $2r+1$ colors

$$\{0, \uparrow 1, \uparrow 2, \cdots \uparrow r, \downarrow 1, \downarrow 2, \downarrow r\}$$

as used in the algorithm of Theorem 4.1 in [9]. In addition we also associate two numbers keeping track of how many elements from the current solution we have moved out and how many new vertices have been selected from the bags below it to the current solution. The rest of the update is done in same way as in proof of Theorem 4.1 in [9]. $\square$

Theorem 3. *Let $h : \mathbb{N} \to \mathbb{N}$ be a given function. Then k-*LS-*r-*Center *can be solved in time $(2r+1)^{h(2k)} \cdot |G|^{\mathcal{O}(1)}$ for graphs $G \in \mathscr{G}_h$.*

Proof. Let (G, w, S, k) be an instance of k-LS-r-Center. Since $G \in \mathscr{G}_h$, we have that the treewidth $\mathbf{tw}(G[B(v, 4rk+r)]) \leq h(4rk+r)$ for all $v \in S$. By [16] a tree decomposition of width $OPT\sqrt{logOPT} \leq h(4rk+r)$ can be obtained in polynomial time. By Lemma 3, we can solve k-LS-Generalized r-Center for instances $(B(v, 4rk+r), S_v, T_v, k)$ for every $v \in S$ in time $((2r+1)k)^{h(4rk+r)} \cdot |B(v, 4rk+r)|^{\mathcal{O}(1)}$. In combination with Lemma 2 this completes the proof. $\square$

Let us remark that by Theorem 3, k-LS-r-Center is FPT for planar graphs, graphs of bounded genus and graphs of bounded maximum degree.

In the Vertex Cover problem one seeks for a vertex subset of minimum weight such that every edge has at least one endpoint in this set. The classical k-Vertex Cover problem is one of the well studied, prototype problems in the area of parameterized complexity, and it is solvable in time $\mathcal{O}(1.2738^k + kn)$ on general graphs [7].

Almost directly, Theorem 3 yields an FPT algorithm for k-LS-Vertex Cover on graphs of bounded local treewidth.

PLEASE EXPLAIN WHY DO WE NEED CONDITION $h(i) \geq 2$. (YV: The construction increase the treewidth to two in case of a tree. It is now stated explicitly.) SHALL WE MENTION THAT IF WE USE DIRECT APPROACH (AS FOR DOM SET), THEN WE GET $2^{h(k)} \cdot |G|^{\mathcal{O}(1)}$?

Corollary 1. *Let $h : \mathbb{N} \to \mathbb{N}$ be a given function such that $h(i) \geq 2$ for every i. Then k-LS-Vertex Cover can be solved in time $3^{h(2k)} \cdot |G|^{\mathcal{O}(1)}$ for graphs $G \in \mathscr{G}_h$. In particular k-LS-Vertex Cover is FPT for planar graphs, graphs of bounded genus and graphs of bounded maximum degree.*

Proof. We reduce k-LS-Vertex Cover to k-LS-Dominating Set. From an instance (G, S, w, k) of k-LS-Vertex Cover we make an instance (G', S, w', k) of k-LS-Dominating Set. The graph G' is obtained from G by adding a subdivided edge between every pair of adjacent vertices. The weight function w' is equal to w on the vertices of G. For each vertex v on one of the newly subdivided edges we let $w'(v)$ be the maximum weight of any vertex in G. From the construction it follows that a set $S \subseteq V(G)$ is a vertex cover of G if and only if it is a dominating set in G'. Furthermore, if a dominating set contains a vertex v on one of the newly subdivided edges then v can be replaced by either of its neighbors. Thus (G, S, w, k) is a Yes instance of k-LS-Vertex Cover if and only if (G', S, w', k) is a Yes instance of k-LS-Dominating Set. Now notice that adding subdivided edges between adjacent vertices does not increase the treewidth of a graph (unless the graph is a tree, in which case the treewidth become at most 2, giving the condition that $h(i) \geq 2$). Also observe that adding a subdivided edge between two adjacent vertices does not change the distance between any two vertices. Thus if $G \in \mathscr{G}_h$, then $G' \in \mathscr{G}_h$. By Theorem 3, the instance (G', S, w', k) can be resolved in time $3^{(h(2k))} \cdot |G'|^{\mathcal{O}(1)} = 3^{(h(2k))} \cdot |G|^{\mathcal{O}(1)}$, concluding the proof. □

Planar graphs, and more generally, graphs of bounded genus and apex-minor-free graphs, are in $\mathscr{G}_h$ for some linear function h, and Theorem 3 yields the following.

Corollary 2. *On apex-minor-free graphs k-LS-r-Center is solvable in time $r^{\mathcal{O}(k)} \cdot |G|^{\mathcal{O}(1)}$ and k-LS-Vertex Cover is solvable in time $2^{\mathcal{O}(k)} \cdot |G|^{\mathcal{O}(1)}$.*

5.2 Odd Cycle Transversal

All the problems we considered so far have a certain "locality" involved. In the Odd Cycle Transversal problem one seeks for a vertex subset S of minimum weight in a graph, such that every cycle of odd length in the graph contains at lest one vertex from S. In other words, after removal of S, the remaining vertices induce a bipartite graph. At first glance Odd Cycle Transversal and its parameterized dual—the Induced Bipartite Subgraph problem, which is to find a vertex subset S of maximum weight such the induced graph $G[S]$ is bipartite— do not look like "local" problems. It does not seem that we can reduce all the local changes in that problems to changes inside a small ball around some of the vertices. However, it is possible by making use of some combinatorial observations to construct local search preserving parameterized reduction to the local search of the Maximum Independent Set problem, which is to find a maximum set of pairwise non-adjacent vertices in a graph.

For ease of presentation we only deal with the unweighted case, but weighted cases can be handled in a similar manner. For a given a graph $G = (V, E)$, we define a new graph $\widetilde{G} = (\widetilde{V}, \widetilde{E})$ as follows. Let $V_i = \{u_i \mid u \in V\}$, $i \in \{1, 2\}$. The vertex set $\widetilde{V}$ consists of two copies of V, i.e. $\widetilde{V} = V_1 \cup V_2$ and $\widetilde{E} = \{u_1u_2 \mid u \in V\} \cup \{u_iv_i \mid uv \in E, i \in \{1, 2\}\}$. In other words, $\widetilde{G}$ is obtained by taking two disjoint copies of G and by adding a perfect matching such that the endpoints of every matching edge are the copies of the same vertex. For a set $T \subseteq V$ such that $G[T]$ is a bipartite graph with bipartition T_1 and T_2, we denote by $\widetilde{T}$ the set $\widetilde{T_1} = \{u_1 \mid u \in T_1\} \cup \widetilde{T_2} = \{u_2 \mid u \in T_2\}$ in $\widetilde{V}$. The graph $\widetilde{G}$ has some interesting properties which we make use of when designing the local search algorithm for Odd Cycle Transversal.

Lemma 4. *The following are equivalent*

(i) (G, S, k) *is an* Yes *instance of* k-LS-Induced Bipartite Subgraph,

(ii) $(\widetilde{G}, \widetilde{S}, k)$ *is an* Yes *instance of* k-LS-Independent Set.

Proof. $(i) \implies (ii)$. Since (G, S, k) is an Yes instance of k-LS-Induced Bipartite Subgraph, there is $S' \in \mathcal{N}_k^{en}(S)$ such that $c(S) < c(S')$. Then $\widetilde{S'}_1$ and $\widetilde{S'}_2$ are independent sets. Furthermore, the only edges between two copies of the graph G are matching edges between two copies of the same vertex. This implies that $\widetilde{S'}$ is an independent set of $\widetilde{G}$, $\widetilde{S'} \in \mathcal{N}_k^{en}(\widetilde{S})$ and $c(\widetilde{S}) < c(\widetilde{S'})$. In other words, $(\widetilde{G}, \widetilde{S}, k)$ is an Yes instance of k-LS-Independent Set.

$(ii) \implies (i)$. For the other direction, consider $\widetilde{S'} \in \mathcal{N}_k^{en}(\widetilde{S})$ such that $c(\widetilde{S}) < c(\widetilde{S'})$ for k-LS-Independent Set. Notice that because $\widetilde{S'}$ is an independent set in $\widetilde{G}$, it does not contain both copies of the same vertex $v \in V$. Now, if $\widetilde{S'}_1 = \widetilde{S'} \cap V_1$ and $\widetilde{S'}_2 = \widetilde{S'} \cap V_2$ then $S' = S'_1 \cup S'_2$ induces a bipartite subgraph of the same size in G. The fact that $S' \in \mathcal{N}^{en}_k(S)$ follows easily. $\square$

Lemma 5. *Let $h : \mathbb{N} \to \mathbb{N}$ and $g = 2h + 1$. If $G \in \mathscr{G}_h$, then $\widetilde{G} \in \mathscr{G}_g$.*

Proof. To prove the lemma we need to show that for every $v_i \in V'$, $i \in \{1, 2\}$, and for every $r \in \mathbb{N}$,

$$\mathbf{tw}(\widetilde{G}[N^r_{\widetilde{G}}(v_i)]) \le g(r).$$

We observe that

$$N^r_{\widetilde{G}}(v_i) = N^r_{\widetilde{G}[V_i]}(v_i) \cup N^{r-1}_{\widetilde{G}[V_{3-i}]}(v_{3-i}).$$

We know that $N^r_{\widetilde{G}[V_i]}(v_i)$ is isomorphic to $N^r_G(v)$ and hence $\mathbf{tw}(G[N^r_G(v))] \le h(r)$. Given a tree decomposition of width $h(r)$ for $G[N^r_G(v))]$ we obtain a tree decomposition of width $h(r)$ for $N^r_{\widetilde{G}[V_i]}(v_i)$. Let $(T, \{B_i\}_{i \in T_V})$ be such a decomposition. We construct a tree decomposition for $\widetilde{G}[N^r_{\widetilde{G}}(v_i)]$ with the same tree T but for every bag B_i containing u_i such that $u_{3-i} \in N^{r-1}_{\widetilde{G}[V_{3-i}]}(v_{3-i})$, we add u_{3-i} to B_i. It is easy to check that this forms a valid tree decomposition. The upper bound on the width follows from the fact that every bag contains at most $2h(r) + 2$ vertices. $\square$

By Corollary 1, we have that k-LS-Vertex Cover is FPT on graphs of bounded local treewidth. Since a set S is a vertex cover of a graph $G = (V, E)$ if and only if $V \setminus S$ is an independent set, we have that the same also holds for k-LS-Independent Set. Combining Corollary 1 with Lemmata 4 and 5, we arrive at the following.

Theorem 4. *Let $h : \mathbb{N} \to \mathbb{N}$ be a given function. Then k-*LS-Odd Cycle Transversal *and k-*LS-Induced Bipartite Subgraph *can be solved in time $3^{g(2k)} \cdot |G|^{\mathcal{O}(1)}$ for graphs $G \in \mathscr{G}_h$ where $g = 2h + 1$.*

As in the case with k-LS-r-Center, we can conclude with the following.

Corollary 3. *On apex-minor-free graphs k-*LS-Odd Cycle Transversal *is solvable in time $2^{\mathcal{O}(k)} \cdot |G|^{\mathcal{O}(1)}$.*

6 Graph Paritioning Problems

In this section we look at local search algorithms for graph partitioning problems such as Max-Cut and Min- (Max-) Bisection.

Let $G = (V, E)$ be a given graph and $w : E \to \mathbb{R}^+$ be a weight function. Then Max-Cut asks for a partition of V into V_1 and V_2 such that the total weight of edges (u, v) with $u \in V_1$ and $v \in V_2$ is maximized. In Max- (Min-) Bisection the objective is to find a partition of V into sets V_1 and V_2 such that

- $\lfloor |V|/2 \rfloor \le |V_1| \le |V_2| \le \lceil |V|/2 \rceil$, and
- the total weight of edges (u, v) with $u \in V_1$ and $v \in V_2$ is maximized (minimized).

We give an algorithm for a local search variant of MIN-BISECTION. Others follow along similar lines. Given a partition (V_1, V_2) let $\mathcal{E}(V_1, V_2)$ be the edges with one endpoint in V_1 and other in V_2, and let $c((V_1, V_2)) = \sum_{e \in \mathcal{E}(V_1, V_2)} w(e)$. We now define the notion of $\mathcal{N}_k^{en}(V_1, V_2)$ for this problem. A partition $(V_1', V_2') \in \mathcal{N}_k(V_1, V_2)$ if there exist subsets $X \subseteq V_1$ and $Y \subseteq V_2$ such that (a) $|X| = |Y| = r$, $r \le k$ and (b) $V_1' = (V_1 \setminus X) \cup Y$ and $V_2' = (V_2 \setminus Y) \cup X$.

Theorem 5. *Let $h : \mathbb{N} \to \mathbb{N}$ be a given function and $\mathscr{G}$ be a minor closed subclass of $\mathscr{G}_h$. Then k-*LS-MINIMUM-BISECTION*, k-*LS-MAXIMUM-BISECTION *and* k-LS-MAXIMUM-CUT *can be solved in time $2^{h(ck)} \cdot |G|^{\mathcal{O}(1)}$ for graphs $G \in \mathscr{G}$.*

Proof. We give the proof for k-LS-MINIMUM-BISECTION, and the proofs for other problems follow the same arguments. Let $(G, w, (V_1, V_2), k)$ be the input to k-LS-MINIMUM-BISECTION. In the first part of the proof we show that we can solve k-LS-MINIMUM-BISECTION by solving an equivalent problem on graphs of bounded diameter (or bounded treewidth).

Reducing to graphs of bounded treewidth: We start with a BFS starting at a vertex $v \in V$. Let the layers created by doing BFS on v be

$$L_0^v, L_1^v, \ldots, L_t^v.$$

If $t \le 6k + 10$, we move to the second phase of the algorithm. From now onwards we assume that $t > 6k + 3$. We create thick layers from the above layers:

$$W_i^v = \bigcup_{j=3i}^{3i+2} L_j^v,$$

where $i \in \{0, \ldots, s = \lfloor \frac{t+1}{3} \rfloor\}$. The last thick layer may contain less than 3 layers. Now we make following partition of the vertex set T_q, $q \in \{0, \ldots, 2k+1\}$. We define the following partition of V

$$T_q = \bigcup W_{q+i(2k+2)}, \; i \in \left\{0, \ldots, \left\lfloor \frac{s+1}{2k+2} \right\rfloor \right\}.$$

If $(G, w, (V_1, V_2), k)$ is an YES instance, then there exists a partition $(V_1', V_2') \in \mathcal{N}_k(V_1, V_2)$ such that the total number of vertices participating in exchange is at most $2k$. By the pigeonhole principle, there is T_a that does not contain any of these changed vertices and does not contain W_s. We can find the desired T_a by trying all T_q 's.

Now for every $W_i^v = \bigcup_{j=3i}^{3i+2} L_j^v$ contained in T_a, we remove the vertices of L_{3i+1}^v (that is, the central layer). Let this set of vertices be called V' and the resultant graph be G' with connected components $C_1, \ldots, C_r$. We show that each connected component C_i of G' has bounded treewidth. More precisely, every connected component C_i of G' is a subset of at most $6k + 10$ layers of the BFS. If we start with G, delete all BFS layers outside of these layers and contract all BFS layers inside of these layers into v we obtain a minor H of G. H has diameter at most $6k + 11$, and also H contains C_i as an induced subgraph. Since every minor of G has bounded local treewidth, C_i has bounded treewidth, that is $\mathbf{tw}(C_i) \leq h(6k+10)$ for every i. The reason for removing central layers from each W_i^v is that this retains all the edges which could participate in improved cuts, as well as reduces the treewidth of the graph. In certain sense, the first and third layers of vertices in W_i^v shields the vertices of the middle layer by not participating in change. Notice that since every connected component of G' has bounded treewidth, G' itself has also bounded treewidth.

Finding appropriate solutions using Dynamic Programming: Let $\widetilde{V_1} = V_1 \setminus V'$ and $\widetilde{V_2} = V_2 \setminus V'$. Furthermore $T' = T_a \setminus V'$. Then there exists $(V_1', V_2') \in \mathcal{N}_k^{en}(V_1, V_2)$ such that $c(V_1, V_2) > c(V_1', V_2')$ if and only if there exists $(\widetilde{V}_1', \widetilde{V}_2') \in \mathcal{N}_k^{en}(\widetilde{V_1}, \widetilde{V_2})$ such that $c(\widetilde{V_1}, \widetilde{V_2}) > c(\widetilde{V}_1', \widetilde{V}_2')$ and $T' \cap (\widetilde{V}_1' \triangle \widetilde{V}_1) = \emptyset$ (that is, changed vertices should not include any vertex from T'). Searching for $(\widetilde{V}_1', \widetilde{V}_2')$ reduces to a problem in G', a graph of bounded treewidth. We can solve this problem using standard dynamic programming over graphs of bounded treewidth in time $2^{h(6k+11)} \cdot n^{\mathcal{O}(1)}$. This step is almost identical to the algorithm finding Max- (Min-) Bisection in graphs of bounded treewidth (see, e.g. [21]). We can also keep appropriate information during dynamic programming to find the desired cut explicitly. This concludes the proof of the theorem. $\square$

Let us note that planar graphs, and more generally bounded genus graphs, and apex-minor free graphs, are the minor-closed classes of graphs of bounded local treewidth. By the result of Demaine and Hajiaghayi [11] $h(r)$ is a linear function on minor closed families of graphs which do not contain some fixed apex graph as a minor. Thus by Theorem 5, we have the following.

Corollary 4. *k-LS-Minimum-Bisection, k-LS-Maximum-Bisection and k-LS-Maximum-Cut can be solved in time $2^{\mathcal{O}(k))} \cdot |G|^{\mathcal{O}(1)}$ on apex-minor-free graphs.*

7 FPT Local Search Algorithms for H Minor Free Graphs

The results for k-LS variant for various problems considered in the previous sections can be extended for graphs excluding a fixed graph H as a minor in a non-trivial manner. This extension is based on the structural theorem of Robertson and Seymour [36], which provides a description of an H-minor free graph in terms of clique-sum decomposition over almost embeddible graphs. By making use of this decomposition it is possible to replace dynamic programming algorithms from previous section over graphs of bounded treewidth by dynamic programming over "clique-sum decomposition of H-minor graphs". For all our algorithms for k-LS problems for H-minor free graphs, we will eventually reach a stage where we need to solve an "appropriate" problem in graphs of bounded diameter (previously at this step we applied a dynamic programming algorithm over graphs of bounded treewidth). We first obtain a clique-sum decomposition for the input graph G and then do two layer dynamic programming over clique-sum decomposition as done in [10, 19].

For this we need to replace our dynamic programming algorithms over graphs of bounded treewidth with dynamic programming over "clique-sum decomposition of H-minor graphs".

To make our presentation clear, we restrict ourselves to the k-LS-Dominating Set problem. Before describing the structural theorem of Robertson and Seymour which we use crucially we need some definitions.

Definition 5 (Clique-Sums). *Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two disjoint graphs, and $k \geq 0$ an integer. For $i = 1, 2$, let $W_i \subset V_i$, form a clique of size h and let G'_i be the graph obtained from G_i by removing a set of edges (possibly empty) from the clique $G_i[W_i]$. Let $F : W_1 \to W_2$ be a bijection between W_1 and W_2. We define the* h-clique-sum *or the* h-sum *of G_1 and G_2, denoted by $G_1 \oplus_{h,F} G_2$, or simply $G_1 \oplus G_2$ if there is no confusion, as the graph obtained by taking the union of G'_1 and G'_2 by identifying $w \in W_1$ with $F(w) \in W_2$, and by removing all the multiple edges. The images of the vertices of W_1 and W_2 in $G_i \oplus G_2$ is called the* join *of the sum.*

We remark that $\oplus$ is not well defined; different choices of G'_i and the bijection F could give different clique-sums. A sequence of h-sums, not necessarily unique, which result in a graph G, is called a *clique-sum decomposition* of G.

Definition 6 (h-nearly embeddable graphs). *Let Σ be a surface with boundary cycles $C_1, \ldots, C_h$. A graph G is h-nearly embeddable in Σ, if G has a subset X of size at most h, called* apices*, such that there are (possibly empty) subgraphs $G_0, \ldots, G_h$ of $G \setminus X$ such that*

- *$G \setminus X = G_0 \cup \cdots \cup G_h$,*
- *G_0 is embeddable in Σ, we fix an embedding of G_0,*
- *$G_1, \ldots, G_h$ are pairwise disjoint,*
- *for $1 \leq \cdots \leq h$, let $U_i := \{u_{i_1}, \ldots, u_{i_{m_i}}\} = V(G_0) \cap V(G_i)$, G_i has a path decomposition (B_{ij}), $1 \leq j \leq m_i$, of width at most h such that*
  - *for $1 \leq i \leq h$ and for $1 \leq j \leq m_i$ we have $u_j \in B_{ij}$*
  - *for $1 \leq i \leq h$, we have $V(G_0) \cap C_i = \{u_{i_1}, \ldots, u_{i_{m_i}}\}$ and the points $u_{i_1}, \ldots, u_{i_{m_i}}$ appear on C_i in this order (either if we walk clockwise or anti-clockwise).*

The class of graphs h-nearly embeddable in a fixed surface Σ has linear local treewidth after removing the set of apices. More specifically, the result of [36] which was made algorithmic by [12], states the following:

Theorem 6 ([36], [12]). *For every graph H there exists an integer h, depending only on the size of H, such that every graph excluding H as a minor can be obtained by h-clique sums from graphs that can be h-nearly embedded in a surface Σ in which H can not be embedded and such a clique-sum decomposition can be obtained in time $n^{\mathcal{O}(1)}$. The exponent in the running time depends only on H.*

Let G be a H-minor free graph, and $(T, \mathcal{B} = \{B_a\})$ be a clique-sum decomposition of G obtained in polynomial time by Theorem 7. Given this rooted tree T, we define $A_a := B_a \cap B_{p(a)}$ where $p(a)$ is the unique parent of the vertex a in T, and $A_r = \emptyset$. Let $\widehat{B}_a$ be the graph obtained from B_a by adding all possible edges between the vertices of A_t and also between the vertices of A_s, for each child s of t, making A_t and A_s's as cliques (these are also called *torso* in the literature [19]). In this way, G becomes an h-clique sum of the graphs $\widehat{B}_a$, according to the above tree T and can also be viewed as a tree decomposition given by $(T, \mathcal{B} = \{B_a\})$, where each $\widehat{B}_a$ is h-nearly embeddable in a surface Σ in which H can not be embedded. Let X_a be the set of apices of $\hat{B}_a$. Then $|X_a| \leq h$, and $\widehat{B}_a \setminus X_a$ has linear local treewidth. By G_a we denote the subgraph induced by all vertices of $B_a \bigcup (\cup_s B_s)$, s being a descendant of a in T.

The structural theorem we need, to obtain the clique-sum decomposition for H-minor graphs was given by Robertson and Seymour and made algorithmic by Demaine et al. [12]

Theorem 7 (Robertson and Seymour [36], Demaine et al. [12]). *For every graph H there exists an integer h, depending only on the size of H, such that every graph excluding H as a minor can be obtained by h-clique sums from graphs that can be h-nearly embedded in a surface Σ in which H can not be embedded and such a clique-sum decomposition can be obtained in time $n^{\mathcal{O}(1)}$. The exponent in the running time depends only on H.*

For all our algorithms for k-LS problems for H-minor free graphs, we will eventually reach a stage where we need to solve an "appropriate" problem in graphs of bounded diameter (previously at this step we applied a dynamic programming algorithm over graphs of bounded treewidth). Here, we first obtain a clique-sum decomposition for the input graph G using Theorem 7. We then do two layer dynamic programming over clique-sum decomposition as done in [10, 19], though we need a non trivial modification to cope with the local search variant of the problem.

THE PROOF OF THE FOLLOWING THEOREM REQUIRES MORE WORK!

Theorem 8. *k-LS-Vertex Cover, k-LS-r-Center, and k-LS-Odd Cycle Transversal are FPT on the class of H-minor free graphs.*

Proof. Let (G, w, S, k) be an instance of k-LS-Dominating Set where G excludes a fixed graph H as a minor. Lemma 1 implies that if the given instance is an Yes instance then there exists a solution $S' \in \mathcal{N}_k^{en}(S)$ with $c(S') < c(S)$ and $z \in S \setminus S'$ such that $S \triangle S' \subseteq B(z, 4rk)$. Hence we know that all the changed vertices, those that go out and those that will come in (that is, vertices in the set $S \triangle S'$) are in $B(v, 4rk)$. Let $T_v := (S \cap (\cup_{j=4rk+1}^{4rk+r} L_j^v))$. Furthermore, for $v \in S$, let $S_v = S \cap B(v, 4rk + r)$. For every $v \in S$, we get the following instance $(B(v, 4rk + r), w, S_v, T_v, k)$ for k-LS-Generalized Dominating Set. An instance (G, w, S, k) is an Yes instance for k-LS-Dominating Set if and only if there exists a $v \in S$ for which $(B(v, 4rk+r), w, S_v, T_v, k)$ is an Yes instance for k-LS-Generalized Dominating Set. Now we solve k-LS-Generalized Dominating Set in time $f(4rk+r)\cdot|B(v, 4rk+r)|^{\mathcal{O}(1)}$ for $G[B(v, 4rk+r)]$ for $v \in S$ which implies that we can solve k-LS-Dominating Set for G in time $f(4rk+r)\cdot|G|^{\mathcal{O}(1)}$.

For our proof we do two level of dynamic programming over clique-sum decomposition as done in [10]. We give a brief sketch of this dynamic programming here. Let $\mathcal{G} = G[B(v, 4rk + r)]$ for some vertex $v \in S$. We obtain a clique-sum decomposition $(T, \mathcal{B} = \{B_a\})$ for $\mathcal{G}$ using Theorem 7. After we obtain a clique-sum decomposition $(T, \mathcal{B} = \{B_a\})$ for $\mathcal{G}$, we do a dynamic programming starting from the leaf of the tree and moving towards the root.

In the dynamic programming for a fixed bag p, we guess the vertices of torso and apices $(A_p \cup X_p)$ and guess whether they are going to be part of the changed solution or not. If we decide that a vertex is not going to be part of the 'changed' dominating set, then we guess a neighbor of this vertex which will be in the 'changed' dominating set. Here we make at most $n^{\mathcal{O}(h)}$ guesses. Furthermore for every guess we also keep track of how many vertices from S have been moved out and how many new vertices have been added to the solution. For every fixed guess and pair (k_1, k_2), $0 \leq k_1, k_2 \leq k$, we store 1 or 0 based on whether there exists a dominating set S' such that

$S \setminus S' \leq k_1$ and $S' \setminus S \leq k_2$ of weight strictly less than $c(S)$ in $\mathcal{G}_p$. This is done by making use of tables stored at its children and by observing the fact that (a) $\mathcal{G}$ is of bounded diameter and (b) the graph induced on the vertices of bag p has bounded local treewidth after the removal of apices. Hence this step can be carried out by the normal dynamic programming over graphs of bounded treewidth. An algorithm returns YES and a changed solution if there exists a guess for the vertices in $A_r = \emptyset$ and X_r for which we have stored 1 corresponding to the pair (k, k). $\square$

8 Hardness Results for Local Search in Graphs of Bounded Degeneracy

Finally, we observe that, in some sense, H-minor free graphs are the most general classes of graphs for which positive results for these local search problems can be expected. We show that for more general classes of sparse graphs many local search problems become $W[1]$-hard. Thus the existence of k-exchange FPT algorithms for these problems would contradict the widely believed assumption from parameterized complexity, namely $FPT \neq W[1]$.

In this section we show that for many local search problems the k-LS version remains $W[1]$-hard even for graphs of bounded degeneracy. This means that in some sense, H-minor free graphs are the most general classes of graphs for which positive results for these local search problems can be expected.

Theorem 9. *k-LS-ODD CYCLE TRANSVERSAL is $W[1]$-hard even when restricted to 2-degenerate graphs. Furthermore, k-LS-INDEPENDENT SET and k-LS-DOMINATING SET are $W[1]$-hard when restricted to 3-degenerate graphs.*

Proof. We first prove that k-LS-ODD CYCLE TRANSVERSAL is $W[1]$-hard in 2-degenerate graphs by reducing from k-LS-ODD CYCLE TRANSVERSAL in general graphs. On input (G, S, k) to k-LS-ODD CYCLE TRANSVERSAL in general graphs we make a 2-degenerate graph G' by subdividing every edge of G twice. The graph (G', S, k) is an instance of k-LS-ODD CYCLE TRANSVERSAL in 2-degenerate graphs. Now, cycles in G correspond to cycles in G'. Furthermore the length of a cycle in G and the length if its corresponding cycle in G' have the same parity. Thus, if (G, S, k) is a YES instance of k-LS-ODD CYCLE TRANSVERSAL then so is (G', S, k). In the other direction, suppose (G', S, k) is a YES instance of k-LS-ODD CYCLE TRANSVERSAL. Then there is an odd cycle transversal S' of G' with $|S \setminus S'| \leq k$ and $|S'| < |S|$. Whenever an odd cycle transversal contains a degree 2 vertex v, it can be replaced by any of its two neighbors. Therefore, without loss of generality $S' \subseteq V(G)$ and thus S' is an odd cycle transversal in G. Hence (G, S, k) is a YES instance of k-LS-ODD CYCLE TRANSVERSAL, implying that k-LS-ODD CYCLE TRANSVERSAL is $W[1]$-hard in 2-degenerate graphs.

Since k-LS-ODD CYCLE TRANSVERSAL is $W[1]$-hard in 2-degenerate graphs, so is k-LS-INDUCED BIPARTITE SUBGRAPH. To show that k-LS-INDEPENDENT SET is $W[1]$-hard in 3-degenerate graphs it is sufficient to observe that the reduction from k-LS-ODD CYCLE TRANSVERSAL to k-LS-INDEPENDENT SET presented in Section 5.2 takes graphs of degeneracy 2 to graphs of degeneracy 3. Now, since k-LS-INDEPENDENT SET is $W[1]$-hard in 3-degenerate graphs then so is k-LS-VERTEX COVER. To reduce k-LS-VERTEX COVER in 3-degenerate graphs to k-LS-DOMINATING SET in 3-degenerate graphs it enough to observe that the reduction from k-LS-VERTEX COVER to k-LS-DOMINATING SET used in the proof of Theorem 1 takes unweighted 3-degenerate graphs to unweighted 3-degenerate graphs. □

9 Conclusions and Future Directions

In this paper we initiated the study of parameterized complexity for local search problems on graphs. We have shown that, one can search k-exchange neighborhood of a solution significantly better than the brute force for many natural problems for general classes of sparse graphs. We also provided hardness proofs indicating that most of our results can not be further generalized. There are several open questions that need to be resolved and directions for further research.

- All our algorithmic results either depend on an observation that allows us to consider small diameter graphs, or are reducible to problems where such an approach is feasible. An important problem for which these techniques do not seem to be applicable is TRAVELLING SALESMAN PROBLEM. For general metric, the problem was shown to be $W[1]$ by Marx [29] however the complexity of the planar graph metric variant of the problem remains open. Other local search problems that seem to inhibit similar non-local properties include FEEDBACK VERTEX SET and CONNECTED DOMINATING SET.

- Running times of our algorithms on planar graphs are of the form $\mathcal{O}(2^{\mathcal{O}(k)}n)$. Is such exponential dependence optimal (up to some assumption from complexity theory) or algorithms of running time $\mathcal{O}(2^{o(k)}n^c)$ are feasible?

- Are there k-exchange problems that have kernels of polynomial size?

References

[1] E. H. L. Aarts, J. Korst, and P. J. M. van Laarhoven, *Simulated annealing*, Local Search in Combintaorial Optimization, Wiley, (1997), pp. 91–120.

[2] E. H. L. Aarts and J. K. Lenstra, *Local Search in Combinatorial Optimization*, Princeton University Press, 1997.

[3] P. Alimonti, *Non-oblivious local search for graph and hypergraph coloring problems*, in WG, vol. 1017 of LNCS, Springer, 1995, pp. 167–180.

[4] ——, *Non-oblivious local search for MAX 2-CCSP with application to max dicut*, in WG, vol. 1335 of LNCS, Springer, 1997, pp. 2–14.

[5] F. Bock, *An algorithm for solving 'traveling-salesman' and related network optimization problems*, Research report, Armour Research Foundation, (1958).

[6] H. L. Bodlaender, *A partial k-arboretum of graphs with bounded treewidth*, Theor. Comput. Sci., 209 (1998), pp. 1–45.

[7] J. Chen, I. A. Kanj, and G. Xia, *Improved parameterized upper bounds for vertex cover*, in MFCS, 2006, pp. 238–249.

[8] G. Croes, *A method for solving traveling-salesman problems*, Operations Research, 6 (1958), pp. 791–812.

[9] E. D. Demaine, F. V. Fomin, M. T. Hajiaghayi, and D. M. Thilikos, *Fixed-parameter algorithms for (k, r)-center in planar graphs and map graphs*, ACM Transactions on Algorithms, 1 (2005), pp. 33–47.

[10] ——, *Subexponential parameterized algorithms on bounded-genus graphs and -minor-free graphs*, J. ACM, 52 (2005), pp. 866–893.

[11] E. D. Demaine and M. T. Hajiaghayi, *Equivalence of local treewidth and linear local treewidth and its algorithmic applications*, in SODA, 2004, pp. 840–849.

[12] E. D. Demaine, M. T. Hajiaghayi, and K. ichi Kawarabayashi, *Algorithmic graph minor theory: Decomposition, approximation, and coloring*, in FOCS, 2005, pp. 637–646.

[13] R. G. Downey and M. R. Fellows, *Parameterized complexity*, Springer-Verlag, New York, 1999.

[14] D. Eppstein, *Diameter and treewidth in minor-closed graph families*, Algorithmica, 27 (2000), pp. 275–291.

[15] U. Faigle and W. Kern, *Note on the convergence of simulatd annealing algorithms*, Siam Journal on Control and Optimization, 29 (1991), pp. 153–159.

[16] U. Feige, M. Hajiaghayi, and J. R. Lee, *Improved approximation algorithms for minimum weight vertex separators*, SIAM J. Comput., 38 (2008), pp. 629–657.

[17] M. Fellows, F. V. Fomin, D. Lokshtanov, F. Rosamond, S. Saurabh, and Y. Villanger, *Local search: Is brute-force avoidable?*, in IJCAI, AAAI, 2009, pp. 486–491.

[18] J. Flum and M. Grohe, *Parameterized Complexity Theory*, Springer-Verlag, Berlin, 2006.

[19] M. Grohe, *Local tree-width, excluded minors, and approximation algorithms*, Combinatorica, 23 (2003), pp. 613–632.

[20] A. Gupta and É. Tardos, *A constant factor approximation algorithm for a class of classification problems*, in STOC, 2000, pp. 652–658.

[21] K. Jansen, M. Karpinski, A. Lingas, and E. Seidel, *Polynomial time approximation schemes for max-bisection on planar and geometric graphs*, SIAM J. Comput., 35 (2005), pp. 110–119.

[22] D. S. Johnson, C. H. Papadimitriou, and M. Yannakakis, *How easy is local search?*, J. Comput. Syst. Sci., 37 (1988), pp. 79–100.

[23] S. Khanna, R. Motwani, M. Sudan, and U. V. Vazirani, *On syntactic versus computational views of approximability*, SIAM J. Comput., 28 (1998), pp. 164–191.

[24] S. Khot and V. Raman, *Parameterized complexity of finding subgraphs with hereditary properties*, Theor. Comput. Sci., 289 (2002), pp. 997–1008.

[25] S. Khuller, R. Bhatia, and R. Pless, *On local search and placement of meters in networks*, SIAM J. Comput., 32 (2003), pp. 470–487.

[26] A. Krokhin and D. Marx, *On the hardness of losing weight*, in ICALP, vol. 5125 of LNCS, Springer, 2008, pp. 662–673.

[27] S. Lin and B. W. Kernighan, *An effective heuristic algorithm for traveling-salesman problem*, Operations Research, 21 (1973), pp. 498–516.

[28] D. Marx, *Local search*, Parameterized Complexity Newsletter, 3 (2008), pp. 7–8.

[29] ——, *Searching the k-change neighborhood for TSP is W[1]-hard*, Oper. Res. Lett., 36 (2008), pp. 31–36.

[30] W. Michiels, E. H. L. Aarts, and J. Korst, *Theoretical Aspects of Local Search*, Springer-Verlag, 2007.

[31] R. Niedermeier, *Invitation to fixed-parameter algorithms*, Oxford University Press, 2006.

[32] C. H. Papadimitriou and K. Steiglitz, *On the complexity of local search for the traveling salesman problem*, SIAM J. Comput., 6 (1977), pp. 76–83.

[33] ——, *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall, 1982.

[34] N. Robertson and P. D. Seymour, *Graph minors. v. excluding a planar graph*, J. Comb. Theory, Ser. B, 41 (1986), pp. 92–114.

[35] ——, *Graph minors. x. obstructions to tree-decomposition*, J. Comb. Theory, Ser. B, 52 (1991), pp. 153–190.

[36] ——, *Graph minors. xvi. excluding a non-planar graph*, J. Comb. Theory, Ser. B, 89 (2003), pp. 43–76.

[37] S. Szeider, *The parameterized complexity of k-flip local search for SAT and MAX SAT*, in SAT, vol. 5584 of Lecture Notes in Computer Science, Springer, 2009, pp. 276–283.