
**Karlsruhe Reports in Informatics 2011,18**
Edited by Karlsruhe Institute of Technology,
Faculty of Informatics
ISSN 2190-4782

# The Density Maximization Problem in Graphs

Mong-Jen Kao, Bastian Katz, Marcus Krug,
D.T. Lee, Ignaz Rutter, Dorothea Wagner

2011

KIT – University of the State of Baden-Wuerttemberg and National Research Center of the Helmholtz Association

# The Density Maximization Problem in Graphs ⋆

Mong-Jen Kao[2], Bastian Katz[1], Marcus Krug[1], D.T. Lee[2] ⋆⋆, Ignaz Rutter[1], Dorothea Wagner[1]

[1] Faculty of Informatics, Karlsruhe Institute of Technology (KIT), Germany
firstname.lastname@kit.edu
[2] Dep. of Computer Science and Information Engineering, National Taiwan University, Taiwan
d97021@csie.ntu.edu.tw, dtlee@csie.ntu.edu.tw

**Abstract.** We consider a framework for bi-objective network construction problems where one objective is to be maximized while the other is to be minimized. Given a *host* graph $G = (V, E)$ with edge weights $w_e \in \mathbb{Z}$ and edge lengths $\ell_e \in \mathbb{N}$ for $e \in E$ we define the density of a *pattern* subgraph $H = (V', E') \subseteq G$ as the ratio $\varrho(H) = \sum_{e \in E'} w_e / \sum_{e \in E'} \ell_e$. We consider the problem of computing a maximum density pattern $H$ with weight at least $W$ and and length at most $L$ in a host $G$. We call this problem the *bi-constrained density maximization problem*. In doing so, we compute a single Pareto-optimal solution with the best weight per cost ratio subject to additional constraints further narrowing down feasible solutions for the underlying bi-objective network construction problem. The problems expressible within this framework include maximization of return on investment for network construction problems in the presence of a limited budget and a target profit.
We consider this problem for different classes of hosts and patterns. We show that it is NP-hard even if the host has treewidth 2 and the pattern is a path. However, it can be solved in pseudo-polynomial linear time if the host has bounded treewidth and the pattern is a graph from a given minor-closed family of graphs. Finally, we present an FPTAS for a relaxation of the density maximization problem, in which we are allowed to violate the upper bound on the length at the cost of some penalty.

## 1 Introduction

Many realistic network construction problems are characterized by complex constraints and muliple, possibly conflicting, objectives. There are several ways to define optimality in the context of more than one objective, among them Pareto-optimality and aggregate optimality. While Pareto-optimality seems to capture the classical notion of optimality best, it is undesirable to confront users with a possibly exponential number of Pareto-optimal solutions. Therefore it is common to combine the objectives into a single new aggregate objective, which is then optimized as a single-criterion objective. Typical aggregate functions include weighted sum or weighted minimum and maximum. These weighted aggregate functions, however, must be guided in that the decision maker has to supply a set of suitable weights at the risk of arbitrariness.

We consider a framework for bi-objective network construction problems, motivated from economics, where one objective must be maximized (profit) while the other must be minimized (cost). Additionally, we assume that we are given an upper bound on the objective to be minimized (limited budget) and a lower bound on the objective to be maximized (target profit). Bi-objective optimization functions of this sort can be aggregated by the ratio of the two optimization goals featuring two main advantages over other aggregate functions. First, we do not need to supply any weights—if we did, it would not alter our notion of optimality. Second, any optimal solution with respect to the ratio is Pareto-optimal. In economics, this ratio is termed return on investment.

Our framework is defined as folllows. Let $G = (V, E)$ be a graph, which we will refer to as the *host*. Given $G$, we write $V(G) = V$ and $E(G) = E$. Throughout the paper we write $n := |V(G)|$ and $m := |E(G)|$.We assume that we are given a weight function $w : E \to \mathbb{Z}$ and a length function $\ell : E \to \mathbb{N}$ on the edges, respectively. As a shorthand we will write $w_e := w(e)$ and $\ell_e := \ell(e)$ and for a subgraph $H \subseteq G$ we define $w(H) := \sum_{e \in E(H)} w_e$ and $\ell(H) := \sum_{e \in E(H)} \ell_e$. We refer to a subgraph of $G$ as a *pattern*. Given $W \in \mathbb{Z}$ and $L \in \mathbb{N}$ a pattern $H$ is called *$W$-viable* if $w(H) \geq W$

⋆ Supported by NSC-DFG Projects NSC98-2221-E-001-007-MY3 and WA 654/18.
⋆⋆ Also supported by the Institute of Information Science, Academia Sinica, Taiwan.

and it is called $(W, L)$-*viable* if it is $W$-viable and $\ell(H) \leq L$. Our goal is to find a $(W, L)$-viable pattern $H$ maximizing $w(H)$ and minimizing $\ell(H)$, which we formalize by maximizing the ratio $\varrho(H) = w(H)/\ell(H)$, defined as the *density of* $H$. Given a tuple $(G, w, \ell, W, L)$ the Bi-constrained Maximum Density Subgraph (BMDS) problem asks for a *connected* $(W, L)$-viable pattern $H \subseteq G$ with maximum density.

In realistic applications, it may be desirable to be able to violate the hard limitations of our framework. For instance, it may be possible to exceed the budget by loaning additional money at the cost of some interest. We model this by introducing the $L$-*deviation* of $H$, defined as $\Delta(H) := \max\{0, \ell(H) - L\}$, and the *penalized density*, defined as $\widetilde{\varrho}(H) := w(H)/(\ell(H) + c \cdot \Delta(H))$, where $c$ is some non-negative constant. Given a tuple $(G, w, \ell, W, L)$ the Relaxed Maximum Density Subgraph (RMDS) problem asks for a *connected* $W$-viable pattern $H \subseteq G$ with maximum *penalized* density. We will consider these problems for different classes of hosts and patterns.

*Overview and Related Work* An overview of recent developments in multi-objective optimization is given in [3]. Bálint [1] proves inapproximability for bi-objective network optimization problems, where the task is to minimize the diameter of a spanning subgraph with respect to a given length on the edges, subject to a limited budget on the total cost of the edges. Marathe et al. [16] study bi-objective network design problems with two minimization objectives. Given a limited budget on the first, they provide a PTAS for minimizing the second objective among a set of feasible graphs. The considered objectives include total edge weight, diameter and maximum degree.

The study of dense segments from in bi-weighted sequences arises from the investigation of non-uniformity of nucleotide composition with genomic sequences [9,15] and has received considerable attention in bio-informatics. Here, we are given a sequence of pairs $(a_i, b_i)$ and we wish to find a subsequence $I$ with length bounded by $A \leq \sum_{i \in I} b_i \leq B$ that maximizes the density $\sum_{i \in I} a_i / \sum_{i \in I} b_i$. For uniform lengths, Lin et al. [13] give an $\mathcal{O}(n \log A)$ algorithm, which is improved to $\mathcal{O}(n)$ by Goldwasser et al. [6]. A linear time algorithm for the non-uniform case is given by Chung and Lu [4]. Lee et al. [12] show how to select a subsequence whose density is closest to a given density $\delta$ in $\mathcal{O}(n \log^2 n)$ time. Without the upper bound on the length $B$ an optimal $\mathcal{O}(n \log n)$-time algorithm is given.

Subsequently, this problem has been generalized to graphs. Hsieh et al. [8,7] show that a maximum density path in a tree subject to lower and upper length bounds can be computed in time $\mathcal{O}(Bn)$ and that it is NP-hard to find a maximum density subtree in a tree, for which they also presented an $\mathcal{O}(B^2 n)$ time algorithm. Wu et al. [21,20] improve on this by presenting an optimal algorithm for computing a maximum density path in a tree in time $\mathcal{O}(n \log n)$ in the presence of both a lower and upper length bounds. They also give an $\mathcal{O}(n \log^2 n)$ algorithm for finding a *heaviest path* in a tree in the presence of length constraints [21], which is improved to $\mathcal{O}(n \log n)$ by Liu and Chao [14] .

*Our Contribution* In Section 2 we prove that the BMDS problem is NP-hard, even if the host has treewidth 2 and the pattern is a path. Then we show how to compute a maximum density path in a tree in Section 2.1 and extend this result to graphs that can be turned into a tree by removing $k$ edges, thus, showing that the problem is FPT with respect to $k$. In Section 2.2 we show how to solve the BMDS and the RMDS problems in pseudo-polynomial linear time if the host has bounded treewidth and the pattern must be contained in a given minor-closed family of graphs. Finally, we present a general FPTAS that can be applied to all RMDS problems that admit algorithms whose running time is pseudo-polynomial in the length in Section 3. We show that it can be used to approximate the maximum penalized density if the host has bounded tree-width and the pattern belongs to a minor-closed family of graphs.

## 2 Bi-constrained Maximum Density Subgraph

In this section we consider the Bi-constrained Maximum Density Subgraph (BMDS) problem. Given an instance $I = (G, w, \ell, W, L)$ we wish to find a connected $(W, L)$-viable subgraph of $G$ with maximum density. Since the BMDS problem can be solved in time $\mathcal{O}(n^2)$ when the host is a tree and the pattern is a path by enumerating all possible paths, it is natural to ask if the BMDS problem can be solved efficiently on more general hosts and patterns. However, we show that it is

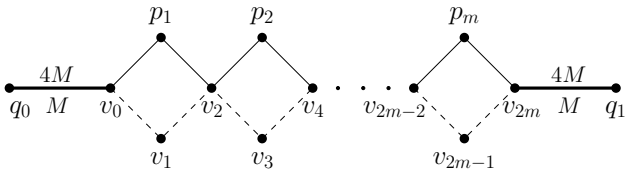


**Fig. 1.** Graph used in the reduction from PARTITION. Bold edges have density 4, all other edges have density 1. Dashed edges have weight and length 1, solid non-bold edges incident to $p_i$ have weight and length $c_i + 1$.

NP-hard to find a maximum density path, even if the host is only slightly more complicated than a tree.

**Theorem 1.** BMDS *is* NP*-hard, even if the host simply connected outerplanar graph with treewidth two, the pattern is a path and we drop the upper bound on the length of the pattern.*

*Proof.* The proof is by reduction from PARTITION. Assume we are given an instance of PARTITION, i.e., a set of positive integers $C = \{c_1, c_2, \ldots, c_m\}$ with $M = \sum_{i=1}^{m} c_i$ and we ask whether there is a subset $I' \subseteq \{1, \ldots, m\}$ with $\sum_{c_i \in I'} c_i = M/2$. We transform this into an instance of BMDS as follows. The transformation is illustrated in Figure 1. First, we create a path $v_0, v_1, \ldots, v_{2m}$ with $w_e = \ell_e = 1$ for each edge $e$ on this path. Besides, we create additional $m$ vertices, $p_1, p_2, \ldots, p_m$, and connect $p_i$ to both $v_{2i-2}$ and $v_{2i}$ with $w_e = \ell_e = c_i + 1$ for $e \in \{p_i v_{2i-2}, p_i v_{2i}\}$. Then we create additional vertices $q_0$ and $q_1$, which we connect to $v_0$ and $v_{2m}$, respectively, such that $w_{q_0 v_0} = w_{q_1 v_{2m}} = 4M$ and $\ell_{q_0 v_0} = \ell_{q_1 v_{2m}} = M$. Furthermore, we set $W = 9M + 2m$. Since the graph is outerplanar its treewidth is bounded by 2.

We claim that there is a path with length at least $W$ and density at least $d := W/(3M + 2m)$ iff the corresponding instance of PARTITION can be solved. Clearly, any partition can be transformed into a path with density $d$ and length $W$. Conversely, assume that $P$ is such a path. Since the weight of the path must be at least $9M$ it must end at $q_0$ and $q_1$, respectively. Let $S$ be the set of indices such that $p_i$ is on the path iff $i \in S$. Then the density of this path can be expressed as $(8M + 2m + 2\sum_{i \in S} c_i)/(2M + 2m + 2\sum_{i \in S} c_i)$, which is strictly decreasing as $2\sum_{i \in S} c_i$ is increasing. Hence we have $2\sum_{i \in S} c_i \leq M$. On the other hand the weight of the path must be at least $W$, which implies $2\sum_{i \in S} c_i \geq M$. Thus, the path induces a valid partition $I' := \{c_i \mid i \in S\}$. □

When both the lower-bounded and upper-bounded constraints are imposed the problem becomes much harder. By setting $L = 3M + 2m$ we can show that it is NP-hard to even compute any feasible solution if we impose both the lower bound on the weight and the upper bound on the length of the pattern. Hence, the problem is not likely to be approximable in polynomial time.

### 2.1 Density Maximization for Trees and Almost-Trees

In the previous section, we have shown that it is NP-hard to compute a maximum density path even if the host graph has treewidth 2. Hence the problem is unlikely to be FPT with respect to the parameter treewidth. In this section, however, we show that the problem of computing a maximum density path is FPT with respect to the number of edges $k$ that must be deleted from a graph in order to obtain a tree. The treewidth of such a graph is bounded by $k + 1$. In this section we consider instances of the density maximization problem where the pattern is a path. First, we show how to compute a maximum density path when the host is a tree. The problem can trivially be solved in $\mathcal{O}(n^2)$ time by enumerating all possible paths, but we show how to solve it in $\mathcal{O}(n \log^3 n)$ time. Our basic approach is similar to one described by Wu [20] and Lau et al. [11] with respect to decomposing the problem into smaller sub-problems. However, we use completely different techniques for the sub-problems to obtain our results, since the results by Wu and Lau et al. are not applicable in our setting. Second, we use this result to show that finding a maximum density path in a general graph is FPT with respect to the number of edges we have to remove in order to obtain a tree.

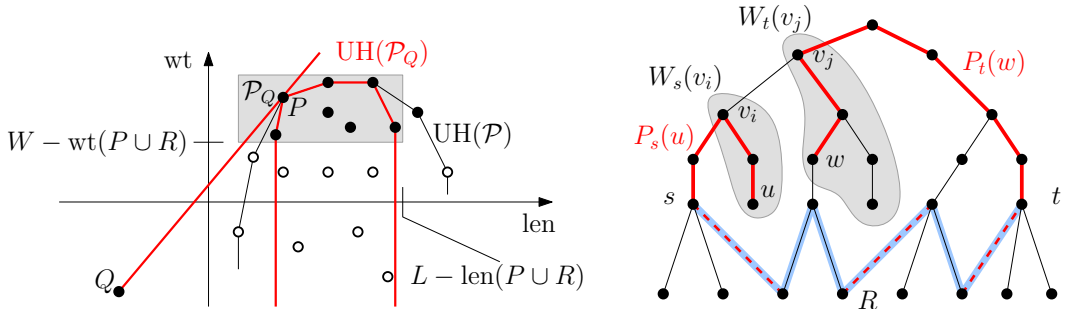


**Fig. 2.** Tangent query to find the best candidate for $Q$ (left) and combination of two paths extending $Q$ (right).

Throughout the section we will be using the key idea that the combined density of two subpaths $P$ and $Q$ is equal to the slope between two points $u_P = (\ell(P), w(P))$ and $-u_Q = (-\ell(Q), -w(Q))$ in the Euclidean plane. This path is feasible iff $w(P) \geq W - w(Q)$ and $\ell(P) \leq L - \ell(Q)$. For a given query path $Q$ this slope is maximized on the convex hull of the set of points $\mathcal{P}_Q$ representing the candidate subpaths for $Q$ in the range $(-\infty, L - \ell(Q)] \times [W - w(Q), \infty)$. Since we wish to maximize the density it suffices to perform tangent queries to the upper chain of the convex hull of $\mathcal{P}_Q$, denoted by UH($\mathcal{P}_Q$), which is more efficient than trying all possible combinations (see Figure 2).

We use a dynamic data structure for the maintenance of the upper chain of the convex hull of a set of points $P$ [17] allowing point insertions in time $\mathcal{O}(\log^2 n)$. It maintains the upper chain of the convex hull by a dynamically maintained ordered binary tree. Each leaf of this tree corresponds to a point in the plane and each inner node $v$ corresponds to the segment of the upper hull of the set of points $P_v$ that does not contribute to the upper hull of the set of points in the subtree of the father of $v$. Each inner node additionally stores the number of points on its upper hull (head or tail) that it inherits from its father. The segments of the upper hulls are represented by concatenable queues which allow insertion, deletion, concatenation and split in $\mathcal{O}(\log n)$ time. Lemma 1 shows how to compute UH($\mathcal{P}_Q$) from a given set of candidate paths $\mathcal{P}$ in time $\mathcal{O}(\log^2 n)$ given the dynamic data structure.

**Lemma 1.** *Given a point $Q$, a set of points $\mathcal{P}$ and a dynamic data structure for the maintenance of* UH($\mathcal{P}$) *as described in [17] we can compute* UH($\mathcal{P}_Q$) *in time $\mathcal{O}(\log^2 n)$.*

*Proof.* If $\mathcal{P}_Q$ is empty, there is nothing to do. Otherwise, let $p_{\min}$ be the leftmost point in $\mathcal{P}_Q$ and let $p_{\max}$ be the rightmost point in $\mathcal{P}_Q$. Let $x_{\min}$ and $x_{\max}$ be the respective $x$-coordinates of these points. Let $\mathcal{P}' := \{(x, y) \in \mathcal{P} \mid x_{\min} \leq x \leq x_{\max}\}$. First, we prove that UH($\mathcal{P}_Q$) $\subseteq$ UH($\mathcal{P}'$). Clearly, $\mathcal{P}' = \mathcal{P}_Q \cup \mathcal{P}''$ where $\mathcal{P}''$ contains all the points $(x, y) \in \mathcal{P}$ with $x_{\min} \leq x \leq x_{\max}$ and $y < W$. Thus, all points in $\mathcal{P}''$ are either in the interior of the convex hull of $\mathcal{P}_Q$ or on a vertical line through $x_{\min}$ or $x_{\max}$, respectively. Hence, the claim holds and we can reduce the problem of computing UH($\mathcal{P}_Q$) to computing the upper hull of a set of points $\mathcal{P}'$ in a vertical strip of the plane, which is supported by the dynamic data structure. Let $\mathcal{T}$ denote the tree used by the dynamic data structure for the maintenance of the upper hull. In order to compute UH($\mathcal{P}'$) we traverse the paths from the root of $\mathcal{T}$ to $p_{\min}$ and $p_{\max}$, respectively, in parallel. In each step we reconstruct the upper hull using the concatenation and split operation of the concatenable queues stored in the nodes of the tree. We split off branches of the tree that are to the left of the path from the root to $p_{\min}$ and to the right of the path from the root to $p_{\max}$. These branches contain only points whose $x$-coordinates are either greater than $x_{\max}$ or smaller than $x_{\min}$. Since $\mathcal{T}$ is balanced we have reconstructed UH($\mathcal{P}'$) after at most $\log n$ steps using time $\mathcal{O}(\log n)$ per step and $\mathcal{O}(\log^2 n)$ time in total. Clearly, we can reconstruct the original data structure with the same complexity. □

**Theorem 2.** *Given an instance $(T, w, \ell, W, L)$ of the* BMDS *problem, where $T = (V, E)$ is a tree, we can compute a $(W, L)$-viable maximum density path in $\mathcal{O}(n \log^3 n)$ time.*

*Proof.* Without loss of generality we may assume that $T$ is a binary tree. Otherwise we can make it binary by adding dummy edges with weight and length 0 in linear time. A centroid of a binary

tree is a vertex whose removal disconnects $T$ into at most 3 subtrees with at most half of the vertices of the original tree in each of the subtrees. We root $T$ in one of its centroids $r$. Let $v_1, v_2$ be two children of $r$ and let $R$ be the path between $v_1$ and $v_2$ via $r$. Then we can compute the maximum density path including $R$ using tangent queries in time $\mathcal{O}(n \log^2 n)$ as follows: First, we compute the set $\mathcal{P}_1$ of paths starting in $v_1$. Each of those paths $P \in \mathcal{P}_1$ is mapped to a point $u_P := (\ell(P \cup R), w(P \cup R))$ in the plane and inserted into the dynamic datastructure for the maintenance of the upper hull. This can be done in $\mathcal{O}(n \log^2 n)$. Then we compute the set of paths $\mathcal{P}_2$ starting in $v_2$. For each of these paths $Q \in \mathcal{P}_2$ we want to compute the best path $P \in \mathcal{P}_1$, i.e., a path $P$ such that the concatenation of $Q$ and $P \cup R$ has maximum density.

To this end, we map each $Q \in \mathcal{P}_2$ to a point $-u_Q := (-\ell(Q), -w(Q))$. Since we have bounds on both the weight and the length of a feasible solution, not all paths in $\mathcal{P}_1$ will be feasible partners for a given $Q \in \mathcal{P}_2$. We require that the length of $P \in \mathcal{P}_1$ is bounded by $w(P) \geq W - w(Q) - w(R)$ and $\ell(P) \leq L - \ell(Q) - \ell(R)$. Using Lemma 1 we can compute the maximum density partner for $Q \in \mathcal{P}_2$ in time $\mathcal{O}(\log^2 n)$. Then we can compute the maximum density path $P^*$ through $r$ in time $\mathcal{O}(n \log^2 n)$. We do this for all combinations of children of $r$ and store the path of maximum density. Next, we recursively compute the best path through each of the children of $r$ in the subtrees rooted in the children. Let $\hat{P}$ be the maximum density path over all the paths computed this way. Then the maximum density path in the tree rooted in $r$ is the maximum density path over $P^*$ and $\hat{P}$. The recurrence relation for the computation is given by $T(n) = \sum_{i=1}^{3} T(n_i) + \mathcal{O}(n \log^2 n)$, where $n_i$ is the number of vertices in the tree rooted in $v_i$. Hence, the running time of this approach is $\mathcal{O}(n \log^3 n)$. □

Next, we show that we can obtain a similar result if the host is a graph that can be turned into a tree by deleting a fixed number of $k$ edges. Roughly, the key idea consists of enumerating all possible subsets of the $k$ edges and computing, for each of those subsets, the maximum density path containing all theses edges. The following lemma can be used to enumerate those paths efficiently.

**Lemma 2.** *Given a graph $G = (V, E \cup F)$ such that $T = (V, E)$ is a tree and $F \cap E = \emptyset$ as well as $F' \subseteq F$ and vertices $s, t \in V$ incident to the edges in $F'$, then there is at most one $s$-$t$-path in $G$ containing all edges in $F'$. We can compute such a path or prove that no path exists in linear time.*

*Proof.* We prove the claim by contradiction. Suppose that there are two different paths $P_1$ and $P_2$ both containing all edges in $F'$ and ending with $s$ and $t$, respectively. Since the paths are different and both contain all edges in $F'$ the symmetric difference $\Delta$ of $E(P_1)$ and $E(P_2)$ is non-empty and contained in $E$. Since both paths end at $s$ and $t$, all vertices of $\Delta$ have even degree. Hence, $\Delta$ contains a cycle contradicting the fact that $\Delta$ is a subgraph of $T$.

We proceed by showing that the uniquely determined feasible path can be computed in linear time, if it exists. We root $T$ in some vertex $r \in V(T)$. By $T_v$ we denote the tree rooted in $v \in V(T)$. For a given $F' \subseteq F$ we call $v \in V(T) \setminus \{s, t\}$ a *loose end* if it is incident to exactly one edge in $F'$. To compute $P$ we traverse $T$ in a bottom-up fashion constructing $P$ by iteratively matching loose ends. For each vertex $v$ we store a reference to the unmatched loose end, if it exists. Let $v$ be a vertex with children $w_1, \ldots, w_\ell$. Clearly, there can only be a valid path if at most two children, say, $w_1$ and $w_2$, contain an unmatched loose end in their subtrees. Otherwise there is no feasible path. If none of the children contains an unmatched loose end, then there is nothing to do. If exactly one child contains an unmatched loose end in its subtree, we store a reference to this vertex in $v$. If exactly two children of $v$ contain unmatched loose ends $\ell_1$ and $\ell_2$ in their subtrees, then we update the path by matching these loose ends and adding the unique path in $T$ that connects $\ell_1$ and $\ell_2$. We accept the resulting graph if it is a path, which can be checked in linear time. □

**Theorem 3.** *Given an instance $(G, w, \ell, W, L)$ of the* BMDS *problem such that $G$ is a tree with $k$ additional edges, we can compute a maximum density $(W, L)$-viable path in time $\mathcal{O}(2^k k^2 n \log^2 n + n \log^3 n)$.*

*Proof.* Given a tree with $k$ additional edges $G = (V, E)$, we first compute an arbitrary spanning tree $T = (V, E')$ of $G$. This leaves exactly $k$ edges, denoted by $F := E \setminus E'$, which may or may not be used by the optimal path. For each $F' \subseteq F$ we compute the maximum density path containing all edges in $F'$ and we return the maximum density path over all $F' \subseteq F$. At first we compute

the maximum density path in $T$ in $\mathcal{O}(n \log^3 n)$ time using Theorem 2. Any path containing some non-empty subset of edges $F' \subseteq F$ can be decomposed into three subpaths $P, Q$ and $R$ such that $R$ starts and ends with edges in $F'$ and contains all edges in $F'$. By Lemma 2 the possible paths $R$ are uniquely determined by choosing a set $F' \subseteq F$ as well as two vertices incident to $F'$ and $R$ can be computed in linear time from this information.

Hence, we iterate over all possible $F' \subseteq F$ and all $s, t \in V$ incident to $F'$. In each of the $2^k k^2$ iterations, we first compute both weight and length of $R$ in linear time, resulting in $\mathcal{O}(2^k k^2 n)$ time. Then we find paths $P$ and $Q$ starting at $s$ and $t$, respectively, such that the density of the concatenation of $P$, $R$ and $Q$ has maximum density among all paths including $R$ in time $\mathcal{O}(\log^2 n)$. For the remainder of the proof we show how this can be accomplished and we thus assume that $R$, $s$ and $t$ are fixed. Our approach is similar to the proof of Theorem 2. However, we must take care of the disjointness of the paths.

Let $s = v_0, \ldots, v_\ell = t$ be the sequence of vertices on the path from $s$ to $t$ in $T$. For each of these vertices $v_i \neq s, t$, we define $W_s(v)$ as the set of vertices in $T_{v_i}$ that are reachable from $s$ in $T$ without crossing the path $R$. Each of the vertices $w \in W_s(v_i)$ defines a path $P_s(w)$. Analogously, we define the set of vertices $W_t(v_i)$ in $T_{v_i}$ that are reachable from $t$ in $T$ without crossing $R$. Each of those vertices $w \in W_t(v_i)$ defines a path $P_t(w)$ from $t$ to $w$. Two paths in $P_s(v_i)$ and $P_t(v_j)$, respectively, are disjoint, whenever $v_i$ is encountered before $v_j$ on the path from $s$ to $t$, i.e., if $i < j$, otherwise they will have at least one vertex in common. See Figure 2 for an illustration.

Now we describe how we insert the paths into the dynamic data structure for the maintenance of the upper hull. As pointed out, paths may not be disjoint, hence, we must insert the paths in a specific order. First, we insert all paths starting in $s$ that do not include any vertex on the path from $s$ to $t$. Then, for each $i = 1, \ldots, \ell - 1$ we insert all paths $P_s(w)$ for all $w \in W_s(v_i)$. After inserting the paths for a specific $i < \ell - 2$ we make tangent queries for all paths $P_t(w)$ for $w \in W_t(v_{i+1})$. Note that at that point, we have included all paths starting in $s$ except those that would not be disjoint to the paths in $P_s(w)$ for $w \in W_t(v_{i+1})$. After we have inserted all paths $P_s(w)$ for all $w \in W_s(v_{\ell-1})$ we have inserted all paths starting in $s$, which do not cross $R$. Then we make tangent queries for all paths starting in $t$ that do not use any vertex on the path from $s$ to $t$.

In order to compute the best path for $F = \emptyset$ we proposed an algorithm with running time $\mathcal{O}(n \log^3 n)$. For each specific non-empty choice of $F' \subseteq F$ and vertices $s$ and $t$ incident to $F'$ we thus insert at most $n$ points into the data structure with a total running time of $\mathcal{O}(n \log^2 n)$ and we perform at most $n$ tangent queries, each with a running time of at most $\mathcal{O}(\log^2 n)$. Hence, the overall running time is $\mathcal{O}(2^k k^2 n \log^2 n + n \log^3 n)$. □

*Remark 1.* The Relaxed Maximum Density Subgraph problem can be solved within the same asymptotic bounds by similar means if the pattern is a path and the host is a tree or a tree with $k$ additional edges, respectively.

### 2.2 Density Maximization in Graphs with Bounded Treewidth

In this section we show that a large class of problems can be solved in pseudo-polynomial FPT time when parameterized by the treewidth $k$ of the host, i.e., in time $\mathcal{O}(f(k)p(L, n))$ where $f$ is a function depending only on $k$ and $p$ is a polynomial depending on the maximum length $L$ of any feasible pattern and the number of vertices $n$ of the graph. In the light of the results on the hardness of the problem this seems to be the best we can hope for. Given a graph $G$ with treewidth $k$ and a finite set of graphs $\mathcal{F}$, we wish to find some *connected* $(W, L)$-viable pattern $H$ with maximum density that does not contain any graph in $\mathcal{F}$ as a minor. Such a graph is called $\mathcal{F}$-minor-free. This includes trees, (outer-)planar graphs as well as graphs from various other minor-closed families of graphs. We give an algorithm for the general case but note that the running time can be improved by considering special classes of graphs. We assume that we are given a tree decomposition of the host as an input; otherwise it can be computed in FPT time [2,10]. We note that the size of the forbidden obstructions is small for many interesting examples, such as trees and (outer-)planar graphs whose sets of forbidden minors include graphs with $\leq 6$ vertices.

Our algorithm is based on dynamic programming on the tree decomposition of the graph and is inspired by Eppstein's work on subgraph isomorphism in planar graphs [5]. Based on Eppstein's

idea of enumerating partial isomorphisms for the bags of the tree decomposition, we enumerate partial minors of the graphs induced by the bags. In the following we present the key ideas in more detail. Let $G = (V, E)$ be a graph. A *tree decomposition* of $G$ is a pair $(\mathcal{X}, \mathcal{T})$ where $\mathcal{X} = \{X_i \mid i \in I\}$ is a collection of subsets of $V$ which are called *bags* and $\mathcal{T} = (I, E_\mathcal{T})$ is a tree with the following properties: (i) $\bigcup_{i \in I} X_i = V$. (ii) For all $e \in E$ there is an $i \in I$ such that $e \subseteq X_i$. (iii) For all $v \in V$, $\mathcal{X}_v = \{i \in I \mid v \in X_i\}$ induces a connected subtree of $\mathcal{T}$. We will refer to the elements in $I$ as nodes—as opposed to vertices in the original graph. The *treewidth* of a tree decomposition equals $\max_{i \in I} |X_i| - 1$. The treewidth of a graph $G = (V, E)$ is equal to the minimum treewidth of a tree decomposition of $G$.

**Theorem 4.** *Let $(G, w, \ell, W, L)$ be an instance of the* BMDS *problem such that $G$ has treewidth at most $k$ and let $\mathcal{F}$ be a non-empty finite set of graphs. Then the maximum density $\mathcal{F}$-minor-free $(W, L)$-viable pattern can be computed in time $2^{\mathcal{O}(k^2 + k \log N + N)} |\mathcal{F}| Ln$ where $N = \max_{F \in \mathcal{F}} |V(F)|$.*

*Proof.* We describe the algorithm for the case that $\mathcal{F}$ consists of only one forbidden obstruction $F$. The extension to a larger family of forbidden obstructions is straightforward. Our algorithm is by dynamic programming on the tree decomposition of the graph. Robertson and Seymour have proven the existence of an $\mathcal{O}(n^3)$ graph minor test for any fixed minor $F$ [18]. However, the proof is non-constructive and involves huge constants. Therefore, we describe an explicit algorithm for the graph minor test which relies on the enumeration of subgraphs instead. We note, however, that we need some explicit representation of the minor mappings for the dynamic programming anyway, thus, this does not change the asymptotic complexity of our approach.

For the proof we assume that we are given a nice tree decomposition. A tree decomposition is called *nice* if $\mathcal{T}$ is a rooted binary tree where each node is of one of the following types: A *leaf node* $X$ contains only one vertex. An *introduce node* $X$ has only one child $Y$ such that $X = Y \cup \{v\}$ for some $v \in V$. We say $X$ *introduces* $v$. A *forget node* $X$ has only one child $Y$ such that $X = Y \setminus \{v\}$ for some $v \in V$. We say $X$ *forgets* $v$. Finally, a *join node* $X$ has two children $Y_1$ and $Y_2$ such that $X = Y_1 = Y_2$. Given a graph with treewidth $k$ we can always find a nice tree decomposition with $\mathcal{O}(n)$ nodes in linear time [10].

Throughout the proof we assume that $G = (V, E)$ is a graph with treewidth at most $k$ and we let $(\mathcal{X}, \mathcal{T})$ be a nice tree decomposition of $G$ with treewidth at most $k$, rooted in a node $r \in I$. For $i \in I$ we denote the graph induced by the union of the bags of all descendants of $i$ (including $i$) by $G_i$. Using standard notation, we denote the graph induced by $X_i$ by $G[X_i]$. Let $\mathcal{C} := \{V_1, \ldots, V_q\}$ be a disjoint collection of connected subsets of $V$. We denote the graph obtained by contracting the vertices in each of the sets $V_i$ into a single vertex by $G/\mathcal{C}$ and we refer to the sets $V_i$ as *branch sets* and to $\mathcal{C}$ as a *contraction set*. Then $F$ is a minor of $G$ iff there is a subgraph $H$ and a contraction set $\mathcal{C}$ such that $H/\mathcal{C}$ is isomorphic to $F$.

Let $\mathcal{C}$ be a contraction set and let $H$ be some subgraph in $G[X_i]$ for some $i \in I$. A *partial minor embedding of $F$ into $H$ with respect to $\mathcal{C}$* is a mapping $\varphi : V(F) \to V(H/\mathcal{C}) \cup \{\bot, \top\}$ such that $uv \in E(F) \Rightarrow \varphi(u)\varphi(v) \in E(H/\mathcal{C})$ for all $uv \in E(F)$ with $u, v \notin \varphi^{-1}(\bot) \cup \varphi^{-1}(\top)$, hence, $\varphi$ maps a subgraph of $F$ to a minor of $H$. The image $\bot$ represents vertices in $G_i - X_i$ and the image $\top$ represents vertices in $G$ which have not been considered yet, i.e., vertices in $G - G_i$. A partial minor embedding $\varphi$ is called *proper* if and only if $\varphi^{-1}(\top) \neq \emptyset$. Otherwise it represents a minor embedding of $F$ into some subgraph of $G_i$, and hence, $H$ may disregarded as a partial solution.

For the algorithm we identify the vertices of $F$ with the numbers $1, \ldots, |V(F)|$. The images of these vertices under $\varphi$ which are not contained in $\varphi^{-1}(\bot) \cup \varphi^{-1}(\top)$ correspond to branch-sets of $H$, i.e., a partition of the vertices of $H$. By considering all $|V(F) + 1|^{k+1}$ labelings of the vertex set of $H$ where each vertex is labeled with some number in $0, \ldots, V(F)$ we obtain a partition of the vertices induced by the labeling. Such a partition is valid only if each set of vertices forms a connected set. Further it defines an implicit mapping $f$ of a subset of the vertices of $F$ to the partitions induced by the labeling. Vertices labeled 0 are considered not be images under $f$. We further encode for each vertex in $F$ which does not have an image under $f$ whether it is mapped to $\bot$ or $\top$. A mapping to $\bot$ means that the vertex can be mapped to some branch-set in the subgraph induced by the descendants of node $i$ whereas a mapping to $\top$ means that we will try to map the vertex to some branch-set we have not encountered, yet. We can check in $\mathcal{O}(k^2)$ time if such an encoding represents a valid partial minor embedding of $F$ into $H$. We check connectedness of the partitions in time $\mathcal{O}(k)$. Further we check if each edge in $F$ is represented by some edge between

the corresponding branch-sets in $H$. This can be done in time $\mathcal{O}(k^2)$ by iterating over all pairs vertices in $H$, and hence, over all pairs of labels and checking corresponding edges in both $F$ and $H$. Using these conventions, we can encode a partial minor embedding $\varphi$. It is not hard to see that there are at most $|V(F)|^{k+1}2^{|V(F)|}$ many partial minor embeddings using this kind of encoding.

By $W(i, H, \Phi, \ell)$ we denote the maximum weight of a subgraph $G'$ of $G_i$ with length $\ell$, such that $G'[X_i] = H \subseteq G[X_i]$ and $\Phi$ represents all partial minor embeddings $\varphi$ of $F$ into $G'$. We call the quadruple $(i, H, \Phi, \ell)$ an *interface for* $i$. An interface is called proper if and only if $\varphi(\top) \neq \emptyset$ for all $\varphi \in \Phi$.

By $G_i^*$ we denote the graph obtained by adding new vertices $\top$ and $\bot$ to $G[X_i]$ which are each connected to all vertices in $X_i$. Both weight and length of the additional edges is equal to zero. We then consider connected subgraphs in $G_i^*$. Note that any connected subgraph $G'$ can be mapped to a connected subgraph in $G_i^*$.

The solution we are looking for will be the maximum over all interfaces $(r, H, \Phi, \ell)$ where $r$ is the root of the tree decomposition, such that $H$ is connected and does not contain $\top$, $\Phi$ is proper and $\ell$ is at most $L$. If the maximum weight is at least $W$, then we return this weight, otherwise there is no feasible solution. We now describe how $W(i, H, \Phi, \ell)$ can be computed in $\mathcal{T}$ in a bottom-up fashion by dynamic programming starting at the leaves of $\mathcal{T}$.

*Leaf node $i$ with $X_i = \{v\}$:* For each $H$ in $G_i^*$ which does not include $\bot$ we compute the set $\Phi$ of partial minor embeddings of $F$ into $H$ and we set $W(i, H, \Phi, 0) = 0$, since both the weight and length of any subgraph of $G_i^*$ are equal to 0 by construction. Note that any vertex in $F$ which is not mapped to $v$ must be mapped to $\top$. Hence, the time complexity is asymptotically bounded by

$$\underbrace{\mathcal{O}(1)}_{\text{subgraphs } H} \cdot \underbrace{|V(F)|^2 \cdot |V(F)|}_{|\Phi|} \cdot \underbrace{\mathcal{O}(1)}_{\text{check mapping}} \cdot L\,.$$

*Introduce node $i$ introducing $v$:* Let $j$ be the only child of $i$ and let $(j, H', \Phi', \ell')$ be an interface for $j$ such that $W_j := W(j, H', \Phi', \ell')$. We consider all connected subgraphs $H$ of $G_i^*$ which can be obtained from $H'$ by adding $v$ and some set of edges $E^+$ incident to both $v$ and some set of vertices in $H'$. For each fixed $H$ obtained this way we further consider the set $\Phi$ of all partial minor embeddings $\varphi$ of $F$ into $H$ which can be obtained from some $\varphi' \in \Phi'$ by choosing some vertex in $\varphi'^{-1}(\top)$ to be mapped to $v$ by $\phi$. If all partial minor embeddings $\varphi$ constructed this way are proper and $\ell := \ell' + \ell(E^+) \leq L$ the interface $(i, H, \Phi, \ell)$ is proper and we compute $W(i, H, \Phi, \ell) = W_j + w(E^+)$ and we set $W(i, H', \Phi', \ell') = W_j$. Hence, the complexity for an introduce node is asymptotically bounded by

$$\underbrace{2^{\binom{k}{2}}}_{\text{subgraphs } H} \cdot \underbrace{|V(F)|^{k+1} \cdot 2^{|V(F)|}}_{|\Phi'|} \cdot \underbrace{2^k}_{|E^+|} \cdot \underbrace{|V(F)|}_{\text{new mappings to } v} \cdot \underbrace{k^2}_{\text{check mapping}} \cdot L$$

*Forget node $i$ forgetting $v$:* Let $j$ be the only child of $i$ and let $(j, H', \Phi', \ell')$ be an interface for $j$ such that $W_j := W(j, H', \Phi', \ell')$. If $H'$ does not contain $v$, then there is nothing to do and we simply set $W(i, H', \Phi', \ell') = W_j$. Otherwise, we consider the set $\Phi$ of all mappings $\varphi$ which can be obtained from mappings $\varphi'$ by removing $v$ from its partition in the branch set. If $v$ is the only vertex in its partition, then the corresponding vertex in $F$ must additionally be mapped to $\bot$. We set $W(i, H, \Phi, \ell') = W_j$ where $H$ is obtained from $H'$ by removing $v$ and mapping all edges from $v$ to any vertex in $X_i$ by a corresponding edge with the end-vertex corresponding to $v$ in $\bot$. The resulting complexity of a forget node is asymptotically bounded by

$$\underbrace{2^{\binom{k}{2}}}_{\text{subgraphs } H} \cdot \underbrace{|V(F)|^{k+1} \cdot 2^{|V(F)|}}_{|\Phi'|} \cdot \underbrace{\mathcal{O}(1)}_{\text{remapping}} \cdot L$$

*Join node $i$ joining $j_1$ and $j_2$:* Let $j_1$ and $j_2$ be the two children of $i$ and let $(j_1, H, \Phi_1, \ell_1)$ and $(j_2, H, \Phi_2, \ell_2)$ be two interfaces. Two partial minor-embeddings $\varphi_1 \in \Phi_1$ and $\varphi_2 \in \Phi_2$ are compatible if all vertices $v \in X_{j_1} \cap X_{j_2}$ satisfy $\varphi_1^{-1}(v) = \varphi_2^{-1}(v)$. If $\varphi_1$ and $\varphi_2$ are compatible, we can obtain a new partial minor embedding by combining the two partial embeddings into a new partial minor embedding $\varphi_{12}$.

Let $\Phi_{12}$ be the set of partial minor embeddings combined in this manner from all pairs of compatible partial minor embeddings in $\Phi_1 \times \Phi_2$. Let $W_1 := W(j, H, \Phi_1, \ell_1)$ and $W_2 := W(j', H, \Phi_2, \ell_2)$. If $\ell := \ell_1 = \ell_2$ and $\Phi := \Phi_1 = \Phi_2$ we set $W(i, H, \Phi_1, \ell) = \max\{W_1, W_2\}$. Otherwise, we set $W(i, H, \Phi_1, \ell_1) = W_1$ and $W(i, H, \Phi_2, \ell) = W_2$. Additionally, we set $W(i, H, \Phi_1 \cup \Phi_2 \cup \Phi_{12}, \ell_1 + \ell_2 - \ell(H)) := W_1 + W_2 - w(H)$.

Since $|\Phi_1 \times \Phi_2|$ is bounded by $|V(F)|^{k+1} \times |V(F)|^{k+1}$, the resulting complexity in total is asymptotically bounded by

$$2^{\binom{k}{2}} \cdot |V(F)|^{2k+2} \cdot 2^{2|V(F)|} \cdot 2^k \cdot |V(F)| \cdot k^2 \cdot M \cdot n = 2^{\mathcal{O}(k^2 + k \log |V(F)| + |V(F)|)} Mn \,.$$

If $\mathcal{F}$ contains more than one obstruction, the running time can be bounded by $2^{\mathcal{O}(k^2 + k \log N + N)} |\mathcal{F}| Ln$ where $N$ denotes the maximum number of vertices of any graph in $\mathcal{F}$. □

The following result can be obtained by a straightforward modification of the approach sketched in this section. With the technique developed in the next section, this result will yield an FPTAS for the Relaxed Maximum Density Subgraph problem.

**Corollary 1.** *Let $(G, w, \ell, W)$ be an instance of the* RMDS *and let $G$ and $\mathcal{F}$ be as in Theorem 4. Then for any $\lambda \in \mathbb{R}$ a maximum penalized density $\mathcal{F}$-minor-free $(W, \lambda)$-viable pattern can be computed in time $\mathcal{O}(2^{\mathcal{O}(k^2 + k \log N + N)} |\mathcal{F}| \lambda n)$ where $N = \max_{F \in \mathcal{F}} |V(F)|$.*

## 3 An FPTAS for Relaxed Density Maximization

In this section we consider the Relaxed Maximum Density Subgraph (RMDS) problem, where the upper bound on the length may be violated at the cost of some penalty. We assume that the weight function is strictly positive. While it is NP-hard to decide whether a feasible solution exists for the original problem, we show that this slight relaxation allows us to give an FPTAS for penalized density. This can be applied to any problem that allows a quasi-polynomial-time algorithm that computes an optimal solution with respect to the penalized density. Note that the relaxed density maximization problem remains NP-hard as we can choose $L$ very small such that every subgraph is penalized and we have that $\widetilde{\varrho}(H) \approx (1/2)\varrho(H)$ for any subgraph $H$. Then the NP-hardness result of Theorem 1 naturally applies to this problem. For simplicity, we will assume that the scaling constant $c$ for the penalized density equals 1.

Let $\Pi$ be a relaxed density maximization problem that admits an algorithm $\mathcal{A}$ that takes as input an instance $I$ of the relaxed density maximization problem and $\lambda \in \mathbb{N}$ and computes an optimal $(W, \lambda)$-viable pattern $H$ with respect to penalized density, $\widetilde{\varrho}$, in $\mathcal{O}(p(\lambda, n))$ time, where $p(\lambda, n)$ is a function that is polynomial in $\lambda$ and $n$. We show how to construct an FPTAS for $\Pi$ that uses $\mathcal{A}$ as a subroutine. We present our algorithm within the terminology introduced by Schuurman and Woeginger for approximation schemes [19]. We first structure the output of our algorithm to form exponentially growing buckets based on the length of the solutions. In order to compute approximately optimal solutions in each of the buckets efficiently we structure the input of algorithm $\mathcal{A}$ by exponentially compressing the lengths and weights in such a way that the error resulting from the compression is proportional to the size of the solutions in each bucket.

Let $k$ be a suitably chosen integer depending on $\varepsilon$, which will be defined later. We structure the output in $\lfloor \log_k B \rfloor - 1$ buckets, where $B = \ell(G)$, such that bucket $i$ with $0 \leq i \leq \lfloor \log_k B \rfloor - 2$ contains solutions with total length at most $k^{i+2}m$, where $m$ is the number of edges. For each bucket we compute an approximately optimal solution and return the overall best solution as output of our algorithm. To compute an approximately optimal solution for bucket $i$ we structure the input by considering instances $I_i = (G, \ell_i, w_i, W_i, L_i)$, where $\ell_i(e) = \lceil \ell(e)/k^i \rceil$, $w_i(e) = w(e)/k^i$ for $e \in E(G)$ and $W_i = W/k^i$ as well as $L_i = L/k^i$. We apply algorithm $\mathcal{A}$ on instance $I_i$ with $\lambda = k^2 m$. A high-level description of this algorithm is listed as Algorithm 1.

When considering the $i$-th bucket, we refer to the deviation of $H \subseteq G$ with respect to $\ell_i$ and $L_i$ as $\Delta_i(H) = \max\{0, \ell_i(H) - L_i\}$. Similarly the penalized density of $H \subseteq G$ is defined as $\widetilde{\varrho}_i(H) = w_i(H)/(\ell_i(H) + \Delta_i(H))$. Lemma 3 shows that our structuring scheme implies that $\widetilde{\varrho}_i(H)$ is monotonically decreasing in $i$ for each $H \subseteq G$. Corollary 2 establishes a length lower bound on a certain set of subgraphs which will be used in Lemma 4 to show that for each subgraph $H$ there is a bucket $i$ such that $\widetilde{\varrho}_i(H) \approx \widetilde{\varrho}(H)$.

**Algorithm 1:** FPTAS for Relaxed Density Maximization

**Input**: An instance $(G, w, \ell, W, L)$ of Relaxed Maximum Density Subgraph, a real number $0 < \epsilon < 1$

**Output**: An $(1-\epsilon)$-approximation of the maximum penalized density subgraph

$k \leftarrow \lceil \frac{2}{\epsilon} \rceil$ ;

**for** $i \leftarrow 0$ **to** $\lfloor \log_k B \rfloor - 1$ **do**

  $H_i \leftarrow$ result of $\mathcal{A}$ on instance $I_i$ with $\lambda = k^2 m$

**return** $\max_{0 \le i \le \lfloor \log_k B \rfloor} \widetilde{\varrho}_i(H_i)$

**Lemma 3.** *For any subgraph $H$ and each $1 \le i < \lfloor \log_k B \rfloor$, we have*

$$\ell(H) \le k^i \cdot \ell_i(H) \le \ell(H) + |E(H)| \cdot k^i, \tag{1}$$

$$\Delta(H) \le k^i \cdot \Delta_i(H) \le \Delta(H) + |E(H)| \cdot k^i, \tag{2}$$

$$\widetilde{\varrho}_i(H) \le \widetilde{\varrho}_{i-1}(H) \le \widetilde{\varrho}(H), \tag{3}$$

$$\ell_i(H) \le k \cdot m \text{ implies that } \ell_{i-1}(H) \le k^2 \cdot m\,. \tag{4}$$

*Proof.* We will use the following equation, which holds for any real positive numbers $r, s \in \mathbb{R}$.

$$r \le s \cdot \left\lceil \frac{r}{s} \right\rceil \le r + s \tag{$\star$}$$

We start out by proving Eq. (1), which relates the length of a graph to the length of the corresponding subgraph in the compressed instance of iteration $i$. By the definition of $\ell$ and Eq. ($\star$) we have

$$\ell(H) = \sum_{e \in E(H)} \ell_e \le k^i \cdot \underbrace{\sum_{e \in E(H)} \left\lceil \frac{\ell_e}{k^i} \right\rceil}_{=\ell_i(H)} \le \sum_{e \in E(H)} \ell_e + |E(H)| \cdot k^i = \ell(H) + |E(H)| \cdot k^i\,.$$

For Eq. (2) note that the first part trivially holds if $\Delta(H) = 0$. If $\Delta(H) > 0$ holds, by applying Eq. (1) we get $\Delta(H) = \ell(H) - L \le k^i \cdot \ell_i(H) - L \le k^i \cdot \Delta_i(H)$. The second part of Eq. (2) again trivially holds if $\Delta_i(H) = 0$. Similarly, if $\Delta_i(H) > 0$, we get $k^i \Delta_i(H) = k^i \ell_i(H) - L \le \ell(H) + |E(H)| \cdot k^i - L = \Delta(H) + |E(H)| \cdot k^i$.

Note that the first part of Eq. (3) implies the second part since $\widetilde{\varrho}_0(H) = \widetilde{\varrho}(H)$ holds for any subgraph $H$. Directly from Inequality ($\star$) we get that $k \cdot \ell_i(H) \ge \ell_{i-1}(H)$ for any subgraph $H$, which immediately implies Eq. (4). Similarly, we also get $k \cdot \Delta_i(H) \ge \Delta_{i-1}(H)$ since $\max\{0, r\} \ge \max\{0, s\}$ for $r \ge s$. Therefore, we obtain

$$\widetilde{\varrho}_i(H) = \frac{w_{i-1}(H)}{k \cdot (\ell_i(H) + \Delta_i(H))} \le \frac{w_{i-1}(H)}{\ell_{i-1}(H) + \Delta_{i-1}(H)} = \widetilde{\varrho}_{i-1}(H)$$

□

using $w_i(H) = w_{i-1}(H)$ and $\ell_i(H) + \Delta_i(H) = k \cdot (\ell_i(H) + \Delta_i(H))$. Let $\Omega(H)$ be the smallest integer such that $\ell_{\Omega(H)}(H) \le k^2 m$. In other words, $\Omega(H)$ denotes the smallest bucket for which $H$ will be considered by algorithm $\mathcal{A}$. Equation (4) immediately implies a lower bound on the length of $H$ in this bucket.

**Corollary 2.** *For any subgraph $H$, $\Omega(H) > 0$ implies $\ell_{\Omega(H)}(H) > km$.*

Now we are ready to bound the density of an instance $H$ in bucket $\Omega(H)$ in terms of $k$ and its true penalized density $\widetilde{\varrho}(H)$.

**Lemma 4.** *For any subgraph $H$, we have $\widetilde{\varrho}(H) \le \frac{k+1}{k-1} \cdot \widetilde{\varrho}_{\Omega(H)}(H)$.*

*Proof.* Clearly, this inequality holds when $\Omega(H) = 0$. For $\Omega(H) \ge 1$, by Lemma 3, Equation 1, and $|E(H)| \le m$ we immediately get $\ell(H) \ge k^{\Omega(H)} \cdot (\ell_{\Omega(H)}(H) - m)$. Together with Corollary 2

this implies $k^{\Omega(H)}m \leq 1/(k-1) \cdot \ell(H) \leq 1/(k-1) \cdot (\ell(H) + \Delta(H))$. We now compute the density of $H$ in Iteration $\omega := \Omega(H)$ by using Equations (1) and (2).

$$\begin{aligned}\widetilde{\varrho}_\omega(H) &= \frac{w_\omega(H)}{\ell_\omega(H) + \Delta_\omega(H)} = \frac{w(H)}{k^\omega \cdot (\ell_\omega(H) + \Delta_\omega(H))} \\ &\geq \frac{wt(H)}{\ell(H) + \Delta(H) + 2m \cdot k^\omega} \geq \frac{w(H)}{\left(1 + \frac{2}{k-1}\right)(\ell(H) + \Delta(H))} = \frac{k-1}{k+1} \cdot \widetilde{\varrho}(H)\ .\end{aligned}$$

□

**Theorem 5.** *Given $0 < \varepsilon < 1$, we can compute a $(1-\varepsilon)$-approximation for the relaxed density maximization problem in $\mathcal{O}(p(m/\varepsilon^2, n) \log B)$ time, where $G$ is the input graph and $B$ is the maximum length of the edges, provided that an $\mathcal{O}(p(\lambda, n))$ time algorithm for the penalized density maximization as described above is available.*

*Proof.* Clearly, the algorithm computes a $W$-viable solution if one exists due to the correctness of algorithm $\mathcal{A}$, the fact that we do not introduce any errors when scaling the weights, and since the union of the buckets covers all feasible solutions.

Next we show that the algorithm indeed produces a $(1-\varepsilon)$ approximation of the optimal penalized density. Let opt be an optimal solution and $H^*$ be the solution returned by our algorithm. By the above lemmas and choosing $k = \lceil \frac{2}{\varepsilon} \rceil$, we have

$$\begin{aligned}\widetilde{\varrho}(H^*) &\geq \max_{i \geq \Omega(H^*)} \widetilde{\varrho}_i(H^*) \geq \max_{i \geq \Omega(\text{opt})} \widetilde{\varrho}_i(\text{opt}) \geq \widetilde{\varrho}_{\Omega(\text{opt})}(\text{opt}) \\ &\geq \left(\frac{k-1}{k+1}\right) \cdot \widetilde{\varrho}(\text{opt}) \geq \left(1 - \frac{2}{k+1}\right) \cdot \widetilde{\varrho}(\text{opt}) \geq (1-\varepsilon) \cdot \widetilde{\varrho}(\text{opt}).\end{aligned}$$

The running time of this approach is clearly $\mathcal{O}(p(m/\varepsilon^2, n) \log B)$ since $k = \lceil \frac{2}{\varepsilon} \rceil \geq 2$, and $\log_k B \leq \log_2 B = \mathcal{O}(\log B)$. □

For reasons of simplicity, we assumed a scaling factor $c = 1$. By choosing $k = \lceil c + 1/\varepsilon \rceil$ we can accomplish the same result for any scaling factor $c \neq 1$. In our analysis, we further assumed that we are given an algorithm $\mathcal{A}$ that computes a $(W, \lambda)$-viable pattern for a given value of $\lambda$. However, our approach still works if $\mathcal{A}$ only computes a $W$-viable pattern with maximum penalized density. In each iteration we pre-process the instance $I_i$ by removing edges which are longer than $k^2 m$ from $G$. Then the maximum length of any $W$-viable pattern considered by $\mathcal{A}$ will naturally be bounded by $k^2 m^2$. The running time of the resulting FPTAS is bounded by $\mathcal{O}(p(m^2/\varepsilon^2, n) \log B)$, assuming that $\mathcal{A}$ has a running time bounded by $\mathcal{O}(p(\ell(G), n))$. Finally, with the results from Corollary 1 we immediately obtain the following result as an application of the FPTAS to the problem of maximizing the penalized density objective function.

**Corollary 3.** *Let $(G, w, \ell, W)$ be an instance of the* RMDS *problem such that $G$ has treewidth at most $k$ and let $\mathcal{F}$ be a finite set of graphs. Let $B := \ell(G)$, $0 < \varepsilon < 1$ and let* opt *be the optimal penalized density of an $\mathcal{F}$-minor-free $W$-viable pattern. Then a $W$-viable $\mathcal{F}$-minor-free pattern with penalized density at least $(1-\varepsilon) \cdot$ opt can be computed in time $\mathcal{O}(2^{\mathcal{O}(k^2 + k \log N + N)} |\mathcal{F}| m/\varepsilon^2 \log B)$.*

## 4 Conclusion and Outlook

We have investigated bi-objective network design problems with one minimization and one maximization objective in the presence of additional constraints by studying the complexity of maximizing the ratio of the two objectives for different classes of graphs. Although the general problem is NP-hard like many multi-objective optimization problems, we were able to efficiently solve some special cases efficiently and give an FPTAS for the relaxed problem. We presented an efficient algorithm for computing a maximum density path in a tree in sub-quadratic time and we showed that the problem for general graphs is FPT with respect to the number of edges we must remove in order to obtain a tree. Further, we proposed pseudo-polynomial-time algorithms for graphs with

bounded treewidth and a framework for obtaining polynomial-time approximation schemes for the relaxed density maximization problem, given a pseudo-polynomial-time algorithm.

Preliminary work on other structural constraints, such as Steiner-constraints requiring a subset of the vertices to be included in any feasible solution, showed that the resulting problems are also NP-hard, even if we drop the bounds and allow only a fixed number of Steiner-vertices.

## References


1. Vojtech Bálint. The non-approximability of bicriteria network design problems. *J. of Discrete Algorithms*, 1:339–355, June 2003.
2. Hans L. Bodlaender. A linear time algorithm for finding tree-decompositions of small treewidth. In *STOC '93: Proceedings of the 25th annual ACM symposium on Theory of computing*, pages 226–234, New York, NY, USA, 1993. ACM.
3. Altannar Chinchuluun and Panos Pardalos. A survey of recent developments in multiobjective optimization. *Annals of Operations Research*, 154:29–50, 2007.
4. Kai-min Chung and Hsueh-I Lu. An optimal algorithm for the maximum-density segment problem. *SIAM J. Comput.*, 34(2):373–387, 2005.
5. David Eppstein. Subgraph isomorphism in planar graphs and related problems. In *Proc. 6th Ann. ACM-SIAM Sympos. Disc. Alg.*, pages 632–640. SIAM, 1995.
6. Michael H. Goldwasser, Ming-Yang Kao, and Hsueh-I Lu. Linear-time algorithms for computing maximum-density sequence segments with bioinformatics applications. *J. Comput. Syst. Sci.*, 70(2):128–144, 2005.
7. Sun-Yuan Hsieh and Chih-Sheng Cheng. Finding a maximum-density path in a tree under the weight and length constraints. *Information Processing Letters*, 105(5):202 – 205, 2008.
8. Sun-Yuan Hsieh and Ting-Yu Chou. *Algorithms and Computation*, volume 3827 of *LNCS*, chapter Finding a Weight-Constrained Maximum-Density Subtree in a Tree, pages 944–953. Springer Berlin / Heidelberg, 2005.
9. Ross B Inman. A denaturation map of the lambda phage dna molecule determined by electron microscopy. *Journal of Molecular Biology*, 18(3):464–476, 1966.
10. T. Kloks. *Treewidth, Computations and Approximations*. LNCS. Springer, 1994.
11. Hoong Chuin Lau, Trung Hieu Ngo, and Bao Nguyen Nguyen. Finding a length-constrained maximum-sum or maximum-density subtree and its application to logistics. *Discrete Optimization*, 3(4):385 – 391, 2006.
12. D. T. Lee, Tien-Ching Lin, and Hsueh-I Lu. Fast algorithms for the density finding problem. *Algorithmica*, 53(3):298–313, 2009.
13. Yaw-Ling Lin, Tao Jiang, and Kun-Mao Chao. Efficient algorithms for locating the length-constrained heaviest segments with applications to biomolecular sequence analysis. *J. Comput. Syst. Sci.*, 65(3):570–586, 2002.
14. Hsiao-Fei Liu and Kun-Mao Chao. Algorithms for finding the weight-constrained k longest paths in a tree and the length-constrained k maximum-sum segments of a sequence. *Theor. Comput. Sci.*, 407(1-3):349–358, 2008.
15. G. Macaya, J.-P. Thiery, and G. Bernardi. An approach to the organization of eukaryotic genomes at a macromolecular level. *Journal of Molecular Biology*, 108(1):237 – 254, 1976.
16. Madhav V. Marathe, R. Ravi, Ravi Sundaram, S. S. Ravi, Daniel J. Rosenkrantz, and Harry B. Hunt. Bicriteria network design problems,. *Journal of Algorithms*, 28(1):142 – 171, 1998.
17. Mark H. Overmars and Jan van Leeuwen. Maintenance of configurations in the plane. *Journal of Computer and System Sciences*, 23(2):166 – 204, 1981.
18. N. Robertson and P. D. Seymour. Graph minors .XIII. The disjoint paths problem. *Journal of Combinatorial Theory, Series B*, 63(1):65 – 110, 1995.
19. P. Schuurman and G. Woeginger. Approximation schemes – a tutorial. preliminary version of a chapter in the book "Lectures on Scheduling", to appear in 2011.
20. Bang Ye Wu. An optimal algorithm for the maximum-density path in a tree. *Inf. Process. Lett.*, 109(17):975–979, 2009.
21. Bang Ye Wu, Kun-Mao Chao, and Chuan Yi Tang. An efficient algorithm for the length-constrained heaviest path problem on a tree. *Inf. Process. Lett.*, 69(2):63–67, 1999.