J Comb Optim (2013) 26:723–754
DOI 10.1007/s10878-012-9465-z

The density maximization problem in graphs

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

Published online: 7 March 2012

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 under various additional constraints. 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.

First, we consider the problem of computing a maximum density pattern with weight at least W and length at most L in a host G. We call this problem the *bi-*

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.

B. Katz · M. Krug · I. Rutter (✉) · D. Wagner
Faculty of Informatics, Karlsruhe Institute of Technology (KIT), Am Fasanengarten 5, 76131 Karlsruhe, Germany
e-mail: ignaz.rutter@kit.edu

B. Katz
e-mail: bastian.katz@kit.edu

M. Krug
e-mail: marcus.krug@kit.edu

D. Wagner
e-mail: dorothea.wagner@kit.edu

M.-J. Kao · D.T. Lee
Dep. of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan

M.-J. Kao
e-mail: d97021@csie.ntu.edu.tw

D.T. Lee
e-mail: dtlee@csie.ntu.edu.tw

constrained density maximization problem. This problem can be interpreted in terms of maximizing the 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.

Second, we consider the maximum density subgraph problem under structural constraints on the vertex set that is used by the patterns. While a maximum density perfect matching can be computed efficiently in general graphs, the maximum density Steiner-subgraph problem, which requires a subset of the vertices in any feasible solution, is NP-hard and unlikely to admit a constant-factor approximation. When parameterized by the number of vertices of the pattern, this problem is W[1]-hard in general graphs. On the other hand, it is FPT on planar graphs if there is no constraint on the pattern and on general graphs if the pattern is a path.

Keywords Subgraph · Multi-criteria optimization · Complexity · Algorithm

1 Introduction

Many realistic network construction problems are characterized by complex constraints and multiple, possibly conflicting objectives, and are therefore formulated within the framework of multi-objective optimization (Chinchuluun and Pardalos 2007). There are several ways to define optimality in the context of more than one objective, among them Pareto-optimality and aggregate optimality. Let R be a multi-objective optimization problem and let I be the set of the instances of R. Further, let $f_1, \dots, f_k : I \to \mathbb{R}$ be the set of objective functions of R and assume that these functions must be maximized. Then an instance $x \in I$ is *Pareto-optimal* if there is no instance $y \in I$ such that $f_i(y) \geq f_i(x)$ for all $1 \leq i \leq k$ and $f_j(y) > f_j(x)$ for at least one index $1 \leq j \leq k$. That is, x is Pareto-optimal if we cannot improve a single objective without diminishing at least one of the other objectives. While Pareto-optimality seems to capture the classical notion of optimality best, there may be many Pareto-optimal instances for a multi-objective optimization problem. Since it is sometimes undesirable to confront users with a possibly large number of Pareto-optimal solutions, 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, that is we try to maximize

$$F(x) := \sum_{i=1}^{k} w_i \cdot f_i(x) \quad \text{or} \quad G(x) := \max\{w_i \cdot f_i(x) \mid 1 \leq i \leq k\},$$

where $w_i \in \mathbb{R}$ are constant weights. 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 while the other must be minimized. We can think of these objectives as profit and cost of an investment, respectively. Additionally, we assume that we are given an upper bound on the objective to be minimized and a lower bound on the objective to be maximized. That is, we are given a limited budget and a target profit. Two 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, termed return on investment, is a common measure for assessing the quality of investments.

Our framework is defined as follows. Let $G = (V, E)$ be a graph, which we will refer to as the *host*. Throughout the paper we write $n := |V(G)|$ and $m := |E(G)|$, respectively. 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 write $w_e := w(e)$ and $\ell_e := \ell(e)$ and for a subgraph $H \subseteq G$ we define

$$w(H) := \sum_{e \in E(H)} w_e \quad \text{and} \quad \ell(H) := \sum_{e \in E(H)} \ell_e$$

as the weight and length of the edge set of a subgraph H, respectively. 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 $w(H) \geq W$ 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 $\varrho(H) = w(H)/\ell(H)$, called the *density of* H. Given a tuple (G, w, ℓ, W, L), the BI-CONSTRAINED MAXIMUM DENSITY SUBGRAPH (BMDS) problem asks for a *connected* (W, L)-viable pattern $H \subseteq G$ with maximum density.

Problem BI-CONSTRAINED MAXIMUM DENSITY SUBGRAPH (BMDS)

Instance:	A graph $G = (V, E)$, weight $w\colon E \to \mathbb{Z}$, length $\ell\colon E \to \mathbb{N}$, $W \in \mathbb{Z}$, $L \in \mathbb{N}$
Solution:	A connected (W, L)-viable subgraph $H \subseteq G$
Goal:	Maximize $\varrho(H)$

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, ℓ, 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.

Problem RELAXED MAXIMUM DENSITY SUBGRAPH (RMDS)

Instance: A graph $G = (V, E)$, weight $w\colon E \to \mathbb{Z}$, $W \in \mathbb{Z}$, $L \in \mathbb{N}$
Solution: A connected W-viable subgraph $H \subseteq G$
Goal: Maximize $\widetilde{\varrho}(H)$

A different set of constraints we consider arises from the fact that we may have to include a given set $S \subseteq V$ of vertices, so-called *terminals*, in any feasible solution, for instance, if we would like to augment an already existing infrastructure network. We will refer to these constraints as *Steiner constraints*. In the presence of Steiner constraints we drop the constraints on the length and weight of the solution.

Problem MAXIMUM DENSITY STEINER SUBGRAPH

Instance: A graph $G = (V, E)$, weight $w\colon E \to \mathbb{Z}$, $W \in \mathbb{Z}$, $S \subseteq V$
Solution: A connected W-viable subgraph $H \subseteq G$ such that $S \subseteq V(H)$
Goal: Maximize $\varrho(H)$

Related Work An overview of recent developments in multi-objective optimization is given in Chinchuluun and Pardalos (2007). Bálint (2003) 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. (1998) 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 in bi-weighted sequences arises from the investigation of non-uniformity of nucleotide composition with genomic sequences (Inman 1966; Macaya et al. 1976) and has received considerable attention in bio-informatics. For this problem, 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. (2002) give an $\mathcal{O}(n \log A)$ algorithm, which is improved to $\mathcal{O}(n)$ by Goldwasser et al. (2005). A linear time algorithm for the non-uniform case is given by Chung and Lu (2005). Lee et al. (2009) show how to select a subsequence whose density is closest to a given density δ in $\mathcal{O}(n \log^2 n)$ time. Without the upper bound on the length B they present an optimal $\mathcal{O}(n \log n)$-time algorithm.

Subsequently, this problem has been generalized to graphs. Previous work on this problem focuses mostly on the cases where the host is a tree subject to the two-sided constraint on the length of the solution. Hsieh et al. (2008, 2005) 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. (2009, 1999) 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 (Wu et al. 1999), which is improved to $\mathcal{O}(n \log n)$ by Liu and Chao (2008).

Problems involving Steiner constraints have been widely studied in computer science for a long time. For instance, it is known that the Steiner tree problem is NP-hard (Garey and Johnson 1979) and can be approximated within a factor of 1.55 (Robins and Zelikovsky 2000). When parameterized by the number of terminals, this problem is FPT (Dreyfus and Wagner 1971), when parameterized by the number of non-terminals in the solution it is W[2]-hard. The latter result is attributed to Bodlaender and can be found in Lokshtanov (2009). For the special case that the set of terminals contains all vertices of the graph, Chandrasekaran (1977) shows that a spanning tree with maximum density can be computed in polynomial time.

Contribution In Sect. 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 Sect. 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 Sect. 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. Additionally, 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 Sect. 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. In Sect. 4 we drop the bounds on the weight and length, respectively, and consider structural constraints instead. First, we consider general vertex constraints in Sect. 4.1. We adapt a generic technique previously used by Chandrasekaran (1977) for solving maximum density Steiner subgraph problems based on parametric search and we show how this technique can be used to solve the maximum density perfect matching problem efficiently. Further, we show how to compute a maximum density tree with k leaves in a tree in polynomial time. Then we focus on Steiner constraints requiring a given subset S of the vertices in any feasible solution in Sect. 4.2. We show that this problem is NP-hard and cannot be approximated by a constant factor unless $P = NP$, even if the pattern is a path and S contains only one vertex. Further, when parameterized by the number of vertices of the pattern, we show that the maximum density subgraph problem is W[1]-hard, that is, it is unlikely to admit an FPT-algorithm. In contrast, we show that this problem is FPT on planar graphs. Then we show that the problem of computing a maximum density path is FPT when parameterized by the number of vertices on the path in general graphs. However, we also show that it is NP-hard to find a maximum density Steiner tree. Table 1 gives an overview of the results obtained in this paper.

2 The bi-constrained maximum density subgraph problem

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

Table 1 Summary of the results obtained in this paper. The symbol $\star$ denotes an arbitrary graph

G	H	Constr.	Results	Reference
Bi-constrained maximum density subgraph problem				
$tw = 2$	path	bi-constr.	NP-hard	Theorem 1
tree	path	bi-constr.	$\mathcal{O}(n \log^3 n)$	Theorem 2
tree $+k$ edges	path	bi-constr	$\mathcal{O}(2^k k^2 n \log^2 n + n \log^3 n)$	Theorem 3
$tw = k$	minor-closed	bi-constr	$2^{\mathcal{O}(k^2 + k \log N + N)} \lvert\mathcal{F}\rvert Ln$	Theorem 4
$tw = k$	minor-closed	relaxed	$2^{\mathcal{O}(k^2 + k \log N + N)} \lvert\mathcal{F}\rvert m/\varepsilon^2 \log B$	Corollary 1
Maximum density steiner subgraph				
$\star$	matching	V	$\mathcal{O}((m + n \log n) n \log(nM))$	Corollary 4
tree	tree with k leaves	$\lvert S\rvert \geq 1$	$\mathcal{O}(k^2 n \log(nM))$	Theorem 7
$\star$	path	$\lvert S\rvert = 1$	NP-hard, $\notin$ APX	Theorem 8
$\star$	$\star$, $\lvert V(H)\rvert \leq k$	$\lvert S\rvert = 1$	W[1]-hard	Theorem 9
planar	$\star$, $\lvert V(H)\rvert \leq k$	$\lvert S\rvert = 1$	FPT	Theorem 10
$\star$	path, $\lvert V(H)\rvert \leq k$	$\lvert S\rvert \geq 1$	$\mathcal{O}((2^{k-s} m + 3^{k-s}) s^2 \log(nM))$	Theorem 11
$\star$	tree	$\lvert S\rvert \geq 1$	NP-hard	Theorem 12

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 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 is a simply connected outerplanar graph with treewidth* 2, *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, that is, a set of positive integers $C = \{c_1, c_2, \dots, c_m\}$ with $M = \sum_{i=1}^m c_i$ and we ask whether there is a subset $I' \subseteq \{1, \dots, m\}$ with $\sum_{c_i \in I'} c_i = M/2$. We transform this into an instance of BMDS as illustrated in Fig. 1. First, we create a path $v_0, v_1, \dots, v_{2m}$ with $w_e = \ell_e = 1$ for each edge e on this path. Besides, we create additional m vertices, $p_1, p_2, \dots, p_m$, 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 $d := W/(3M + 2m)$ if and only if the corresponding instance of PARTITION can be solved. Clearly, any partition can be transformed into a path with density d and length W. Let (C_1, C_2) be a partition of C and let S denote the indices of the elements in C_1, that is, $\sum_{i \in S} c_i = M/2$. Consider the simple path P from q_0 to q_1 that visits all vertices v_i with $i \in S$ and that contains none of the vertices v_j with $j \notin S$. This path

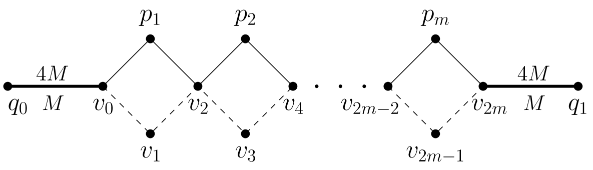

Fig. 1 Graph used in the reduction from PARTITION. *Bold edges* have density 4, all *other edges* have density 1. *Dashed edges* have weight and length 1, *solid non-bold edges* incident to p_i have weight and length $c_i + 1$

has weight $8M + 2m + 2\sum_{i\in S} c_i = 9M + 2m$ and length $2M + 2m + \sum_{i\in S} c_i = 3M + 2m$. Hence, it has density d.

Conversely, assume that P is a path with density at least d. 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 if and only if $i \in S$. Then the density of this path can be expressed as

$$\frac{8M + 2m + 2\sum_{i\in S} c_i}{2M + 2m + 2\sum_{i\in S} c_i},$$

which is strictly decreasing as $2\sum_{i\in S} c_i$ is increasing. Hence we have $2\sum_{i\in S} c_i \le M$. On the other hand the weight of the path must be at least W, which implies $2\sum_{i\in S} c_i \ge M$. 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 $L = 3M + 2m$ in the reduction above 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. Note that the treewidth of such a graph is bounded by $k + 1$. We prove this result in two steps. 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 (2009) and Lau et al. (2006) 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.

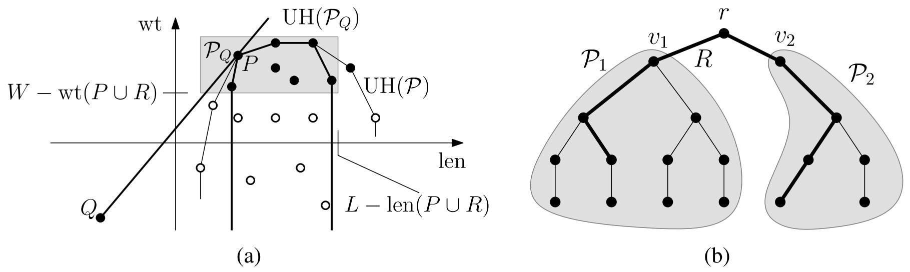
wt
$\mathrm{UH}(\mathcal{P}_Q)$
$\mathcal{P}_Q$
P
$W - \mathrm{wt}(P \cup R)$
$\mathrm{UH}(\mathcal{P})$
len
Q
$L - \mathrm{len}(P \cup R)$
(a)
r
v_1
v_2
$\mathcal{P}_1$
R
$\mathcal{P}_2$
(b)

Fig. 2 Tangent query to find the best candidate for Q (**a**) and combination of two paths using vertex r (**b**)

Throughout the section we use the key idea that the combined density of two disjoint 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 if and only if $w(P) \geq W - w(Q)$ 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 $(-\infty, L - \ell(Q)] \times [W - w(Q), \infty)$ as illustrated in Fig. 2(a). 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 $\mathrm{UH}(\mathcal{P}_Q)$, which is more efficient than trying all possible combinations.

We use a dynamic data structure for the maintenance of the upper chain of the convex hull of a set of points P (Overmars and van Leeuwen 1981) 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 that it inherits from its father and, whether these points are at its left or right boundary. 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 $\mathrm{UH}(\mathcal{P}_Q)$ 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* $\mathrm{UH}(\mathcal{P})$ *as described in* Overmars and van Leeuwen (1981) *the upper convex hull* $\mathrm{UH}(\mathcal{P}_Q)$ *can be computed 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 $\mathcal{P}_Q$. They can be computed in $\mathcal{O}(\log n)$ time using a dynamic priority search tree (McCreight 1985). 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 $\mathrm{UH}(\mathcal{P}_Q) \subseteq \mathrm{UH}(\mathcal{P}')$. Clearly, $\mathcal{P}' = \mathcal{P}_Q \uplus \mathcal{P}''$ 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 $\mathrm{UH}(\mathcal{P}_Q)$ 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 $\mathrm{UH}(\mathcal{P}')$ 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 $\mathrm{UH}(\mathcal{P}')$ 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, ℓ, W, L) *of the* BMDS *problem, where* $T = (V, E)$ *is a tree, a* (W, L)*-viable maximum density path can be computed 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 auxiliary vertices and edges with weight and length 0 in linear time such that the resulting tree has linear size. A *centroid* of a binary tree is a vertex whose removal disconnects T into at most three 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. Clearly, a centroid can be computed in linear time by aggregating weights of the tree starting in the leaves. Let v_1, v_2 be two children of r and let R be the path between v_1 and v_2 via r as illustrated in Fig. 2(b). 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 and compute their density in linear time. 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 $\mathcal{O}(n \log^2 n)$ time. 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 $P \in \mathcal{P}_1$, that is, 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 $\mathcal{O}(\log^2 n)$. During the computation of the upper convex hull we can simultaneously perform the necessary tangent queries using binary search on the constructed hull. Then we can compute the maximum density path P^* through r in time $\mathcal{O}(n \log^2 n)$. We do this for all (at most 6) 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 these edges. The following lemma can be used to enumerate these 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'. There is a linear-time algorithm that either computes such a path or concludes that no path exists.*

Proof We prove the existence of at most one path 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 Δ 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 Δ have even degree. Hence, Δ contains a cycle contradicting the fact that Δ 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 $w_1, \dots, w_\ell$. 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 ℓ_1 and ℓ_2 in their subtrees, then we update the path by matching these loose ends and adding the unique path in T that connects ℓ_1 and ℓ_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, ℓ, 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 a maximum density path containing all edges in F' and we return the maximum density path over all $F' \subseteq F$. 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'. Moreover, R can be computed in linear time from this information.

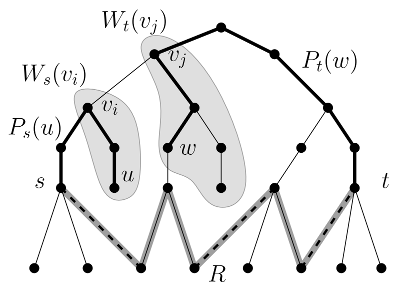

Fig. 3 Illustration for the proof of Theorem 3. Extension of the unique path R using all edges in F (*dashed*) with end vertices s and t

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 $s = v_0, \dots, v_\ell = t$ 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, that is, if $i < j$, otherwise they will have at least one vertex in common as illustrated in Fig. 3.

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 $i = 1, \dots, \ell - 1$ 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 $\mathcal{O}(n \log^3 n)$ running time. 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 $\mathcal{O}(2^k k^2 n \log^2 n + n \log^3 n)$. $\square$

Note, that 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, that is, 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 a *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 (Bodlaender 1993; Kloks 1994) with respect to the treewidth of the graph. 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 ≤ 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 (Eppstein 1995). 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) The union of all bags $\bigcup_{i\in I} X_i$ is equal to V.
(ii) For all edges $e\in E$ there is an index $i\in I$ such that $e\subseteq X_i$.
(iii) For all vertices $v\in V$, the tree induced by the set of nodes $\mathcal{X}_v=\{i\in I \mid v\in X_i\}$ 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,ℓ,W,L) be an instance of the* BMDS *problem such that G has treewidth at most k, let $\mathcal{F}$ be a non-empty finite set of graphs and let $N:=\max_{F\in\mathcal{F}}|V(F)|$. Then a maximum density connected $\mathcal{F}$-minor-free (W,L)-viable pattern can be computed in $2^{\mathcal{O}(k^2+k\log N+N)}|\mathcal{F}|Ln$ time.*

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 (Robertson and Seymour 1995). 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 (Kloks 1994).

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 $\mathcal{C} := \{V_1, \dots, V_q\}$ 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 that 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 $\varphi : V(F) \to V(H/\mathcal{C}) \cup \{\perp, \top\}$ such that $uv \in E(F) \Rightarrow \varphi(u)\varphi(v) \in E(H/\mathcal{C})$ for all $uv \in E(F)$ with $u, v \notin \varphi^{-1}(\perp) \cup \varphi^{-1}(\top)$, hence, φ maps a subgraph of F to a minor of H. The image $\perp$ represents vertices in $G_i - X_i$ and the image $\top$ represents vertices in G which have not been considered yet, that is, vertices in $G - G_i$. A partial minor embedding φ 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 must be disregarded as a partial solution.

For the algorithm we identify the vertices of F with the numbers $1, \dots, |V(F)|$. The images of these vertices under φ that are not contained in $\varphi^{-1}(\perp) \cup \varphi^{-1}(\top)$ correspond to branch-sets of H, that is, a partition of the vertices of H. By considering all $|V(F) + 1|^{k+1}$ labellings of the vertex set of H where each vertex is labeled with some number in $0, \dots, V(F)$, 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 to be images under f. We further encode for each vertex in F that does not have an image under f whether it is mapped to $\perp$ or $\top$. A mapping to $\perp$ 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 of 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 φ. 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 ℓ, such that $G'[X_i] = H \subseteq G[X_i]$ and Φ represents all partial minor embeddings φ of F into G'. We call the quadruple (i, H, Φ, ℓ) 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 $\perp$ 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, Φ, ℓ) where r is the root of the tree decomposition, such that H is connected and does not contain $\top$, Φ is proper and ℓ 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 subgraph H in G_i^* that does not include $\perp$ we compute the set Φ 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', Φ', ℓ') 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 Φ of all partial minor embeddings φ of F into H that can be obtained from some $\varphi' \in \Phi'$ by choosing some vertex in $\varphi'^{-1}(\top)$ to be mapped to v by φ. If all partial minor embeddings φ constructed this way are proper and $\ell := \ell' + \ell(E^+) \leq L$, the interface (i, H, Φ, ℓ) is proper, and we compute $W(i, H, \Phi, \ell) = W_j + w(E^+)$ and 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', Φ', ℓ') 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 Φ of all mappings φ that can be obtained from mappings φ' 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 $\perp$. 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 $\perp$. The resulting complexity of a forget node is asymptotically bounded by

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

Join node i joining j_1 and j_2: Let j_1 and j_2 be the two children of i and consider two interfaces of j_1 and j_2, respectively, given by (j_1, H, Φ_1, ℓ_1) and (j_2, H, Φ_2, ℓ_2). 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 φ_1 and φ_2 are compatible, we can obtain a new partial minor embedding by combining the two partial embeddings into a new partial minor embedding φ_{12}.
Let Φ_{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 $W(i, H, \Phi_1 \cup \Phi_2 \cup \Phi_{12}, \ell_1 + \ell_2 - \ell(H)) := W_1 + W_2 - w(H)$. Clearly, the weight and length of H must be subtracted, since otherwise, these values would be counted twice.

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

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

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, ℓ, 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, λ)-viable pattern can be computed in time $2^{\mathcal{O}(k^2 + k \log N + N)} |\mathcal{F}| \lambda n$ 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 an additional penalty term. 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 FPTAS 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 $\widetilde{\varrho}(H) \approx \varrho(H)/2$ 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 Π 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, λ)-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 λ and n. We show how to construct an FPTAS for Π that uses $\mathcal{A}$ as a subroutine. We present our algorithm within the terminology introduced by Schuurman and Woeginger for approximation schemes (Schuurman and Woeginger 2011). 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.

Assume that we are given a graph T. Let k be a suitably chosen integer depending on ε, 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 $0 \le i \le \lfloor \log_k B \rfloor - 2$ 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 can think of these instances as being *compressed*. We apply algorithm $\mathcal{A}$ on the compressed instances 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 ℓ_i and L_i as the *compressed deviation* $\Delta_i(H) = \max\{0, \ell_i(H) - L_i\}$. Similarly, the penalized density of $H \subseteq G$ is defined as the *compressed penalized density* $\widetilde{\varrho}_i(H) = w_i(H)/(\ell_i(H) + \Delta_i(H))$.

In order to show that Algorithm 1 is an FPTAS we proceed in several steps. First, we bound the compressed penalized density used in the i-th iteration of the algorithm in terms of the ordinary penalized density. In Lemma 4 we use this bound to derive an approximation ratio for the penalized density. Finally, we show that the algorithm is an FPTAS in Theorem 5.

Algorithm 1: FPTAS for Relaxed Density Maximization

```
Input: An instance $(G, \mathrm{wt}, \mathrm{len}, W, L)$ 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 $\max_{0 \le i \le \lfloor \log_k B \rfloor} \widetilde{\varrho}(H_i)$
```

Lemma 3 *For any subgraph H and each $1 \le i < \lfloor \log_k B \rfloor$, the following holds*

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

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

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

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

Proof We 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 (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 ℓ and ($\star$) we have

$$\begin{aligned}\ell(H) &= \sum_{e \in E(H)} \ell_e \\ &\le k^i \cdot \sum_{e \in E(H)} \left\lceil \frac{\ell_e}{k^i} \right\rceil \quad \text{by } (\star) \\ &= k^i \ell_i(H)\end{aligned}$$

On the other hand,

$$\begin{aligned}k^i \ell_i(H) &= k^i \cdot \sum_{e \in E(H)} \left\lceil \frac{\ell_e}{k^i} \right\rceil \\ &\le \sum_{e \in E(H)} \left(\ell_e + k^i\right) \quad \text{by } (\star) \\ &= \ell(H) + \left|E(H)\right| \cdot k^i.\end{aligned}$$

Next, we consider (2). Note that the first inequality trivially holds if $\Delta(H) = 0$. So, we may assume that $\Delta(H) > 0$. By applying (1) and the definition of the compressed deviation we obtain

$$\Delta(H) = \ell(H) - L \le k^i \cdot \ell_i(H) - L \le k^i \cdot \Delta_i(H).$$

The second inequality of Equation (2) again trivially holds if $\Delta_i(H) = 0$. For $\Delta_i(H) > 0$, we obtain

$$k^i \Delta_i(H) = k^i \ell_i(H) - L \le \ell(H) + \bigl|E(H)\bigr| \cdot k^i - L = \Delta(H) + \bigl|E(H)\bigr| \cdot k^i$$

using (1) and the definition of the compressed deviation.

Finally, we consider (3) and (4). Note that the first inequality of (3) implies the second inequality since $\widetilde{\varrho}_0(H) = \widetilde{\varrho}(H)$ holds for any subgraph H. From Inequality ($\star$) we obtain that

$$\ell_{i-1}(H) = \sum_{e \in E(H)} \left\lceil \frac{\ell_e}{k^{i-1}} \right\rceil \le \sum_{e \in E(H)} k \cdot \left\lceil \frac{\ell_e}{k \cdot k^{i-1}} \right\rceil = k \cdot \ell_i(H)$$

for any subgraph H, which immediately implies (4). Similarly, we also obtain

$$k \cdot \Delta_i(H) = k \cdot \max\bigl\{0, \ell_i(H) - L\bigr\} \ge k \cdot \max\bigl\{0, \ell(H) - L\bigr\} = \Delta_{i-1}(H)$$

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 $\ell_i(H) + \Delta_i(H) = k \cdot (\ell_i(H) + \Delta_i(H))$, which concludes the proof. □

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 penalized 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 $\widetilde{\varrho}_{\Omega(H)}(H) \ge \frac{k-1}{k+1} \cdot \widetilde{\varrho}(H)$.*

Proof Clearly, this inequality holds if $\Omega(H) = 0$. For $\Omega(H) \ge 1$, we have

$$\begin{aligned} \ell(H) &\ge k^{\Omega(H)} \cdot \ell_{\Omega(H)}(H) - |E(H)| \cdot k^{\Omega(H)} && \text{by (1)} \\ &\ge k^{\Omega(H)} \cdot \bigl(\ell_{\Omega(H)}(H) - m\bigr) && \text{since } |E(H)| \le m. \\ &\ge k^{\Omega(H)} \cdot (km - m) && \text{due to Corollary 2} \\ &= (k-1) \cdot k^{\Omega(H)} m. && (5) \end{aligned}$$

This implies

$$k^{\Omega(H)}m \le \frac{\ell(H)}{k-1} \le \frac{\ell(H)+\Delta(H)}{k-1}. \tag{6}$$

Then the penalized density satisfies

$$\begin{aligned}
\widetilde{\varrho}_{\Omega(H)}(H) &= \frac{w_{\Omega(H)}(H)}{\ell_{\Omega(H)}(H)+\Delta_{\Omega(H)}(H)} && \text{by definition of } \widetilde{\varrho} \\
&= \frac{w(H)}{k^{\Omega(H)}\cdot(\ell_{\Omega(H)}(H)+\Delta_{\Omega(H)}(H))} && \text{by definition of } w_{\Omega(H)} \\
&\ge \frac{w(H)}{\ell(H)+\Delta(H)+2|E|\cdot k^{\Omega(H)}} && \text{by (1) and (2)} \\
&\ge \frac{w(H)}{\ell(H)+\Delta(H)+2m\cdot k^{\Omega(H)}} && \text{since } |E|\le m \\
&\ge \frac{w(H)}{(1+\frac{2}{k-1})(\ell(H)+\Delta(H))} && \text{by (6)} \\
&= \frac{k-1}{k+1}\cdot\widetilde{\varrho}(H).
\end{aligned}$$

This concludes the proof. □

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 total length of the edges, provided that an $\mathcal{O}(p(\lambda,n))$ time algorithm for the penalized density maximization as described above exists.*

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

$$\begin{aligned}
\widetilde{\varrho}(H^*) &\ge \max_{i\ge\Omega(H^*)}\widetilde{\varrho}_i(H^*) && \text{by Algorithm 1} \\
&\ge \max_{i\ge\Omega(\text{opt})}\widetilde{\varrho}_i(\text{opt}) && \text{by Algorithm 1} \\
&\ge \widetilde{\varrho}_{\Omega(\text{opt})}(\text{opt}) && \text{by (3)} \\
&\ge \left(\frac{k-1}{k+1}\right)\cdot\widetilde{\varrho}(\text{opt}) && \text{by Lemma 4} \\
&= \left(1-\frac{2}{k+1}\right)\cdot\widetilde{\varrho}(\text{opt}) \\
&\ge (1-\varepsilon)\cdot\widetilde{\varrho}(\text{opt}) && \text{by definition of } k.
\end{aligned}$$

The running time of this approach clearly is $\mathcal{O}(p(m/\varepsilon^2, n)\log B)$ 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, λ)-viable pattern for a given value of λ. 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 that are longer than $k^2 m$ from G. Then the maximum length of any W-viable pattern considered by $\mathcal{A}$ is naturally 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, ℓ, 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 $(1-\varepsilon)\cdot \text{opt}$ can be computed in time $\mathcal{O}(2^{\mathcal{O}(k^2 + k\log N + N)}|\mathcal{F}|m/\varepsilon^2 \log B)$.*

4 Maximum density subgraphs with structural constraints

In this section we drop the lower bound on the weight as well as the upper bound on the length. Instead we impose structural constraints on the set of vertices by requiring a subset of the vertices to be contained in any feasible solution. This models a scenario in which we would like to inter-connect a subset of the given vertices in a certain way. For instance, we may wish to connect the sites to form a spanning tree or a perfect matching or we may wish to inter-connect a (small) subset of the vertices.

4.1 Parametric search and application

One of the most natural constraints on the vertex set is to require the pattern to span the whole set of vertices. Chandrasekaran (1977) shows that a spanning tree with maximum density can be computed in polynomial time. We provide an adapted version of the main theorem that makes this possible, along with a proof of its correctness, and show how this can be used as a generic tool in order to obtain efficient algorithms for the maximum density subgraph problem.

Theorem 6 (Chandrasekaran 1977) *Let $G = (V, E)$ be a graph with edge weights $a_e \in \mathbb{Z}$, $b_e \in \mathbb{N}$ for $e \in E$ such that $b_e > 0$ for all $e \in E$ and let $\mathcal{S} \in 2^E$ be an arbitrary collection of subsets of the edges. Let*

$$\theta^* = \max_{X \in \mathcal{S}} \left\{ \frac{\sum_{e\in X} a_e}{\sum_{e \in X} b_e} \right\} \quad \textit{and} \quad \varphi(\theta) = \max_{X \in \mathcal{S}} \left\{ \sum_{e \in X} (a_e - \theta b_e) \right\},$$

then

$$\varphi(\theta) \begin{cases} > 0 & \Leftrightarrow \quad \theta < \theta^*, \\ = 0 & \Leftrightarrow \quad \theta = \theta^*, \\ < 0 & \Leftrightarrow \quad \theta > \theta^*. \end{cases}$$

Proof Let $X^* \in \mathcal{S}$ be such that $\varphi(\theta) = \sum_{e \in X^*}(a_e - \theta \cdot b_e)$:

$$\begin{aligned} \varphi(\theta) = \sum_{e \in X^*} (a_e - \theta \cdot b_e) < 0 \\ \Leftrightarrow \quad & \sum_{e \in X} (a_e - \theta \cdot b_e) < 0 \quad \forall X \in \mathcal{S} \\ \Leftrightarrow \quad & \frac{\sum_{e \in X} a_e}{\sum_{e \in X} b_e} < \theta \quad \forall X \in \mathcal{S} \\ \Leftrightarrow \quad & \theta^* < \theta \end{aligned}$$

On the other hand, assume $\theta^* > \theta$. Then and only then there is some $X' \in \mathcal{S}$ such that

$$\begin{aligned} \frac{\sum_{e \in X'} a_e}{\sum_{e \in X'} b_e} > \theta \\ \Leftrightarrow \quad & \exists X' \in \mathcal{S} : \sum_{e \in X'} (a_e - \theta \cdot b_e) > 0 \\ \Leftrightarrow \quad & \varphi(\theta) = \sum_{e \in X^*} (a_e - \theta \cdot b_e) > 0. \end{aligned}$$

Equality for the case $\varphi(\theta) = 0$ follows from the previous observations. □

In order to solve maximum density subgraph problems we adapt the optimization algorithm suggested by Chandrasekaran to our setting. Essentially, the algorithm performs binary search on the density space using Theorem 6 and is listed as Algorithm 2. In Lemma 5 we show that this algorithm finds an optimal solution after a polynomial number of steps, when the interval containing the optimal density is smaller than some value depending on the input numbers. In this case, we return the maximum density of the two solutions corresponding to the interval boundaries.

Lemma 5 *Let $G = (V, E)$ be a graph with edge weights $a_e \in \mathbb{Z}, b_e \in \mathbb{N}$ for $e \in E$ and let $\mathcal{S} \subseteq 2^E$ be a set of feasible solutions. If we can compute*

$$\operatorname{opt}(\theta) = \operatorname*{argmax}_{X \in \mathcal{S}} \left\{ \sum_{e \in X} (a_e - \theta b_e) \right\}$$

in time $f(n)$, then Algorithm 2 computes a solution $X^* \in \mathcal{S}$ with maximum density in $\mathcal{O}(f(n) \log(nM))$ time where M denotes the largest input number.

Algorithm 2: Parametric Search

```
Input: Graph $G = (V, E)$, $a_e \in \mathbb{Z}$, $b_e \in \mathbb{N}$ for $e \in E$, set of feasible solutions
       $\mathcal{S} \subseteq 2^E$
Output: $S^* \subseteq \mathcal{S}$ with maximum density
$[\alpha, \beta] \leftarrow \left[\min_{e\in E}\left\{\frac{a_e}{b_e}\right\}, \max_{e\in E}\left\{\frac{a_e}{b_e}\right\}\right]$
while $\beta - \alpha \geq \left(\sum_{e\in E} b_2\right)^{-2}$ do
    $k \leftarrow \frac{\alpha+\beta}{2}$
    $d \leftarrow \max_{X\in\mathcal{S}}\left\{\sum_{e\in X}(a_e - k \cdot b_e)\right\}$
    if $d > 0$ then
        $[\alpha, \beta] \leftarrow [k, \beta]$
    else if $d < 0$ then
        $[\alpha, \beta] \leftarrow [\alpha, k]$
    else
        $[\alpha, \beta] \leftarrow [k, k]$
$S_\alpha \leftarrow \operatorname{argmax}_{X\in\mathcal{S}}\left\{\sum_{e\in X}(a_e - \alpha \cdot b_e)\right\}$
$S_\beta \leftarrow \operatorname{argmax}_{X\in\mathcal{S}}\left\{\sum_{e\in X}(a_e - \beta \cdot b_e)\right\}$
if $\varrho(S_\alpha) > \varrho(S_\beta)$ then
    return $S_\alpha$
else
    return $S_\beta$
```

Proof First, we show that, if an interval I containing the optimal density is smaller than some threshold value depending on the input numbers, then the optimal solution corresponds to a solution that can be associated with the densities at the boundaries of the interval. Note that every $X \in \mathcal{S}$ corresponds to a linear function $f_X : \theta \mapsto \sum_{e\in X} a_e - \theta \cdot \sum_{e\in X} b_e$. Essentially we are interested in examining the upper envelope of the set of functions $\mathcal{F}_\mathcal{S} := \{f_X \mid X \in \mathcal{S}\}$. This upper boundary is composed of linear segments, such that each segment s corresponds to some $X_s \in \mathcal{S}$ and to some interval I_s on the θ-axis. This interval contains all θ for which X_s maximizes $\sum_{e\in X}(a_e - \theta \cdot b_e)$ over all $X \in \mathcal{S}$. We say that X_s is *dominating* on I_s. See Fig. 4 for an illustration.

Let $f_P, f_Q, f_R, f_S \in \mathcal{F}_\mathcal{S}$ and let θ_{PQ} and θ_{RS} be the intersections of f_P, f_Q and f_R, f_S, respectively, such that $\theta_{PQ} < \theta_{RS}$. Then the length of the interval $[\theta_{PQ}, \theta_{RS}]$ can be bounded from below as follows. Let $A_X := \sum_{e\in X} a_e$ and let $B_X := \sum_{e\in X} b_e$. Then we have

$$f_P(\theta_{PQ}) = A_P + \theta_{PQ} B_P = A_Q + \theta_{PQ} B_Q = f_Q(\theta_{PQ})$$

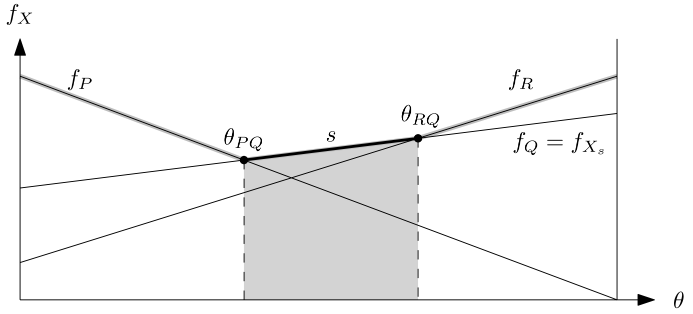

Fig. 4 Upper boundary of the set of functions $\mathcal{F}_{\mathcal{S}}$ (*gray line*), segment s on the upper boundary (*bold black segment*) and corresponding linear function f_{X_s} associated with $X_s \in \mathcal{S}$ and interval that is dominated by s (*gray area*)

$$f_R(\theta_{RS}) = A_R + \theta_{RS} B_R = A_S + \theta_{RS} B_S = f_S(\theta_{RS})$$

which, by a simple transformation, is equivalent to

$$\theta_{PQ} = \frac{A_P - A_Q}{B_Q - B_P} \quad \text{and} \quad \theta_{RS} = \frac{A_R - A_S}{B_S - B_R}.$$

It follows that, if $|\theta_{PQ} - \theta_{RS}| > 0$, we have

$$\begin{aligned} |\theta_{PQ} - \theta_{RS}| &= \left| \frac{A_P - A_Q}{B_Q - B_P} - \frac{A_R - A_S}{B_S - B_R} \right| \\ &= \left| \frac{(A_P - A_Q)(B_S - B_R) - (A_R - A_S)(B_Q - B_P)}{(B_Q - B_P)(B_S - B_R)} \right| \\ &\geq \left| \frac{1}{(B_Q - B_P)(B_S - B_R)} \right| \end{aligned}$$

since the numerator must be an integer

$$\geq \frac{1}{(\sum_{e \in E} b_e)^2}$$

since both B_Q and B_S are bounded by $\sum_{e \in E} b_e$. Hence, whenever $|\theta_{RS} - \theta_{PQ}| > 0$, we have

$$|\theta_{RS} - \theta_{PQ}| > \left(\sum_{e \in E} b_e \right)^{-2}.$$

As a consequence, let $[\alpha, \beta]$ be an arbitrary non-degenerate interval of length less than this value containing the optimal density θ^*. Then this interval may contain at most one intersection point of all the pairs of linear functions $f_P, f_Q \in \mathcal{F}_{\mathcal{S}}$, that is, it intersects at most two segments on the upper boundary of $\mathcal{F}_{\mathcal{S}}$. Then, clearly, either the solution opt(α) corresponding to α or the solution opt(β) corresponding to β must be optimal. Note, that the optimal density will, in general, match neither α nor β in this case.

Next, we prove the bound on the running time. It is clear that $\delta_{\min} \leq \theta^* \leq \delta_{\max}$, where $\delta_{\min}$ and $\delta_{\max}$ denote the minimum and maximum density of an edge of G, respectively. Hence we only need to search the optimal value in the interval $[\delta_{\min}, \delta_{\max}]$. The size of this interval is at most $|2a_{\max}|$, where $a_{\max}$ denotes the maximum weight of an edge. since we assumed $b_e \geq 1$ for all $e \in E$. Algorithm 2 performs binary parameter search on this interval using Theorem 6. In each step we bisect the previous interval, that is, after t steps the interval has size at most $2|a_{\max}|/2^t$. Thus, after $t > \log|a_{\max}| + 2 \cdot \log\sum_{e\in E} b_e + 1$ steps the size of the interval is smaller than $(\sum_{e\in E} b_e)^{-2}$. By previous arguments either the solution corresponding to the left boundary of the interval or the solution corresponding to the right boundary of the interval is an optimal solution. Hence, it suffices to compute two optimal solutions for α and β, respectively, and to compare their densities. The running time of the algorithm is in $\mathcal{O}(f(n) \cdot \log(nM))$ where M is the largest absolute value of the input numbers. Hence, the running time is polynomial in the input size. □

The algorithm provides a generic tool that can be applied to various problems, whenever the corresponding single-objective optimization problem can be solved efficiently in the presence of both positive and negative numbers. For instance, since perfect weighted matchings can be computed in time $\mathcal{O}((m + n\log n)n)$ (Gabow 1990) the lemma immediately implies the following.

Corollary 4 *A perfect maximum density matching can be computed in* $\mathcal{O}((m + n\log n)n\log(nM))$ *time.*

Next, we show that a maximum density subtree with k leaves can be computed in a tree in polynomial time, again using the fact that we can solve the underlying single-objective optimization problem efficiently.

Theorem 7 *Given a tree* $T = (V, E)$, *a maximum density subtree with exactly* k *leaves can be computed in time* $\mathcal{O}(k^2 n\log(nM))$ *where* M *denotes the largest input number.*

Proof The proof exploits a combination of the parametric search technique and dynamic programming. By applying the parametric search we reduce the problem to finding—for various values of $\theta \in \mathbb{R}$—a longest subtree of T with k leaves, where the new length of each edge e is given by $a_e - \theta b_e$. For an edge $e = \{u, v\}$ we denote this new length by $\lambda(u, v) = a_e - \theta b_e$.

In order to do compute the longest subtree of T with k leaves, we root the tree in some vertex $r \in V$. For a vertex $v \in V$ we denote the number of children of v by $n(v)$ and we denote the children by $u_1^v \ldots, v_{n(v)}^v$.

Suppose we are given an optimal solution T^* for this problem. Let v^* be the topmost vertex in T^* with respect to the rooting. Then either v^* is a leaf in T^* and there are only $k - 1$ leaves of T^* in the subtrees rooted in the children of v or v^* is an internal vertex with k leaves in the subtrees rooted in the children of v^*. Let $u \in V$ be a vertex of T and let T_u denote the subtree of T rooted in u. Let T'_u be a subtree of T_u containing u. Further, let T^*_u denote the subtree of T^* that is contained in T'_u and let k' denote the number of leaves of T^* contained in T'_u. Then, clearly T^*_u is the longest subtree in T'_u rooted in u with k' leaves. We use this observation to decompose the problem into smaller sub-problems.

Let $v \in V$, $1 \leq i \leq k-1$ and $1 \leq j \leq n(v)$. Then we denote by $\Lambda(v,i,j)$ the length of a longest tree $\widehat{T}$ with i leaves, where v does not count as a leaf, such that $\widehat{T}$ is contained in the tree induced by v and the subtrees rooted in its children $u^v_1, \dots, u^v_j$. We can compute these values from the following equation

$$\Lambda(v,i,j+1) = \max \begin{Bmatrix} \Lambda(v,i,j), \\ \Lambda(v,i-1,j) + \lambda(v,u_{j+1}), \\ \max_{1\leq t\leq i}\{\Lambda(v,i-t,j) + \Lambda(u_{j+1},t,n(u_{j+1}))\} + \lambda(v,u_{j+1}) \end{Bmatrix}$$

in a bottom-up manner on the tree, using $\Lambda(v,0,0) = 0$. This can be done in $\mathcal{O}(k^2 n)$ time.

Further, we denote by $\overline{\Lambda}(v,i,j)$ the length of a longest tree $\widehat{T}$ with i leaves, where v does not count as a leaf, such that $\widehat{T}$ is contained in the tree induced by v and exactly one subtree rooted in some child u^v_r of v, where $j \leq r \leq n(v)$. These values can be computed in $\mathcal{O}(kn)$ time from the following equation using the values computed in the previous step

$$\overline{\Lambda}(v,i,j) = \max_{j\leq r\leq n(v)} \bigl\{\Lambda\bigl(u_r,i,n(u_r)\bigr) + \lambda(v,u_r)\bigr\}.$$

Clearly, a tree with k leaves can only be found in a subtree of T with at least $k-1$ leaves. For each vertex v such that the tree T_v rooted in v contains at least $k-1$ leaves, we compute the longest subtree with k leaves, denoted by $\Lambda^*(v)$, as follows

$$\Lambda^*(v) = \max \begin{Bmatrix} \overline{\Lambda}(v,k-1,1), \\ \max_{1\leq t\leq k-1, 1\leq r\leq n(v)}\{\Lambda(v,k-t,r) + \overline{\Lambda}(v,t,r+1)\} \end{Bmatrix}.$$

As mentioned earlier, this equation reflects the fact that v is either a leaf itself or an internal vertex, in which case v must have at least two children in T_v. We achieve this by "guessing" an index r such that the computed tree contains at least on child in $u^v_1, \dots, u^v_r$ and one child in $u^v_{r+1}, \dots, u^v_{n(v)}$. Again, these values can be computed in $\mathcal{O}(kn)$ time. Finally, we return the solution corresponding to the maximum value $\Lambda^*(v)$ over all $v \in V$ with at least $k-1$ leaves in the subtree T_v. Using Lemma 5 the total time complexity is $\mathcal{O}(k^2 n \log(nM))$. □

4.2 Maximum density subgraphs spanning a subset of the vertices

In this section we consider maximum density subgraph problems with Steiner constraints. Given a graph $G = (V,E)$ and a set of *terminals* $S \subseteq V$ the MAXIMUM DENSITY STEINER SUBGRAPH problem asks for a maximum density subgraph H containing all vertices in S. First we show that this problem NP-hard and inapproximable unless P = NP, even if the pattern is a path and there is only one terminal.

Theorem 8 *The* Maximum Density Steiner Subgraph *problem is* NP*-hard, even if all weights are positive, all numbers are chosen from two distinct values, there is only one terminal and the pattern is a path. Furthermore, unless* P = NP, *this problem can not be approximated within a constant factor in polynomial time under the same conditions.*

We prove this theorem by a two-step reduction from the Longest Path problem. Given a graph $G = (V, E)$ the Longest Path problem is to compute a path with maximum length, where the length is given by the number of edges on this path. This problem is NP-hard (Garey and Johnson 1979) and cannot be approximated within a constant factor unless P = NP due to Karger et al. (1997). First, we show that this problem remains NP-hard and inapproximable if we require that the path starts in a predefined vertex $r \in V$. We refer to this problem as the Rooted Longest Path problem and we refer to r as the root.

Lemma 6 *The* Rooted Longest Path *problem is* NP*-hard and cannot be approximated within a constant factor unless* P = NP.

Proof Let $\mathcal{A}$ be an algorithm that approximates the Rooted Longest Path problem within a factor r. Then we immediately obtain an algorithm $\mathcal{A}'$ approximating the longest path by running $\mathcal{A}$ with root v once for each $v \in V$ and returning the maximum of these values. Clearly, $\mathcal{A}'$ approximates the longest path within a factor of r. The claim then follows from the results of Karger et al. (1997). □

Proof of Theorem 8 We make a reduction from the Rooted Longest Path problem. Assume we are given an instance $I = (G, r)$ of the Rooted Longest Path problem, where $G = (V, E)$ is a graph and $r \in V$ is the root. We construct a new instance $I' = (G', S)$ of the Maximum Density Steiner Subgraph problem as follows. We let $G' = (V', E')$ such that $V' = V \cup \{x\}$ for some new vertex $x \notin V$ and we set $E' = E \cup \{\{x, r\}\}$ and $S = \{x\}$. Further, let $M := n^2 + 1$. We set $w_e = 1$ and $\ell_e = M$ for $e = \{x, r\}$ and we set $w_e = M$ and $\ell_e = 1$ for all $e \in E$.

We claim that there is a path in G' rooted in x with density at least θ if and only if there is a path in G with length at least $\lceil \theta - 1 \rceil$ rooted in r. Note that a path in G' of length $i + 1$ rooted in x has density $\theta_i = (M + iM)/(M + i)$ for $i \geq 0$. Since θ_i is monotonically increasing in i and we have $i < \theta_i < i + 1$ for $i \leq n$ and $M \geq n^2$ it follows that $\lceil \theta_i - 1 \rceil = i$ and, hence, the claim holds.

Suppose that $\mathcal{A}$ is an approximation algorithm that approximates the Maximum Density Steiner Subgraph problem for path patterns within a factor $r_{\mathcal{A}}$. We show that we can use $\mathcal{A}$ to approximate the rooted longest path problem within a factor $3r_{\mathcal{A}}$. If r is isolated in G, we return r as a longest path, which is an optimal solution in this case. Otherwise, let e be an arbitrary edge incident to r in G. For a given instance $I = (G, r)$ let $P_{\mathcal{A}}$ be the path computed by $\mathcal{A}$ for the instance $I' = (G', S)$ constructed from I as described above and let $\theta_{\mathcal{A}}$ be its density. If $\theta_{\mathcal{A}} \geq 2M/(M + 1)$ we return $P_{\mathcal{A}}$, otherwise we return the single edge e, which together with the edge $\{x, r\}$, forms a path of length two with density $2M/(M + 1)$ in G'.

In the following, let opt_I denote the longest path in G rooted in r and let $\text{opt}_{I'}$ denote the maximum density Steiner path in G'. Further, let $\text{apx}_{I'}$ denote the density of the approximation as described above. Note that since $2M/(M+1) > 3/2$ for $n > 1$ we have $\mathrm{apx}_{I'} > 3/2$. It follows that

$$\begin{aligned}\frac{\mathrm{opt}_I}{\mathrm{apx}_{I'}} &= \frac{\lceil \mathrm{opt}_{I'} - 1 \rceil}{\lceil \mathrm{apx}_{I'} - 1 \rceil} \\ &\le \frac{\mathrm{opt}_{I'}}{\mathrm{apx}_{I'} - 1} \\ &\le \frac{1}{1 - 1/\mathrm{apx}_{I'}} \cdot \frac{\mathrm{opt}_{I'}}{\mathrm{apx}_{I'}} \\ &< \frac{1}{1 - 2/3} r_{\mathcal{A}} \\ &= 3 \cdot r_{\mathcal{A}}.\end{aligned}$$

The claim then follows from Lemma 6. □

Although the Maximum Density Steiner Subgraph problem is NP-hard and unlikely to be approximable if the pattern is a path, we may still be able to obtain fixed-parameter tractable algorithms. First, we show that it is unlikely that the general problem is FPT when parameterized by the number of vertices in the solution, when we have no constraint on the feasible patterns.

Theorem 9 Maximum Density Steiner Subgraph *is $W[1]$-hard when parameterized by the number of vertices of the Steiner subgraph, even if S contains only one vertex.*

Proof We prove the theorem by reduction from the $W[1]$-hard problem k-Clique (Downey and Fellows 1995). Given an instance of k-Clique, that is, a graph $G = (V, E)$, we wish to decide if G has a clique of size at least k. We transform this into an instance of Maximum Density Steiner Subgraph as follows. We construct a graph G' by adding a new vertex x to G that is connected to all vertices in V. We set $w_{vw} = \ell_{vw} = 1$ for all $vw \in E$ and we set $w_{xv} = 0$ and $\ell_{xy} = 1$ for all $v \in V$. Further, we set $S = \{x\}$. Clearly, there is a clique of size k with $m = \binom{k}{2}$ edges in G if and only if G' has a subgraph H with $x \in H$ and density at least $m/(m+1)$. □

While the Maximum Density Steiner Subgraph problem is W[1]-hard on general graphs, it turns out to be FPT on planar graphs.

Theorem 10 Maximum Density Steiner Subgraph *is FPT on planar graphs when parameterized by the number of vertices of the subgraph.*

Proof Let G be a planar graph and let $S \subseteq V$ be a non-empty set of terminals. Since we are looking for a connected subgraph and since $S \neq \emptyset$, it suffices to consider the subgraph G' consisting of the $(k-1)$-neighborhood of some vertex $s \in S$. This graph is $(k-1)$-outerplanar and has radius at most $k-1$. Hence, by a result of Robertson and Seymour (1984), G' has treewidth bounded by $3k-2$. Note, that a path is a simple, connected graph that does not contain a triangle as a minor. Hence, using a modification of the algorithm proposed in Theorem 4 in combination with Theorem 5, we can compute a subgraph of G' with maximum density in FPT time. □

In contrast to the W[1]-hardness of the Maximum Density Steiner Subgraph problem without constraints on the pattern, we show that the problem is FPT when parameterized by the number of vertices if the pattern is a path. In order to obtain this result we use the parametric search technique introduced in Sect. 4.1 in combination with Color Coding (Alon et al. 1995). Color Coding was introduced by Alon et al. (1995) as a method for finding subgraphs of bounded size in arbitrary graphs in FPT time. It can be used to find paths and cycles in expected time $2^{\mathcal{O}(k)}m$. The algorithms are randomized by construction, but can be derandomized. We can directly apply Color Coding to find a maximum density Steiner path with at most k vertices with high probability (WHP) in time $2^{\mathcal{O}(k)}m\log(nM)$. We can slightly improve on the standard algorithm by coloring the terminals in a deterministic way, thus, using only $k-s$ random colors and thereby improving the probability of obtaining an optimal solution. In the following we further improve on this result when the number of Steiner vertices is large compared to k.

Theorem 11 *Given a set S of s terminals a maximum density Steiner path with k vertices can be computed WHP in time $\mathcal{O}((2^{k-s}m + 3^{k-s})s^2\log(nM))$.*

Proof Since we apply the parametric search technique, we need to find, for various values of θ, a longest path with respect to the edge-lengths $\lambda(e) := a_e - \theta b_e$ for all $e \in E$. Suppose we are given a random coloring of the vertices in $V \setminus S$ with $k-s$ colors using some color set C and we wish to find a longest *colorful* Steiner path for S. A Steiner path is called colorful, if it contains each color exactly once. Note, that this implies that the path is simple, that is, it does not contain multiple copies of any vertex. Our approach is based on a decomposition of any optimal colorful Steiner path P for S into three optimal subpaths P_1, P_2 and P_3 as illustrated in Fig. 5 such that the following holds:

(i) P_2 starts and ends at terminals s_1 and s_2, respectively, and contains all terminals
(ii) P_1 and P_2 end at terminals s_1 and s_2, respectively, and do not contain any other terminals.

In order to compute P_2, we further decompose this path into fragments each starting and ending with a terminal and containing no other terminals apart from the end-vertices. Let $c(v)$ denote the color of vertex v. For each vertex $v \in V \setminus S$, $x \in S$ and each set of colors $C' \subseteq C$ with $c(v) \in C'$ we compute the maximum length $L(x, v, C')$ of any colorful path from x to v using all colors in C'. Similarly, we compute for each $z \in V$ the length $\widehat{L}(z, C')$ of the longest path ending in z using

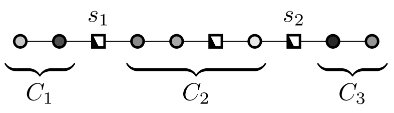

Fig. 5 Decomposition of an optimal colorful Steiner path. Terminals are depicted as boxes

all colors in C'. We compute these values by dynamic programming in $\mathcal{O}(2^{k-s} \cdot s \cdot m)$ time using the equations

$$L\big(x, v, C'\big) = \max_{w \in N(v) \setminus S} \big\{L\big(x, w, C' \setminus \{c(v)\}\big) + \lambda(v, w)\big\}$$

$$\widehat{L}\big(z, C'\big) = \max_{w \in N(z) \setminus S} \big\{\widehat{L}\big(w, C' \setminus \{c(v)\}\big) + \lambda(w, z)\big\}.$$

For each $x, y \in S$ and each set of colors $C' \subseteq C$ we subsequently compute the maximum length $L(x, y, C')$ of any path from x to y using all colors in C' as well as for all $z \in V$ the maximum length $\widehat{L}(z, C')$ of any path ending in z using all colors in C' by

$$L\big(x, y, C'\big) = \max_{w \in N(y) \setminus S} \big\{L\big(x, w, C'\big) + \lambda(w, y)\big\}.$$

in time $\mathcal{O}(2^{k-s} \cdot s^2 \cdot m)$. Implicitly, we obtain a combinatorial description of the longest paths starting and ending at terminals as a complete multigraph $G_S = (S, M)$ on the set of terminals in which each edge is annotated with its length and the set of colors used to attain this length. The weighted multiset of edges in our multigraph $G_S = (S, M)$ is defined by

$$M = \big\{\big(xy, C', L\big(x, y, C'\big)\big) \mid x, y \in S, C' \subseteq C\big\}.$$

Then we compute for each vertex $x \in S$ and each subset $X \subseteq S$ with $x \in X$ and each set of colors $C' \subseteq C$ the length $\widetilde{L}(x, X, C')$ of the longest path in G_S ending in x using all vertices in X and all colors in C' using the equation

$$\widetilde{L}\big(x, X, C'\big) = \max_{y \in S, C'' \subseteq C'} \big\{\widetilde{L}\big(y, X \setminus \{x\}, C' \setminus C''\big) + L\big(x, y, C''\big)\big\}$$

where $L(x, \{x\}, C') = \widehat{L}(x, C')$. In order to do this efficiently, we consider all partitions of C into three sets $C \setminus C'$, C'' and $C' \setminus C''$. The number of these partitions is at most 3^{k-s}. Hence, the computation can be performed in time $\mathcal{O}(3^{k-s})s^2$. The optimal solution can then be computed as

$$\max_{x \in S, C' \subseteq C} \big\{\widetilde{L}\big(x, S, C'\big) + \widehat{L}\big(x, C \setminus C'\big)\big\} \tag{7}$$

in time $\mathcal{O}(2^{k-s}s)$. Hence, the dynamic programming for fixed θ takes $\mathcal{O}(2^{k-s}s^2m + 3^{k-s}s^2)$ time and the problem can be solved in time $\mathcal{O}((2^{k-s}m + 3^{k-s})s^2 \log(nM))$.

The probability that a path of length at most k interconnecting a set of s vertices is colorful is given by $p(k, s) = (k - s)!/(k - s)^{(k-s)} > \sqrt{2\pi(k - s)}e^{-(k-s)}$. Hence, a colorful path can be found with probability $\leq \varepsilon$ if the number of trials is at least $t_\varepsilon(k, s) = \frac{\ln \varepsilon}{\ln(1-p(k,s))} = |\ln \varepsilon| \cdot \mathcal{O}(e^{k-s})$. □

We conclude this section by showing that we cannot hope to extend this result to more general patterns.

Theorem 12 *It is* NP*-hard to decide whether there is a Steiner tree with at most k vertices and density at least θ, even if we allow only one terminal.*

Proof We show NP-hardness by reduction from the NP-hard k-MST problem. Given an edge-weighted graph G, a non-negative integer k and a weight W, the k-MST problem is to decide whether there is a tree spanning at least k vertices with weight at least W. This problem has been shown to be NP-hard by Ravi et al. (1996) and it obviously remains NP-hard if we require the solution to contain exactly k vertices.

We observe that deciding whether there is a tree with k vertices and density at least θ is equivalent to deciding whether there is a tree with k vertices and length *at most* 0 where the length of each edge is given by $\theta b_e - a_e$. To see this, note that

$$\frac{\sum_{e\in E'} a_e}{\sum_{e\in E'} b_e} \geq \theta \quad \Leftrightarrow \quad \sum_{e\in E'} (\theta b_e - a_e) \leq 0$$

by simple equivalent transformations.

Assume we are given an instance of k-MST, that is, a graph $G = (V, E)$ and we wish to decide if there is a tree with k vertices and length at most θ where the edge-lengths are integral. We transform this into a set of $|V|$ k-Maximum Density Steiner Tree problems G_v such that G is solvable if and only if at least one of the new instances G_v is solvable. We choose $G_v = (V \cup \{x\}, E \cup \{xv\})$ for some new vertex x. Further, we choose the $b_e \equiv 1$ and $a_e := \ell_e - \theta$ and $a_{xv} = 2\theta$ and $S = \{x\}$. Hence, $\theta b_e - a_e = \ell_e$, $\theta b_{xv} - a_{xv} = -\theta$ and each solution contains the edge sx.

By the above observations there is a tree with k vertices and length at most θ if and only if there is a tree with $k+1$ vertices including x with length at most 0, where edge lengths are defined by $\theta b_e - a_e$. The latter is equivalent to deciding whether there is a tree with k vertices and density at least θ. □

5 Conclusion and outlook

We have investigated the complexity of a framework for 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 hosts and patterns. Like many multi-objective optimization problems, the problems we considered turned out to be NP-hard in general. Nevertheless, we presented efficient algorithms for restricted variants of the problem as well as an FPTAS for a relaxed variant of the problem.

While the bi-constrained density maximization problem is already NP-hard if the host is an outerplanar graph, that is, a graph with treewidth 2, and the pattern is a path, we were able to obtain a pseudo-polynomial time algorithm for the case when the host has bounded tree-width and the pattern is a graph from a minor-closed family of graphs. In some sense, this is the best we can hope for, since the existence of a polynomial time algorithm for this case would imply P = NP. Additionally, we presented an efficient algorithm for computing the maximum density path in a tree and in a graph that can be turned into a tree by removing only a fixed number of edges, respectively. Again, this is in some sense the best we can hope for. While graphs that can be turned into a tree by removing at most k edges have treewidth bounded by $k+1$, we cannot hope to extend this result to the class of graphs with treewidth k

unless P = NP. Interestingly, this result is obtained by applying geometric arguments to a problem that does not have any geometric context at first glance. Further, we showed that a relaxed variant of the density maximization problem admits a FPTAS, if we allow violating the upper bound on the length of the sought solution at some extra penalty. That is, for this relaxed variant, we can get rid of the pseudo-polynomiality at the cost of a loss of exactness.

Finally, we considered the density maximization problem in the presence of structural constraints on the vertex sets. Although some density maximization problems involving constraints on the vertex set of the graph, such as the problem of computing perfect matchings, can be solved efficiently by parametric search in the density-space, the general maximum density Steiner subgraph problem is NP-hard and inapproximable. Even worse, this problem is W[1]-hard when parameterized by the number of vertices of the sought solution. In contrast to this, we showed that the special case when the pattern is a path is fixed-parameter tractable when parameterized by this number. Again, we cannot hope to generalize this problem very much, since it turns out to be NP-hard already for general trees.

Open problems Due to the hard nature of both the bi-constrained density maximization problem and the relaxed density maximization problem, it seems unlikely that there exist efficient exact algorithms for all instances of the problem. Therefore we studied approximation algorithms and FPT algorithms. Whereas we only studied parameterizations with respect to treewidth and the number of vertices of the pattern, there may be other interesting parameters of the problem. Further, it seems to be an interesting challenge to devise efficient and effective heuristics for the problem. Another interesting problem concerns the question whether there is a meaningful class of graphs admitting the efficient computation of a maximum density path between the class of graphs with bounded tree-width and the class of graphs that can be turned into a tree by removing a fixed number of edges. Finally, it seems to be worthwhile to study variants of the problem in which we know more about the weights and the lengths of the edges, for instance, if we consider the problem on geometric graphs and the length corresponds to the Euclidean distance whereas the weight is proportional to the number of points in the vicinity of the edge, say. This problem variant can be motivated from network construction problems such as the construction of a new power supply system where each edge in the graph corresponds to a possible route for the network and points in the vicinity of an edge correspond to users that can profit from this line.

References

Alon N, Yuster R, Zwick U (1995) Color-coding. J ACM 42(4):844–856

Bálint V (2003) The non-approximability of bicriteria network design problems. J Discrete Algorithms 1:339–355

Bodlaender HL (1993) 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. ACM, New York, pp 226–234

Chandrasekaran R (1977) Minimal ratio spanning trees. Networks 7(4):335–342

Chinchuluun A, Pardalos P (2007) A survey of recent developments in multiobjective optimization. Ann Oper Res 154:29–50

Chung KM, Lu HI (2005) An optimal algorithm for the maximum-density segment problem. SIAM J Comput 34(2):373–387

Downey RG, Fellows MR (1995) Fixed-parameter tractability and completeness II: On completeness for W[1]. Theor Comput Sci 141(1–2):109–131

Dreyfus S, Wagner R (1971) The Steiner problem in graphs. Networks 1(3):195–207

Eppstein D (1995) Subgraph isomorphism in planar graphs and related problems. In: Proc 6th ann ACM-SIAM sympos disc alg SIAM, Philadelphia, pp 632–640

Gabow HN (1990) Data structures for weighted matching and nearest common ancestors with linking. In: Proceedings of the first annual ACM-SIAM symposium on discrete algorithms, SODA'90. Society for Industrial and Applied Mathematics, Philadelphia, pp 434–443

Garey MR, Johnson DS (1979) Computers and intractability. A guide to the theory of NP-completeness. Freeman, New York

Goldwasser MH, Kao MY, Lu HI (2005) Linear-time algorithms for computing maximum-density sequence segments with bioinformatics applications. J Comput Syst Sci 70(2):128–144

Hsieh SY, Cheng CS (2008) Finding a maximum-density path in a tree under the weight and length constraints. Inf Process Lett 105(5):202–205

Hsieh SY, Chou TY (2005) Finding a weight-constrained maximum-density subtree in a tree. In: Algorithms and computation. LNCS, vol 3827. Springer, Berlin, pp 944–953

Inman RB (1966) A denaturation map of the lambda phage DNA molecule determined by electron microscopy. J Mol Biol 18(3):464–476

Karger D, Motwani R, Ramkumar G (1997) On approximating the longest path in a graph. Algorithmica 18:82–98

Kloks T (1994) Treewidth, Computations and approximations. LNCS. Springer, Berlin

Lau HC, Ngo TH, Nguyen BN (2006) Finding a length-constrained maximum-sum or maximum-density subtree and its application to logistics. Discrete Optim 3(4):385–391

Lee DT, Lin TC, Lu HI (2009) Fast algorithms for the density finding problem. Algorithmica 53(3):298–313

Lin YL, Jiang T, Chao KM (2002) Efficient algorithms for locating the length-constrained heaviest segments with applications to biomolecular sequence analysis. J Comput Syst Sci 65(3):570–586

Liu HF, Chao KM (2008) 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

Lokshtanov D (2009) New methods in parameterized algorithms and complexity. PhD thesis, University of Bergen Norway

Macaya G, Thiery JP, Bernardi G (1976) An approach to the organization of eukaryotic genomes at a macromolecular level. J Mol Biol 108(1):237–254

Marathe MV, Ravi R, Sundaram R, Ravi SS, Rosenkrantz DJ, Hunt HB (1998) Bicriteria network design problems. J Algorithms 28(1):142–171

McCreight EM (1985) Priority search trees. SIAM J Comput 14(2):257–276

Overmars MH, van Leeuwen J (1981) Maintenance of configurations in the plane. J Comput Syst Sci 23(2):166–204

Ravi R, Sundaram R, Marathe MV, Rosenkrantz DJ, Ravi SS (1996) Spanning trees—short or small. SIAM J Discrete Math 9:178–200

Robertson N, Seymour PD (1984) Graph minors. iii. Planar tree-width. J Comb Theory, Ser B 36(1):49–64

Robertson N, Seymour PD (1995) Graph minors. XIII. The disjoint paths problem. J Comb Theory, Ser B 63(1):65–110

Robins G, Zelikovsky A (2000) Improved steiner tree approximation in graphs. In: Proceedings of the eleventh annual ACM–SIAM symposium on discrete algorithms, SODA'00. Society for Industrial and Applied Mathematics, Philadelphia, pp 770–779

Schuurman P, Woeginger G (2011) Approximation schemes—a tutorial. URL www.win.tue.nl/~gwoegi/papers/ptas.pdf. Preliminary version of a chapter in the book Lectures on Scheduling, to appear

Wu BY (2009) An optimal algorithm for the maximum-density path in a tree. Inf Process Lett 109(17):975–979

Wu BY, Chao KM, Tang CY (1999) An efficient algorithm for the length-constrained heaviest path problem on a tree. Inf Process Lett 69(2):63–67