

Local Search: Is brute-force avoidable?

Michael R. Fellows* Fedor V. Fomin† Daniel Lokshantov†
Frances A. Rosamond* Saket Saurabh † Yngve Villanger†

July 3, 2008

Abstract

Heuristic algorithms are the most common practical approach for solving many intractable optimization problems. In general, such algorithms start from a feasible solution and iteratively try to improve the current solution. Local search algorithms, also known as neighborhood search algorithms, form a large class of improvement algorithms. To perform local search, a problem specific neighborhood distance function is defined on the solution space and a better solution is searched for in the neighborhood of the current solution. In particular, 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 $n^{O(k)}$ time, which is not practical even for very small values of k . In this paper 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 $O(\tau(k) \cdot n^c)$, 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 subset problems like VERTEX COVER, ODD CYCLE TRANSVERSAL, DOMINATING SET, and r -CENTER, and partition problems like MAX-CUT and MIN-BISECTION. In particular, on planar graphs, all our algorithms searching for a k -local improvement run in time $O(2^{O(k)} \cdot n^2)$. Thus the running time of our algorithms is strongly polynomial for $k = O(\log n)$. Finally, we complement the algorithms with complexity results indicating that—brute force search is unavoidable—in more general classes of sparse graphs.

*University of Newcastle, Newcastle, Australia. {michael.fellows|frances.rosamond}@newcastle.edu.au

†Department of Informatics, University of Bergen, N-5020 Bergen, Norway.
{fomin|daniello|saket.saurabh|yngvev}@ii.uib.no

1 Introduction

One of the most common approaches used to attack hard optimization problems in practice is the use of local search. Local search is also used as a subroutine in iterated algorithms and evolutionary algorithms. The use of local search in combinatorial optimization and operations research has a long history dating 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, this is called the k -exchange neighborhood [24, 29]. Another classical example is the MIN-BISECTION problem [19], 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 [30, 3, 4, 20, 18]. The second direction of the theoretical work is on the asymptotic convergence of local search in probabilistic settings, such as simulated annealing [14, 1]. 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 [19] 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, 27] 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 $n^{O(k)}$, 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 $O(\tau(k)n^c)$, 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 $\tau(k) \cdot n^{O(1)}$, 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 the books [12, 16, 28].

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

local search problems [25]:

“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 was done by Marx [26] who proved that finding a local improvement in k -exchange neighborhood for TSP is $W[1]$ -hard. Surprisingly, finding k -local optimum is not $W[1]$ -hard for every NP-hard problem. A striking example can be found in the work of Khuller, Bhatia, and Pless [22] 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 [22] is that checking whether it is possible to improve a solution by replacing at most k edges can be done in time $O(n^2 + n\tau(k))$, i.e., it is FPT parameterized by k . Very recently, Krokhn and Marx [23] investigated the local search problem for finding a minimum weight assignment for a Boolean constraint satisfaction instance.

Our results. In this paper, we initiate the systematic study of parameterized complexity of local search for different graph problems. 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. 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 $O(2^{O(k)} \cdot n^2)$ for many of the problems mentioned above. We also extend these results to more general classes of graphs, namely, graphs excluding a fixed graph as a minor. Finally, we show that existence of FPT algorithms is highly unlikely for larger classes of sparse graphs. We prove that most of the problems remain $W[1]$ -hard on 3-degenerated graphs.

2 Preliminaries

Let $G = (V, E)$ be an undirected graph where V (or $V(G)$) is the set of vertices and E (or $E(G)$) is the set of edges. We denote the number of vertices by n and number of edges by m . 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 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 $A \subseteq V(G)$. Given a weight function $w : V \rightarrow \mathbb{R}^+ \cup \{0\}$ and $A \subseteq V(G)$, $w(A) = \sum_{u \in A} w(u)$. Given two sets S_1 and S_2 , the *symmetric difference* between S_1 and S_2 is defined as $S_1 \triangle S_2 = (S_1 \setminus S_2) \cup (S_2 \setminus S_1)$.

Given an edge $e = (u, v)$ of a graph G , the graph G/e is obtained by contracting the edge (u, v) ; that is, we get G/e 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 the removal of one vertex makes it a planar graph.

A *tree decomposition* of a (undirected) graph G is a pair (X, T) where 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 $V(G)$ such that (a) $\bigcup_{i \in V(T)} X_i = V(G)$, (b) for each edge $(v, w) \in E(G)$, there is an $i \in V(T)$ such that $v, w \in X_i$, and (c) for each $v \in V(G)$ the set of nodes $\{i \mid v \in X_i\}$ forms a subtree of T . The *width* of a tree decomposition $(\{X_i \mid i \in V(T)\}, T)$ equals $\max_{i \in V(T)} \{|X_i| - 1\}$. 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* [13, 17].

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

A graph class \mathcal{G} has *bounded local treewidth* if there exists a function $f : \mathbb{N} \rightarrow \mathbb{N}$ such that for each graph $G \in \mathcal{G}$, and for each integer $r \in \mathbb{N}$, we have $\mathbf{ltw}^G(r) \leq f(r)$. By a result of Robertson and Seymour [31] (see also Bodlaender [6]), $f(r)$ can be chosen as $3r$ for planar graphs. Similarly Eppstein [13] showed that $f(r)$ 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 $g(\Sigma)$.

3 Problem Definitions and 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. Additional necessary definitions of combinatorial optimization problems, neighborhood function and locally optimum are deferred to Appendix A due to space constraints.

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, 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 \mathcal{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 $w : V \rightarrow \mathbb{R}^+$ be a weight function. Then the cost function $c : \mathcal{S} \rightarrow \mathbb{R}^+$ is defined as $\sum_{v \in s} w(v)$ for all $s \in \mathcal{S}$. For a pair of elements $s_1, s_2 \in \mathcal{S}$, 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-WEIGHTED P (k -LS-WEIGHTED P)

Input: A graph $G = (V, E)$, a weight function $w : V \rightarrow \mathbb{R}^+$, a solution S and an integer $k \geq 0$.

Parameter: A positive integer k .

Question: Does there exists 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)?

We refer to the local search version of a particular problem by k -LS-WEIGHTED P , and if we take a unit weight function ($w(v) = 1$ for all $v \in V$), then we refer to it as k -LS- P . 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 consists of five parts: $(G = (V, E), w, \mathcal{M}, S, k)$, where S is an initial solution, $w : V \rightarrow \mathbb{R}^+$, \mathcal{M} is a distance function defined on the solution space \mathcal{S} and $k \in \mathbb{N}$ is a positive integer. If w is a unit weight function then we ignore w and only have four parts $(G = (V, E), \mathcal{M}, S, k)$. Unless otherwise mentioned, we only consider the Hamming distance H in this paper, and hence this term is deleted from the input.

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

In this section we study local search problems for graphs of bounded local treewidth. This is a wide class, which contains planar graphs, graphs of bounded genus and graphs of bounded maximum degree as a proper subclass. We show that, quite surprisingly, many local search problems become fixed parameter tractable in graphs of bounded local treewidth.

4.1 FPT Local Search Algorithms for Domination Problems

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

k -LS-WEIGHTED r -CENTER: Given an undirected graph $G = (V, E)$, with weight function $w : V \rightarrow \mathbb{R}^+$, a r -center $S \subseteq V$ and integers k, r . The problem 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 this is a k -LS- r -CENTER (k -LS- r -C) problem, and for $r = 1$ this is the k -LS-WEIGHTED 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 $w : V \rightarrow \mathbb{R}^+$, such that the cardinality of the symmetric difference $S_1 \Delta S_2$ is at most 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 $F_1 \cup F_2$ is in $B(z, 2pr)$.*

Proof: Let $X = S_1 \setminus S_2$ and $Y = S_2 \setminus S_1$. Furthermore, (a) $X \cap Y = \emptyset$ and (b) $S_1 \setminus X = S_2 \setminus Y = S_1 \cap S_2$. 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 $G^* = (X \cup Y = S_1 \Delta S_2, E')$, where we give an edge between two vertices $u, v \in (X \cup Y)$ if $d(u, v) \leq 2r$, that is, the shortest path distance between u and v is at most $2r$ in G . Let C_1, \dots, 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 $w(X) > w(Y)$, there exists a connected component C_j such that $x > y$ in $\mu(C_j)$. We claim that we can take $S_1 \setminus (X \cap C_j) \cup (Y \cap C_j)$ as the desired S , $F_1 = X \cap C_j$ and $F_2 = Y \cap C_j$.

First, $c(S) = w(S_1) - w(X \cap C_j) + w(Y \cap C_j) < c(S_1)$, since $x > y$. Now we show 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 vertex adjacent in this component 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, $(F_1 \cup F_2) \subseteq B(z, 2pr)$. \square

We define a generalized form of k -LS-WEIGHTED r -CENTER which we call k -LS-GENERALIZED WEIGHTED 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 WEIGHTED r -CENTER looks like (G, w, S, T, k) .

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

Proof: Let (G, w, S, k) be an instance of k -LS-WEIGHTED r -CENTER where $G \in \mathcal{G}$. 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 $S \triangle S' \subseteq B(z, 4rk)$.

We go through every vertex v in S as the desired z and do as follows. We do BFS starting at v . Let the layers created by doing BFS on v be $L_0^v, L_1^v, \dots, L_t^v$. 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 WEIGHTED r -CENTER. In the other case, we first take $4rk + r$ layers; that is,

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

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 $T_v := (S \cap (\cup_{j=4rk+1}^{4rk+r} L_j^v))$. 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 WEIGHTED r -CENTER. An instance (G, w, S, k) is an YES instance for k -LS-WEIGHTED r -CENTER 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 WEIGHTED r -CENTER. Since we can solve k -LS-GENERALIZED WEIGHTED r -CENTER in time $f(4rk + r) \cdot |B(v, 4rk + r)|^{O(1)}$ for $G[B(v, 4rk + r)]$ for $v \in S$, we can solve k -LS-WEIGHTED r -CENTER for G in time $f(4rk + r) \cdot |G|^{O(1)}$. This concludes the lemma. \square

Lemma 3 $[\star]^1$ *Let \mathcal{G} be the class of graphs of treewidth at most t , then k -LS-GENERALIZED WEIGHTED r -CENTER can be solved in time $\mathcal{O}((2r + 1)^t \cdot |G|^{O(1)})$ for graphs $G \in \mathcal{G}$.*

Theorem 1 *Let $h : \mathbb{N} \rightarrow \mathbb{N}$ be a given function. Then k -LS-WEIGHTED r -CENTER can be solved in time $\mathcal{O}((2r + 1)^{h(2k)} \cdot |G|^{O(1)})$ for graphs $G \in \mathcal{G}_h$. In particular k -LS-WEIGHTED r -CENTER is FPT for planar graphs, graphs of bounded genus and graphs of bounded maximum degree.*

Proof: Let (G, w, S, k) be an instance of k -LS-WEIGHTED r -CENTER. Since $G \in \mathcal{G}_h$, the treewidth $\mathbf{tw}(G[B(v, 4rk + r)]) \leq h(4rk + r)$ for all $v \in S$. Using Lemma 3 we solve k -LS-GENERALIZED WEIGHTED r -CENTER for instances $(B(v, 4rk + r), S_v, T_v, k)$ for every $v \in S$ in time $\mathcal{O}((2r + 1)^{h(4rk+r)} \cdot |B(v, 4rk + r)|^{O(1)})$. This in combination with Lemma 2 implies the result. \square

4.2 FPT Local Search Algorithm for Vertex Cover

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]. In this section we give an FPT algorithm for k -LS-WEIGHTED VERTEX COVER. Our result is based on the FPT algorithm for k -LS-WEIGHTED r -CENTER given in the previous section. In fact we

¹Proofs of results labeled with $[\star]$ have been moved to the appendix because of space constraints.

give a *local search preserving parameterized reduction* from k -LS-WEIGHTED VERTEX COVER to k -LS-WEIGHTED DOMINATING SET that preserves the property of local treewidth.

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

Proof: We reduce k -LS-WEIGHTED VERTEX COVER to k -LS-WEIGHTED DOMINATING SET. From an instance (G, S, w, k) of k -LS-WEIGHTED VERTEX COVER we make an instance (G', S, w', k) of k -LS-WEIGHTED 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-WEIGHTED VERTEX COVER if and only if (G', S, w', k) is a YES instance of k -LS-WEIGHTED DOMINATING SET. Now, notice that adding subdivided edges between adjacent vertices can not increase the treewidth of a graph (unless the graph is a tree, in which case the treewidth increases to 2). Also, observe that adding a subdivided edge between two adjacent vertices does not change the distance between any two vertices. Thus, if $G \in \mathcal{G}_h$ then $G' \in \mathcal{G}_h$. By Theorem 1 the instance (G', S, w', k) can be resolved in time $\mathcal{O}(3^{h(2k)} \cdot |G'|^{O(1)}) = \mathcal{O}(3^{h(2k)} \cdot |G|^{O(1)})$ concluding the proof. \square

4.3 FPT Local Search Algorithm for k -LS-ODD CYCLE TRANSVERSAL

All the problems we considered so far had a certain “locality” involved. In this section we look at the ODD CYCLE TRANSVERSAL problem, which at first glance does not look like a local property. For ease of presentation we only deal with the unweighted case, but weighted cases can be handled in a similar manner. Our algorithm is based on a local search preserving parameterized reduction to k -LS-INDEPENDENT SET.

Given a graph $G = (V, E)$, we define a new graph $\tilde{G} = (V', E')$ as follows. Let $V_i = \{u_i \mid u \in V\}$, $i \in \{1, 2\}$. Take $V' = V_1 \cup V_2$ and $E' = \{u_1 u_2 \mid u \in V\} \cup \{u_i v_i \mid uv \in E, i \in \{1, 2\}\}$. Given a set $T \subseteq V$ such that $G[T]$ is a bipartite graph with bipartition T_1 and T_2 then by \tilde{T} denote the set $\tilde{T}_1 = \{u_1 \mid u \in T_1\} \cup \{u_2 \mid u \in T_2\}$ in V' . The graph \tilde{G} has some interesting properties which we make use of when designing the local search algorithm for ODD CYCLE TRANSVERSAL. Our result is based on the reformulation of following lemma.

Lemma 4 ([33]) *A graph G has an induced bipartite subgraph of size t if and only if \tilde{G} has an independent set of size t .*

Lemma 5 $[\star]$ *Let (G, S, k) be an instance of k -LS-INDUCED BIPARTITE SUBGRAPH. Then (G, S, k) is a YES instance of k -LS-INDUCED BIPARTITE SUBGRAPH if and only if $(\tilde{G}, \tilde{S}, k)$ is a YES instance of k -LS-INDEPENDENT SET.*

Lemma 6 $[\star]$ *Let $h : \mathbb{N} \rightarrow \mathbb{N}$ and $g = 2h + 1$ be two given functions. If $G \in \mathcal{G}_h$ then $\tilde{G} \in \mathcal{G}_g$.*

The Theorem 2 implies that that k -LS-VERTEX COVER is fixed parameter tractable 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 get that the same result also holds for k -LS-INDEPENDENT SET. Combining Theorem 2 and Lemmas 5 and 6 we obtain the following result.

Theorem 3 *Let $h : \mathbb{N} \rightarrow \mathbb{N}$ be a given function. Then k -LS-ODD CYCLE TRANSVERSAL and k -LS-INDUCED BIPARTITE SUBGRAPH can be solved in time $\mathcal{O}(3^{g(2k)} \cdot |G|^{O(1)})$ for graphs $G \in \mathcal{G}_h$ where $g = 2h + 1$. In particular k -LS-ODD CYCLE TRANSVERSAL and k -LS-INDUCED BIPARTITE SUBGRAPH are FPT for planar graphs, graphs of bounded genus and graphs of bounded maximum degree.*

Lemma 4 combined with Lemma 6 has an interesting application in finding maximum induced bipartite subgraph for graphs of bounded treewidth.

Theorem 4 *Given a graph G together with a tree decomposition of width w , MINIMUM ODD CYCLE TRANSVERSAL and MAXIMUM INDUCED BIPARTITE SUBGRAPH can be solved in time $\mathcal{O}(3^w \cdot n^{O(1)})$.*

Theorem 4 improves the $\mathcal{O}(27^w \cdot n^{O(1)})$ time algorithm designed by Fiorini et al. [15].

5 Graph Partitioning Problems

In this section we look at local search algorithms for graph partitioning problems such as WEIGHTED MAX-CUT and WEIGHTED MIN- (MAX-) BISECTION. Let $G = (V, E)$ be a given graph and $w : E \rightarrow \mathbb{R}^+$ be a weight function. Then WEIGHTED 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 whereas in WEIGHTED MAX- (MIN-) BISECTION the objective is to find a partition of V into V_1 and V_2 such that (a) $|V_1| = |V_2| = |V|/2$ and (b) 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) $|X| = |Y| = r$, $r \leq k$ and (b) $V'_1 = (V_1 \setminus X) \cup Y$ and $V'_2 = (V_2 \setminus Y) \cup X$.

Theorem 5 *Let $h : \mathbb{N} \rightarrow \mathbb{N}$ be a given function. Then k -LS-WEIGHTED MINIMUM-BISECTION, k -LS-WEIGHTED MAXIMUM-BISECTION and k -LS-WEIGHTED MAXIMUM-CUT can be solved in time $\mathcal{O}(2^{h(ck)} \cdot |G|^{O(1)})$ for graphs $G \in \mathcal{G}_h$ such that for every minor H of G , $H \in \mathcal{G}_h$. In particular k -LS-WEIGHTED MINIMUM-BISECTION is FPT for planar graphs and graphs of bounded genus.*

Proof: We give the proof for k -LS-WEIGHTED 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-WEIGHTED MINIMUM-BISECTION. In the first part of the proof we show that we can solve k -LS-WEIGHTED 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 $L_0^v, L_1^v, \dots, L_t^v$. If $t \leq 6k + 10$, 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 $i \in \{0, \dots, s = \lfloor \frac{t+1}{3} \rfloor\}$. The last thick layer may contain less than 3 layers. Now we make following partition of the vertex set T_q , $q \in \{0, \dots, 2k + 1\}$. We define $T_q = \bigcup W_{q+i(2k+2)}$, $i \in \{0, \dots, \lfloor \frac{s+1}{2k+2} \rfloor\}$. It is clear from the definition of T_q that it partitions the vertex set. If the input is a 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 which will flip its side (or vertices which will participate in change) is upper bounded by $2k$. Using the pigeon hole principle, we conclude that there exist T_a such that it 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, \dots, 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 $\text{tw}(C_i) \leq h(6k + 10)$ 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 $(V'_1, V'_2) \in \mathcal{N}_k^{\text{en}}(V_1, V_2)$ such that $c(V_1, V_2) > c(V'_1, V'_2)$ if and only if there exists $(\widetilde{V}'_1, \widetilde{V}'_2) \in \mathcal{N}_k^{\text{en}}(\widetilde{V}_1, \widetilde{V}_2)$ such that $c(\widetilde{V}_1, \widetilde{V}_2) > c(\widetilde{V}'_1, \widetilde{V}'_2)$ and $T' \cap (\widetilde{V}'_1 \triangle \widetilde{V}'_2) = \emptyset$ (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 $2^{h(6k+11)} \cdot n^{O(1)}$. We can also keep appropriate information during dynamic programming to find the desired cut explicitly. This concludes the proof of the theorem. \square

6 Finding the Limits for FPT Algorithms for k -LS Problems

6.1 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. 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”. We refer to Appendix B for definitions of clique-sum and other related concepts. 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. [11]

Theorem 6 (Robertson and Seymour [32], Demaine et al. [11]) *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 $n^{O(1)}$. 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 11. We then do two layer dynamic programming over clique-sum decomposition as done in [10, 17], though we need a non trivial modification to cope with the local search variant of the problem. We postpone the complete details of this step for the longer version of the paper and provide a sketch in the appendix.

Theorem 7 $[\star]$ *The following local search problems are FPT with respect to k -EXN for H -minor free graphs: k -LS-WEIGHTED VERTEX COVER, k -LS-WEIGHTED INDEPENDENT SET, k -LS-WEIGHTED r -CENTER and k -LS-ODD CYCLE TRANSVERSAL.*

6.2 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 [21] studied the parameterized complexity of Π -SUBSET problems and showed the following result.

Theorem 8 (Khot and Raman [21]) *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 9 $[\star]$ *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 k -LS- Π -SUBSET problem and the k -LS- Π -DELETION problem.*

Theorems 8 and 9 together yield hardness results for the local search variants of many Π -SUBSET problems. Examples include 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 9, one can show the hardness of local search variants of $W[2]$ -hard problems such as DOMINATING SET or INDEPENDENT DOMINATING SET. We arrive at the following corollary.

Corollary 1 *The following local search problems are not in FPT, unless $FPT = W[1]$: k -LS-VERTEX COVER, k -LS-INDEPENDENT SET, k -LS-FEEDBACK VERTEX SET, k -LS-INDUCED FOREST, k -LS-ODD CYCLE TRANSVERSAL, k -LS-INDUCED BIPARTITE SUBGRAPH. The following local search problems are not in FPT, unless $FPT = W[2]$: k -LS-DOMINATING SET, k -LS-INDEPENDENT DOMINATING SET.*

6.3 Hardness Results for Local Search in Graphs of Bounded Degeneracy

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. A graph G is d -degenerated if every induced subgraph of G has a vertex of degree at most d . 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 10 *The problems k -LS-ODD CYCLE TRANSVERSAL and k -LS-FEEDBACK VERTEX SET are $W[1]$ -hard, even 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 of 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. The same construction can be used to reduce k -LS-FEEDBACK VERTEX SET in general graphs to k -LS-FEEDBACK VERTEX SET in 2-degenerate graphs. The correctness proof of this reduction is identical to the proof for k -LS-ODD CYCLE TRANSVERSAL.

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 4.3 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-WEIGHTED VERTEX COVER to k -LS-WEIGHTED DOMINATING SET used in the proof of Theorem 2 takes unweighted 3-degenerate graphs to unweighted 3-degenerate graphs. \square

7 Conclusions and Future Directions

In this paper we studied parameterized complexity of local search for different graph problems. We have shown that, quite surprisingly, one can search k -exchange neighborhood of a solution significantly better than the brute force for many natural problems for several classes of sparse graphs. In particular, we showed that deciding whether a given solution S of weight W to WEIGHTED INDEPENDENT SET, WEIGHTED DOMINATING SET, WEIGHTED ODD CYCLE TRANSVERSAL or WEIGHTED MINIMUM-BISECTION can be improved to another solution S' of weight less than W by exchanging at most k vertices of S can be checked in time $O(2^{O(k)}n)$ in planar graphs. We also showed that these problems admit $O(f(k)n^c)$ time algorithms for more general graph classes and provided hardness proofs that indicate that most of our results can not be further generalized.

There are several open questions that need to be resolved. For instance, 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 the TRAVELLING SALESMAN PROBLEM. Thus, the main problem we leave open in this paper is to resolve the complexity of the k -LS-TRAVELLING SALESMAN PROBLEM in planar graphs. Other local search problems that seem to inhibit similar non-local properties include PLANAR k -LS-FEEDBACK VERTEX SET and PLANAR k -LS-CONNECTED DOMINATING SET and their parametrized complexity remain open. It would also be interesting to see if the local search variant of any reasonable NP-complete graph problem could turn out to be fixed parameter tractable in general graphs. Finally, one wonders whether running times of the form $O(2^{O(k)}n)$ are the best one can hope for local search problems in planar graphs, or whether algorithms running in time $O(2^{o(k)}n^c)$ could be feasible.

References

- [1] E. H. L. AARTS, J. KORST, AND P. J. M. VANLAARHOVEN, *Simulated annealing*, Local Search in Combinatorial 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, 1995, pp. 167–180.
- [4] ———, *Non-oblivious local search for max 2-ccsp with application to max dicut*, in WG, 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, M. T. HAJIAGHAYI, AND K. ICHI KAWARABAYASHI, *Algorithmic graph minor theory: Decomposition, approximation, and coloring*, in FOCS, 2005, pp. 637–646.
- [12] R. G. DOWNEY AND M. R. FELLOWS, *Parameterized complexity*, Springer-Verlag, New York, 1999.
- [13] D. EPPSTEIN, *Diameter and treewidth in minor-closed graph families*, Algorithmica, 27 (2000), pp. 275–291.
- [14] U. FAIGLE AND W. KERN, *Note on the convergence of simulated annealing algorithms*, Siam Journal on Control and Optimization, 29 (1991), pp. 153–159.
- [15] S. FIORINI, N. HARDY, B. REED, AND A. VETTA, *Planar graph bipartization in linear time*, Discrete Applied Mathematics, 156 (2008), pp. 1175–1180.
- [16] J. FLUM AND M. GROHE, *Parameterized Complexity Theory*, Texts in Theoretical Computer Science. An EATCS Series, Springer-Verlag, Berlin, 2006.
- [17] M. GROHE, *Local tree-width, excluded minors, and approximation algorithms*, Combinatorica, 23 (2003), pp. 613–632.
- [18] A. GUPTA AND É. TARDOS, *A constant factor approximation algorithm for a class of classification problems*, in STOC, 2000, pp. 652–658.

- [19] D. S. JOHNSON, C. H. PAPADIMITRIOU, AND M. YANNAKAKIS, *How easy is local search?*, J. Comput. Syst. Sci., 37 (1988), pp. 79–100.
- [20] 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.
- [21] S. KHOT AND V. RAMAN, *Parameterized complexity of finding subgraphs with hereditary properties*, Theor. Comput. Sci., 289 (2002), pp. 997–1008.
- [22] S. KHULLER, R. BHATIA, AND R. PLESS, *On local search and placement of meters in networks*, SIAM J. Comput., 32 (2003), pp. 470–487.
- [23] A. KROKHIN AND D. MARX, *On the hardness of losing weight*, in ICALP, 2008, p. To appear.
- [24] S. LIN AND B. W. KERNIGHAN, *An effective heuristic algorithm for traveling-salesman problem*, Operations Research, 21 (1973), pp. 498–516.
- [25] D. MARX, *Local search*, Parameterized Complexity Newsletter, 3 (2008), pp. 7–8.
- [26] ———, *Searching the k -change neighborhood for tsp is $w[1]$ -hard*, Oper. Res. Lett., 36 (2008), pp. 31–36.
- [27] W. MICHIELS, E. H. L. AARTS, AND J. KORST, *Theoretical Aspects of Local Search*, Monographs in Theoretical Computer Science. An EATCS Series, Springer-Verlag, 2007.
- [28] R. NIEDERMEIER, *Invitation to fixed-parameter algorithms*, vol. 31 of Oxford Lecture Series in Mathematics and its Applications, Oxford University Press, Oxford, 2006.
- [29] 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.
- [30] ———, *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall, 1982.
- [31] N. ROBERTSON AND P. D. SEYMOUR, *Graph minors. v. excluding a planar graph*, J. Comb. Theory, Ser. B, 41 (1986), pp. 92–114.
- [32] ———, *Graph minors. xvi. excluding a non-planar graph*, J. Comb. Theory, Ser. B, 89 (2003), pp. 43–76.
- [33] S. SAURABH, *Exact Algorithms for Optimization and Parameterized versions of some Graph Theoretic Problems*, Homi Bhabha National Institute, Thesis, 2008.

A Framework of Local Search

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

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

Definition 3 Let (\mathcal{S}, f) be an instance of a combinatorial optimization problem P . A neighborhood function is a mapping $\mathcal{N} : \mathcal{S} \rightarrow 2^{\mathcal{S}}$, which defines for each solution $s \in \mathcal{S}$, a set $\mathcal{N}(s) \subseteq \mathcal{S}$ of solutions. The set $\mathcal{N}(s)$ is the neighborhood of solution s and each $s' \in \mathcal{N}(s)$ 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 (\mathcal{S}, c) be an instance of a combinatorial optimization problem P and let \mathcal{N} be a neighborhood function. A solution \hat{s} is *locally optimum* (minimal/maximal) with respect to \mathcal{N} if $c(\hat{s}) \leq c(s)$ ($c(\hat{s}) \geq c(s)$) for all $s \in \mathcal{N}(\hat{s})$ if P is a minimization (maximization) problem. We denote the set of locally optimum solution by \mathcal{S} .

There are various neighborhood structure known in the literature for different problems.

A.1 Proof of Lemma 3

Proof: Let (G, w, S, T, k) be an instance of k -LS-GENERALIZED WEIGHTED r -CENTER. The proof follows the same pattern as the proof of Theorem 4.1 in [9], and so we just give a proof-sketch. We increase the size of the table kept for each of the bags in the tree decomposition in Theorem 4.1 of [9]. We associate $2r + 1$ colors to

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

each of the vertices, and we also associate a tuple from

$$\{0, 1, \dots, k\} \times \{0, 1, \dots, k\}$$

to each coloring of the bags of the tree decomposition, remembering 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 last entry represents sum of the weights of vertices in the current solution. For vertices in T we assign 0. The rest of the update is done in same way as in proof of Theorem 4.1 in [9]. \square

A.2 Proof of Lemma 5

Proof: Let $S' \in \mathcal{N}_k^{en}(S)$ such that $c(S) < c(S')$. Clearly \tilde{S}'_1 and \tilde{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 \tilde{S}' is an independent set of \tilde{G} and $S' \in \mathcal{N}_k^{en}(\tilde{S})$ for k -LS-INDEPENDENT SET.

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

A.3 Proof of Lemma 6

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

$$\mathbf{tw}(\tilde{G}[N_{\tilde{G}}^r(v_i)]) \leq g(r).$$

We observe that

$$N_G^r(v_i) = N_{\tilde{G}[V_i]}^r(v_i) \cup N_{\tilde{G}[V_{3-i}]}^{r-1}(v_{3-i}).$$

We know that $N_{\tilde{G}[V_i]}^r(v_i)$ is isomorphic to $N_G^r(v)$ and hence $\mathbf{tw}(G[N_G^r(v)]) \leq h(r)$. Given a tree decomposition of width $h(r)$ for $G[N_G^r(v)]$ we obtain a tree decomposition of width $h(r)$ for $N_{\tilde{G}[V_i]}^r(v_i)$. Say this tree decomposition is given by $(\mathcal{T}, \{B_i\}_{i \in \mathcal{T}(V)})$. Now for a tree decomposition for $\tilde{G}[N_G^r(v_i)]$ we have the same tree \mathcal{T} but we go through every bag B_i and if it contains u_i such that $u_{3-i} \in N_{\tilde{G}[V_{3-i}]}^{r-1}(v_{3-i})$ then we add u_{3-i} to this bag. It is easy to check that this forms a valid tree decomposition. The upper bound on the width follows from the fact that any bag contains at most $2h(r) + 2$ vertices. \square

A.4 Proof of Theorem 9

Proof: For a subset S of V , (G, S, k) is a YES instance to k -LS-II-SUBSET if and only if $(G, V \setminus S, k)$ is a YES instance to the k -LS-II-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-II-SUBSET implies that II-SUBSET is FPT. Let $(G = (V, E), k)$ be an instance of II-SUBSET and \mathcal{A} be an FPT algorithm for k -LS-II-SUBSET. Since $\emptyset \in \Pi$, we have that \emptyset is a feasible solution for II-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 $f(k) \cdot |V|^{O(1)}$, for some function f , then II-SUBSET is solvable in time $kf(2k) \cdot |V|^{O(1)}$, making the II-SUBSET problem FPT and thus proving k -LS-II-SUBSET $W[1]$ -hard. A similar reduction can be obtained from the II-DELETION problem to the k -LS-II-DELETION problem. This concludes the proof. \square

B H -minor free graphs – Basic Definitions and a Proof Sketch

To make our presentation clear, we restrict ourselves to the k -LS-WEIGHTED 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 $F : W_1 \rightarrow W_2$ 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, \dots, 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, \dots, G_h of $G \setminus X$ such that

- $G \setminus X = G_0 \cup \dots \cup G_h$,
- G_0 is embeddable in Σ , we fix an embedding of G_0 ,

- G_1, \dots, G_h are pairwise disjoint,
- for $1 \leq \dots \leq h$, let $U_i := \{u_{i_1}, \dots, u_{i_{m_i}}\} = V(G_0) \cap V(G_i)$, 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 $V(G_0) \cap C_i = \{u_{i_1}, \dots, u_{i_{m_i}}\}$ and the points $u_{i_1}, \dots, u_{i_{m_i}}$ 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 Robertson and Seymour [32] which was made algorithmic by Demaine et al. in [11], states the following:

Theorem 11 (Robertson and Seymour [32], Demaine et al. [11]) *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 $n^{O(1)}$. 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 11. 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 [17]). 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 \widehat{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 $B_a \cup (\cup_s B_s)$, s being a descendant of a in T .

B.1 Proof Sketch of Theorem 7

Proof Sketch: Let (G, w, S, k) be an instance of k -LS-WEIGHTED 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 WEIGHTED DOMINATING SET. An instance (G, w, S, k) is an YES instance for k -LS-WEIGHTED 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 WEIGHTED DOMINATING SET. Now we solve k -LS-GENERALIZED WEIGHTED DOMINATING SET in time $f(4rk + r) \cdot |B(v, 4rk + r)|^{O(1)}$ for $G[B(v, 4rk + r)]$ for $v \in S$ which implies that we can solve k -LS-WEIGHTED DOMINATING SET for G in time $f(4rk + r) \cdot |G|^{O(1)}$.

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 11. 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 $n^{O(h)}$ 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