Algorithms for graphs with small octopus ⋆

Fedor V. Fomin

Department of Informatics, University of Bergen N-5020 Bergen, Norway.
Email:`fomin@ii.uib.no`

Dieter Kratsch

Université de Metz, Laboratoire d'Informatique Théorique et Appliquée,
57045 Metz Cedex 01, France. Email: `kratsch@sciences.univ-metz.fr`

Haiko Müller

School of Computing, University of Leeds, Leeds, LS2 9JT, United Kingdom.
Email: `hm@comp.leeds.ac.uk`

Abstract

A d-octopus of a graph $G = (V, E)$ is a subgraph $T = (W, F)$ of G such that W is a dominating set of G, and T is the union of d (not necessarily disjoint) shortest paths of G that have one endpoint in common.

First we study the complexity of finding and approximating a d-octopus of a graph. Then we show that for some NP-complete graph problems that are hard to approximate in general there are efficient approximation algorithms with worst case performance ratio $c \cdot d$ for some small constant $c > 0$ (depending on the problem) assuming that the input graph G is given together with a d-octopus of G. For example, there is a linear-time algorithm to approximate the bandwidth of a graph within a factor of $8d$. Furthermore the minimum number of subsets in a partition of the vertex set of a graph into clusters of diameter at most k can be approximated in linear time within a factor of $3d$ (for $k = 2$) and $2d$ (for $k \geq 3$).

Finally, we show that there are $O(n^{7d+2})$ time algorithms to compute a minimum cardinality dominating set respectively total dominating set for graphs having a d-octopus.

Key words: approximation algorithm, graph algorithm, bandwidth, clustering, λ-coloring, bilateral orientation, domination.

⋆ The work of FVF was done in part while he was at the Centro de Modelamiento

1 Introduction

The study of graph algorithms is often motivated by practical applications from various areas. Since a large part of the practically and theoretically important graph problems is NP-complete (for graphs in general), several main approaches to intractable graph problems have been developed over the years. One of the most popular approaches is to relax the requirement to compute an optimal solution of an optimization problem; thus the goal will be to find a (polynomial time) approximation algorithm (see e.g. [3]).

In this paper our goal will be to find fast approximation algorithms with constant worst case ratio. This will be achieved by allowing our approximation algorithms access to a certain advice called a d-octopus of the graph. Among others we consider the k-clustering problem which appears quite naturally in the context of transport, warehousing and networking, and the bandwidth minimization problem which plays an important role in computational linear algebra.

Another approach to intractable graph problems requires that the class of input instances is severely constrained and that for these instances there is an efficient algorithm using special properties of the restricted instances. Among others, this leads to the study of the structural properties of special graph classes and algorithms on special graph classes (see [7]). A popular approach in this directions is the design of algorithms for graphs of bounded treewidth because many combinatorial optimization problems on graphs can be solved in polynomial and often even linear time on graphs of bounded treewidth (see e.g. [4]). Quite similar, bounded bandwidth allows a considerable speed-up for many matrix-operations and, as we shall show, graphs with small octopus allow a constant-factor linear-time approximation algorithm for the bandwidth problem. More general, graphs with small octopus allow efficient constant-factor approximation algorithms on a variety of NP-complete and often hard to approximate graph problems. However we must admit that our concept does not apply to such a wide range of problems as the treewidth approach.

The recent development of complexity theory highlights a class of problems that are hard to approximate even for very restricted graph classes. A well-known example of such a problem is the BANDWIDTH minimization problem. As it was shown by Unger in [37], unless P=NP there is no (polynomial time) approximation algorithm with constant performance ratio for BANDWIDTH

Matemático, Universidad de Chile and UMR 2071-CNRS, supported by FONDAP and while he was a visiting postdoc at DIMATIA-ITI partially supported by GAČR 201/99/0242 and by the Ministry of Education of the Czech Republic as project LN00A056.

even when restricted to a small subclass of trees, namely caterpillars with hair length three. Hence even for graphs with treewidth one it is unlikely that there is a polynomial time algorithm to approximate the bandwidth within a constant factor. The hardness result implies a natural question—what kind of structure in a graph guarantees the existence of an approximation algorithm for BANDWIDTH having a constant performance ratio?

To answer this question we introduce the concept of a *d-octopus* and show that the existence of a small octopus can be important for approximation algorithms. Fortunately, a small octopus in a graph can be used not only for BANDWIDTH approximation but also for designing approximation algorithms for a variety of other optimization problems. We provide several examples of such optimization problems. Almost all these problems have the following properties: They are NP-complete even for very restricted subclasses of graphs with 1-octopus and unless P=NP there is no (polynomial time) approximation algorithm with constant performance ratio for these problems on general graphs.

The paper is organized as follows. In Section 2 we give some preliminaries and introduce the concept of a *d-octopus* which is a generalization of the notion of a dominating shortest path in a graph. (For the definition of a d-octopus see Definition 1.) In Section 3 we prove a number of complexity results concerning the computation and the approximation of an octopus in a graph. In particular, we present a polynomial time $\ln n$-approximation algorithm to compute the minimum d for which a graph has a d-octopus. In some sense this result is best possible since we also prove that there is a constant $c > 0$ such that there is no polynomial time algorithm to approximate the minimum d for which a graph has a d-octopus within a factor of $c \log n$ unless P=NP.

In Section 4 we consider approximation algorithms for different optimization problems on graphs with small octopuses. In particular, in Subsection 4.1 we consider the k-clustering problem ($k \geq 2$ a fixed integer) which is known to be hard to approximate on graphs in general. We provide linear time approximation algorithms for the k-clustering problem on graphs, where a d-octopus of the graph is supposed to be part of the input. Our algorithm finds a k-clustering within $3d$ times the optimum for $k = 2$, and it finds a k-clustering within $2d$ times the optimum for $k \geq 3$. In Subsection 4.2 we provide a linear-time algorithm to approximate the bandwidth of a graph within $8d$, assuming a d-octopus of the graph is part of the input. In Subsections 4.3, 4.4, 4.5 and 4.6 we present simple and fast approximation algorithms with worst case performance ratio $c \cdot d$ for some small constant $c > 0$ for a number of problems (λ-coloring, also known as radio-coloring, interval and chordal completions with the smallest max-degree, domino-treewidth and domino-pathwidth), where the constant c depends on the problem. Again all algorithms assume that a

d-octopus of the graph is part of the input.

In Section 5 we present exact polynomial time algorithms for the problems Domination and Total Domination for graphs with d-octopus (d a constant). These algorithms differ from all other algorithms presented in the paper in the way that they do not need a d-octopus to be part of the input and they do not even attempt to compute a d-octopus of the given graph. Finally in Section 6 we discuss the limits of our approach.

2 Preliminaries

Let $G = (V, E)$ be an undirected, simple (without loops and multiple edges) and finite graph with the vertex set V and the edge set E. Unless otherwise specified, n denotes the number of vertices of G. We denote by $G[W]$ the subgraph of G induced by $W \subseteq V$. For convenience, for a vertex x of G we write $G - x$ instead of $G[V \setminus \{x\}]$. Analogously, for a subset $X \subseteq V$ we write $G - X$ instead of $G[V \setminus X]$. For an edge $e \in E$, we denote by $G - e$ the graph with vertex set V and edge set $E \setminus \{e\}$.

The *maximum degree* of the graph G denoted by $\Delta(G)$ is the maximum degree of a vertex of G. The *(open) neighborhood* of a vertex v is $N(v) = \{u \in V : \{u, v\} \in E\}$ and the *closed neighborhood* of v is $N[v] = N(v) \cup \{v\}$. For a set $S \subseteq V$ we put $N[S] = \bigcup_{v \in S} N[v]$ and $N(S) = N[S] \setminus S$. A set $D \subseteq V$ is said to be a *dominating set* of $G = (V, E)$ if for every vertex $u \in V \setminus D$ there is a vertex $v \in D$ such that $\{u, v\} \in E$. Thus D is a dominating set iff $N[D] = V$. We also say that $A \subseteq V$ *dominates* $B \subseteq V$ in the graph $G = (V, E)$ if $B \subseteq N[A]$. A dominating set $D \subseteq V$ is said to be a *total dominating set* of $G = (V, E)$ if $G[D]$ has no isolated vertex. A dominating set $D \subseteq V$ is said to be a *connected dominating set* of $G = (V, E)$ if $G[D]$ is a connected graph. The minimum cardinality of a dominating set, total dominating set and connected dominating set, respectively, of a graph G is denoted by $\gamma(G)$, $\gamma_{\text{total}}(G)$ and $\gamma_{\text{conn}}(G)$, respectively.

The *distance* $d_G(u, v)$ between two vertices u and v of G is the length (number of edges) of a shortest path between u and v in the graph G. The *eccentricity* $\text{ecc}(v, G)$ of the vertex v in the graph $G = (V, E)$ is defined to be $\max\{d_G(v, w) : w \in V\}$. The *diameter* of a graph G, denoted by $\text{diam}(G)$, is defined to be the maximum distance between two vertices of G. Thus $\text{diam}(G) = \max\{\text{ecc}(v, G) : v \in V\}$.

An edge e of a connected graph G is called a *bridge* if $G - e$ is disconnected. A connected graph G is *bridgeless* if $G - e$ is connected for every edge e.

For all definitions and properties of special graph classes not given in the paper we refer to [7].

Now we give the definition of a d-octopus.

Definition 1 *A* d-octopus $T = (W, F)$ *of a graph* $G = (V, E)$ *is defined as a subgraph of* G *(i.e.* $W \subseteq V$ *and* $F \subseteq E$*) such that* W *is a dominating set of* G *and there are vertices* $r, l_1, l_2, \ldots, l_d$ *of* G *and for each* i*,* $1 \leq i \leq d$*, there is a shortest path* P_i *from* r *to* l_i *in* G*, such that* T *is the union of the paths* $P_1, P_2, \ldots, P_d$*.*
The common endpoint r *of the* d *shortest paths is called the* root *of the* d*-octopus* T*. Note that the paths* $P_1, P_2, \ldots, P_d$ *need not to be disjoint.*

Clearly every graph with a d-octopus (for some d) is connected. Furthermore every connected graph has a d-octopus (for some d), e.g. any shortest-path tree.

An algorithm is said to be *c-approximation* algorithm for a minimization problem P if for any instance of P it yields a solution whose value is at most c times the optimum.

3 Finding and approximating a d-octopus

For some graphs it is easy to find a small octopus. Here graphs of bounded asteroidal number should be mentioned, which are defined as follows. A set A of vertices of a graph G is an *asteroidal set* if for every vertex $a \in A$ there is one component of $G - N[a]$ containing all vertices of $A \setminus \{a\}$. The *asteroidal number* of a graph G is the maximum cardinality of an asteroidal set of G. Graphs of bounded asteroidal number are an extension of AT-free graphs. In fact the AT-free graphs (usually defined in a slightly different way) are exactly the graphs of asteroidal number at most two. Note that the AT-free graphs contain well-known graph classes like interval, permutation, cobipartite and cocomparability graphs.

All connected AT-free graphs have a *dominating pair*, i.e. two vertices such that the vertex set of every path between the two vertices is a dominating set. Such a dominating pair of a connected AT-free graph can be found in linear time [15]. Thus all connected AT-free graphs have a 1-octopus that can be constructed as a shortest path between the two vertices of a dominating pair in linear time.

Furthermore every connected graph G has a $(\mathrm{dt}(G)-1)$-octopus, where $\mathrm{dt}(G)$ denotes the dominating target number of G. (See [30] for the definition.) Since the asteroidal number of a graph is an upper bound for its dominating target number [30], every connected graph of asteroidal number at most k has a $(k-1)$-octopus. Such a $(k-1)$-octopus can be constructed in time $O(n^3)$ using an algorithm to construct a dominating target of size at most k. This algorithm can be obtained from [30].

From now on we consider graphs in general and we shall need a d-octopus as part of the input in all of our efficient constant factor approximation algorithms. Since the performance of our approximation algorithms for graphs with d-octopus is best for small d, it is important to find a small d-octopus in general graphs.

Theorem 2 *The problem 'Given a graph G and an integer d, decide whether G has a d-octopus' is NP-complete, and it remains NP-complete when restricted to split graphs and when restricted to bipartite graphs.*

PROOF.

We use a reduction from DOMINATING SET to d-OCTOPUS. Let $G = (V, E)$ be a graph, $d \in \mathbb{N}$ and suppose $V = \{v_1, \ldots, v_n\}$. $G' = (V', E')$ is constructed as follows:

$$\begin{aligned} V' &= \{v_1, \ldots, v_n, u_1, \ldots, u_n, a, b\} \text{ and} \\ E' &= \{\{v_i, v_j\} : 1 \leq i, j \leq n, i \neq j\} \cup \{\{v_i, u_j\} : v_i \in N[v_j]\} \cup \\ &\quad \{\{a, v_i\} : 1 \leq i \leq n\} \cup \{\{a, b\}\}. \end{aligned}$$

Notice that G' is a split graph since its vertex set can be partitioned into the clique $C = \{v_1, \ldots, v_n, a\}$ and the independent set $I = \{u_1, \ldots, u_n, b\}$. It is sufficient to show that G has a dominating set of size at most d iff G' has a d-octopus. Suppose $D \subseteq V$ is a dominating set of G. Then G' has a d-octopus T with root a such that T is the union of the shortest a, l-paths in G' (each consisting of a single edge) for all $l \in D$. Suppose $T = (W, F)$ is a d-octopus of G'. W.l.o.g. we may assume $a \in W$ and $W \setminus \{a\} \subseteq V$. Otherwise if $b \in W$ or $u_i \in W$ each such vertex can simply be removed from T and we obtain another d-octopus. Since $V \cup \{a\}$ is a clique in G', $W \setminus \{a\}$ is a dominating set in G', and thus $W \setminus \{a\}$ is a dominating set of size d in G.

The proof for bipartite graphs uses a similar construction (in E' we omit the edges $\{v_i, v_j\}$) and follows the same lines. $\square$

For the same reason it is of interest to know whether the decision problem (taken d as parameter) is fixed parameter tractable (see [17]). On the positive side, it turns out that the problem is polynomial time solvable for any fixed d. For example, a 1-octopus of a given graph can be found in time $O(n^5)$, if the graph has one.

Our algorithm is based on a technique to compute a domination-type structure of a graph by dynamic programming through its breadth-first-search levels (see [32]).

Theorem 3 *For any fixed d, there is an $O(n^{3d+3})$ time algorithm to decide whether a given graph has a d-octopus. On the other hand, the problem is W[2]-hard and thus unlikely to be fixed parameter tractable.*

PROOF. To show the first part of the theorem we describe our algorithm, which verifies for every vertex $v \in V$ whether the input graph $G = (V, E)$ has a d-octopus T with root v. (Note that for $d' < d$ every d'-octopus is a d-octopus by definition.)

To decide whether G has a d-octopus with root v, the algorithm executes a breadth-first-search starting at vertex v. Hence the vertex set of G is partitioned into the levels $L_i = \{w \in V : d_G(v, w) = i\}$, $0 \leq i \leq t$. For a fixed d-octopus let A_i be the set of its vertices in level L_i. By m_w we denote the number of paths in our octopus containing vertex w. Let $B_i = \{(w, m_w) : w \in A_i\}$ and $C_i = \{w_j : w \in A_i \text{ and } 1 \leq j \leq m_w\}$. For consistency we set $A_{t+1} = B_{t+1} = C_{t+1} = \varnothing$. The use of C_i-sets is to represent the paths of the caterpillar, which may have vertices in common, by vertex-disjoint paths in the auxiliary graphs G_i defined below. In the algorithm we will store the B_i-sets only. Note that it is fairly easy to reconstruct A_i and C_i from B_i. Since we deal with a d-octopus we have

$$|C_i| = \sum\{m_w : w \in A_i\} \leq d\,. \tag{1}$$

We consider the bipartite graph $G_i = (C_{i-1}, C_i, E_i)$ with $E_i = \{\{u_k, w_j\} : \{u, w\} \in E\}$. Then

$$G_i \text{ has a matching of size } |C_i| \tag{2}$$

because every vertex $w \in A_i$ is linked by m_w paths from the octopus to predecessors in A_{i-1}. Especially, all paths extend from L_{i-1} to L_i if and only if G_i has a perfect matching. More general, exactly $|C_{i-1}| - |C_i|$ paths of our octopus end at level $i - 1$. Finally we know

$$L_{i-1} \subseteq N(A_{i-2}) \cup N[A_{i-1}] \cup N(A_i) \tag{3}$$

because the vertices of the octopus form a dominating set in G.

On the other hand,the existence of a sequence $(B_0, B_1, \ldots B_t)$ with $B_0 = \{(v, d)\}$ fulfilling the matching property (2) and the domination property (3) for $1 \leq i \leq t$ ensures that G has a d-octopus with root v. Note that $B_0 = \{(v,d)\}$ and the matching property (2) inductively imply property (1). Now an i-partial solution ($1 \leq i \leq t$) is a pair (B_{i-1}, B_i) such that

- for $i = 1$ we have $B_0 = \{(v,d)\}$ and $\sum\{m_w : (w, m_w) \in B_1\} \leq d$,
- for $1 < i < t$ there is a $(i-1)$-partial solution (B_{i-2}, B_{i-1}) such that properties (2) and (3) hold true, and
- for $i = t$ there is a $(t-1)$-partial solution (B_{t-2}, B_{t-1}) such that (2) holds for $i = t$ and (3) holds for $i = t, t+1$.

Basically the dynamic programming algorithm computes all i-partial solutions for $i = 1, 2, \ldots, t$. Note that there are at most n^d possible sets B_i for the level L_i by property (1). That is, in round i we compute up to n^{2d} potential i-partial solutions (B_{i-1}, B_i). For a given potential i-partial solution (B_{i-1}, B_i) this is done by checking for each of the at most n^d $(i-1)$-partial solutions (B_{i-2}, B_{i-1}) whether properties (2) and (3) are satisfied. Thus there are $O(n^{3d})$ triples (B_{i-2}, B_{i-1}, B_i) to verify. For each of them we can test property (2) in constant time $O(d^{2.5})$ (because $|C_i| \leq d$) and property (3) in linear time $O(dn)$. These tests are executed for $t < n$ levels. Thus the overall running time of the algorithm (for all vertices $v \in V$) is $O(n^{3d+3})$.

The second part of the theorem follows directly from the W[2]-hardness result for DOMINATING SET (shown e.g. in [17]) when using the construction given in the proof of Theorem 2. □

As is usually the case for dynamic programming algorithms, a pointer structure (here from each i-partial solution add a pointer to the $(i-1)$-partial solution from which it was obtained) can be used to compute a d-octopus, if the input graph G has a d-octopus, within the same timebound.

As a consequence of Theorem 3 we obtain

Corollary 4 *For every fixed $d \geq 1$, there is a polynomial time algorithm to decide whether a given graph G has a d-octopus, and to compute one if it does.*

We consider the complexity of approximating the optimization problem MINIMUM d-OCTOPUS ‘Given a graph G, find the minimum d for which G has a d-octopus’.

Theorem 5 *There is a constant $c > 0$ such that there is no polynomial time algorithm to approximate the minimum d for which a graph has a d-octopus within a factor of $c \log n$ unless P=NP.*

PROOF. We use the reduction from DOMINATING SET to d-OCTOPUS described in the proof of Theorem 2 and a result of [34] stating that there is some $c > 0$ such that the existence of an algorithm approximating DOMINATING SET within $c \log n$ would imply P=NP. $\square$

Finally we present an approximation algorithm for the problem MINIMUM d-OCTOPUS having worst case performance ratio $\ln n$. Our greedy algorithm is a modification of the well-known greedy algorithm for the MINIMUM SET COVER problem discovered independently by D.S. Johnson [28] and L. Lovász [33].

Theorem 6 *There is an $O(n^5)$ time algorithm to approximate the minimum d for which a given graph has a d-octopus within a factor of* $\ln n$.

PROOF. On first view our algorithm can be seen as the greedy set cover algorithm applied to n particular instances of MINIMUM SET COVER. For each vertex v of G, we use an instance $(V, \mathcal{M}_v)$ of the set cover problem to approximate a minimum d-octopus T with root v of the given graph $G = (V, E)$. For fixed vertex v, $\mathcal{M}_v$ is defined as follows: $A \in \mathcal{M}_v$ iff there is a vertex $u \in V$ and a shortest path P between u and v such that $A = N[V(P)]$. (Here $V(P)$ denotes the vertex set of the path P.)

The greedy set cover algorithm is executed on instance $(V, \mathcal{M}_v)$ for every vertex v. The solution of the greedy algorithm is an $\ln n$-approximation of the minimum set cover for each instance $(V, \mathcal{M}_v)$ [28,33]. Using the shortest u, v-paths in G corresponding to the sets in the solution of the algorithm one can easily obtain a d-octopus T of G as the union of all these paths. Note that d is exactly the number of sets in the solution of $(V, \mathcal{M}_v)$.

To see that our algorithm has performance ratio $\ln n$ assume that $T_{\min}$ is a $d_{\min}$-octopus of G with minimum possible value of d. Let $v_{\min}$ be the root of $T_{\min}$. By the construction of $(V, \mathcal{M}_{v_{\min}})$ the solution of the set cover algorithm on instance $(V, \mathcal{M}_{v_{\min}})$ contains at most $d_{\min} \cdot \ln n$ sets, since the sets corresponding to the $d_{\min}$ different $u, v_{\min}$-paths whose union forms T are a solution of $(V, \mathcal{M}_{v_{\min}})$ with cardinality $d_{\min}$. Hence if the algorithm finds a d^*-octopus T then $d^* \leq d_{\min} \cdot \ln n$.

Unfortunately, the size of an instance $(V, \mathcal{M}_v)$ could be exponential in n, thus the algorithm as presented above has a worst case running time that cannot be guaranteed to be polynomial. To obtain a polynomial time algorithm we modify our algorithm as follows: instead of computing a possibly huge set cover instance $(V, \mathcal{M}_v)$ and running the set cover greedy algorithm on it, we compute (again for every fixed v) in each round of the greedy algorithm for every suitable vertex u only one u, v-path in G.

Now let us present our algorithm. Given a graph G, the algorithm computes for every vertex v of G a d-octopus T with root v and outputs an octopus with smallest d (among the n octopuses computed). We describe the algorithm for root v. It starts with T_0 consisting of vertex v. In round $i = 1, 2, \ldots$ the algorithm computes a vertex u_i and T_i is obtained from T_{i-1} by adding a shortest u_i, v-path to T_{i-1} such that the number of dominated vertices increases as much as possible. Consequently T_i is the union of i shortest u_i, v-paths in G.

To find vertex u_i in round i ($i \geq 1$) of the algorithm we need a procedure `best-path` to compute for every vertex y not belonging to T_{i-1}, the value $\mathrm{dom}(y)$ which is the maximum number of vertices of $V \setminus N[V(T_{i-1})]$ (i.e. those vertices not dominated by $V(T_{i-1})$) dominated by any shortest y, v-path in G. The procedure `best-path` also stores a shortest y, v-path in G corresponding to $\mathrm{dom}(y)$. This procedure can be implemented to run in time $O(n^4)$ by a dynamic programming approach using a breadth-first-search with start vertex v. The procedure `best-path` uses techniques similar to the $O(n^{3d+2})$ algorithm of Theorem 3.

To compute $\mathrm{dom}(y)$ and a best shortest y, v-path for each suitable vertex y w.r.t. to T_{i-1}, an ℓ-partial solution, $\ell \geq 1$, is a triple (x, y, k) where $x \in L_{\ell-1}, y \in L_\ell$ and there is a shortest v, y-path P $v, \ldots, x, y$ dominating within the levels $L_0, L_1, \ldots, L_{\ell-1}$ exactly k vertices which are not already dominated by $V(T_{i-1})$, i.e. $k = |(N[V(P)] \cap \bigcup_{j=0}^{\ell-1} L_j) \setminus N[V(T_{i-1})])|$, where $N[V(T_{i-1})]$ is the set of all vertices already dominated by $V(T_{i-1})$.

The procedure `best-path` starts in round 1 with the computation of all 1-partial solutions $(v, w, 0)$ for all $w \in L_1$. In round ℓ, each $(\ell-1)$-partial solution (x, y, k) is extended to an ℓ-partial solution (y, z, k'), if $\{y, z\} \in E$ and $k' = k + |(L_{\ell-1} \cap N[\{x, y, z\}]) \setminus N[V(T_{i-1})]|$. Finally for every suitable vertex y ($y \notin V(T_{i-1})$) with $y \in L_j$, the value $\mathrm{dom}(y)$ is the largest k-value over all j-partial solutions (x, y, k) (thus $x \in L_{j-1}$). The corresponding shortest y, v-path can be found using a suitable pointer structure. To obtain the timebound $O(n^4)$ the procedure stores in round ℓ an ℓ-partial solution (x, y, k) only if so far no ℓ-partial solution (x, y, k') with $k' \geq k$ has been found.

In round i, the algorithm first calls the procedure `best-path` w.r.t. T_{i-1} and then it chooses a vertex $y_{\max}$ such that $\mathrm{dom}(y_{\max}) = \max\{\mathrm{dom}(y) : y \in V \setminus V(T_{i-1})\}$. Then T_i is obtained by adding to T_{i-1} the shortest $y_{\max}, v$-path corresponding to $\mathrm{dom}(y_{\max})$. Thus $|N[V(T_i)]| = |N[V(T_{i-1})]| + \mathrm{dom}(y_{\max})$. The algorithm terminates in round i if $V(T_i)$ is a dominating set of G, and thus T_i is an i-octopus of G with root v. Otherwise it starts the next round.

Finally notice that each path P chosen during the execution of our $O(n^5)$ algorithm corresponds to a set $N[V(P)]$ of the set cover instance $(V, \mathcal{M}_v)$ that the greedy set cover algorithm may choose. Hence the modified algorithm also has performance ratio $\ln n$. $\square$

4 Approximation algorithms

In this section we provide approximation algorithms for different problems on graphs with small octopus.

4.1 Clustering

For any fixed integer $k \geq 1$, a *k-clustering* of a graph $G = (V, E)$ is a partition $\mathcal{P} = \{C_1, C_2, \ldots, C_t\}$ of V, such that $\mathrm{diam}(G[C_i]) \leq k$ for each set C_i, $i \in \{1, 2, \ldots, t\}$. Each C_i is called a *cluster* of G. By $\mathrm{cl}_k(G)$ we denote the minimum number of clusters in a k-clustering of G. The k-CLUSTERING problem is 'Given a graph G and an integer $l \geq 1$, decide whether $\mathrm{cl}_k(G) \leq l$?'

The 1-CLUSTERING problem is also known as the NP-complete problem PARTITION INTO CLIQUES ([GT15] in [25]) and it remains NP-complete on AT-free graphs [8]. The k-CLUSTERING problem is NP-complete and it remains NP-complete on bipartite graphs for every fixed $k \geq 2$ [1].

The problem MINIMUM k-CLUSTERING is hard to approximate. More precisely for any fixed $k \geq 1$ and every $\varepsilon < 1$, there is no (polynomial time) algorithm to approximate MINIMUM k-CLUSTERING for graphs in general within a factor of $n^{1-\varepsilon}$ unless P=NP [16].

On the other hand for any $k \geq 2$, a linear-time constant-factor approximation algorithm for MINIMUM k-CLUSTERING on graphs with dominating diametral path (i.e. a subclass of graphs with 1-octopus) is given in [16].

Our approximation algorithm for graphs with d-octopus constructs a $(k-2)$-clustering of the d-octopus $T = (W, F)$ using a partition of W in breadth-first-search levels of T. (A 0-clustering is simply a partition of W into singletons.) Then a k-clustering of G is constructed from the $(k-2)$-clustering of T using that W is a dominating set of G.

Theorem 7 *For any fixed $k \geq 2$, there is a linear time approximation algorithm for the* MINIMUM k-CLUSTERING *problem that takes as input a graph G and a d-octopus T of G. The algorithm approximates* $\mathrm{cl}_k(G)$ *within a factor $3d$ for $k = 2$ and within a factor of $2d$ for all $k \geq 3$.*

PROOF. Let $k \geq 2$, $G = (V, E)$ be a graph and let $T = (W, F)$ be a

d-octopus of G with root r. Since W is a dominating set of G every $(k-2)$-clustering $\mathcal{P}$ of T can be extended to a k-clustering $\mathcal{P}'$ of G such that $|\mathcal{P}| = |\mathcal{P}'|$ as follows: For every vertex $v \in V \setminus W$ choose a neighbor $w(v) \in W$. Then for every $C \in \mathcal{P}$, the cluster $C' \in \mathcal{P}'$ corresponding to C is defined as $C' = C \cup \{v \in V \setminus W : \; w(v) \in C\}$.

For $i \geq 0$ let $L_i = \{w \in W : d_T(r, w) = i\}$ be the ith level of T. The neighbourhood of every vertex $v \in L_i$ is partitioned into a set of *predecessors* $u \in L_{i-1}$ and a set of *successors* $w \in L_{i+1}$. (Note that each L_i is an independent set in T.) Every vertex in $W \setminus \{r\}$ has a predecessor and every vertex without successor is an endpoint of one of the d paths defining the octopus T.

We show how to construct a $(k-2)$-clustering of T with at most $d \cdot \ell$ clusters, where $\ell = \lceil \frac{1}{k-1}(\mathrm{ecc}(r, T) + 1) \rceil$. Let S be the union of $k-1$ consecutive levels of T, namely $\lceil (k-3)/2 \rceil$ *upper levels* followed by one or two *central levels* and $\lceil (k-3)/2 \rceil$ *lower levels*. To show $\mathrm{cl}_k(T) \leq d \cdot \ell$ it suffices to prove $\mathrm{cl}_k(T[S]) \leq d$ because we can apply the following argument to the levels $L_0 \dots L_{k-2}$, then to the levels $L_{k-1} \dots L_{2k-2}$, and so on.

We select a spanning subgraph of $T[S]$ by choosing one successor (if there is one) for each vertex in an upper level, one predecessor for each vertex in a lower level and a maximal matching between the central levels (if there are two of them). Every connected component of this subgraph is called *hourglass*. Note that each hourglass H is a tree such that every path in H contains at most two vertices of the same level. Since H contains at most one vertex from each central level this implies $\mathrm{diam}(H) \leq k-2$.

Fig. 1. An hourglass (left) and a partition into 5 hourglasses (right).

Hence the hourglasses define a $(k-2)$-clustering of $T[S]$. To prove that the number of clusters is at most d we observe that every hourglass without vertices in a central level does not contain vertices in lower levels either and is therefore called *upper hourglass*.

Now we show that S is covered by at most d hourglasses. If we have d' upper hourglasses in the partition every central level consists of at most $d - d'$ vertices because T is covered by d paths and d' of them end in upper levels. Hence there are at most $d - d'$ hourglasses containing a central vertex, either because S contains only one central level or because the edges in hourglasses form a maximal matching between the two central levels.

Finally we observe that $(k-2)$-clustering of T defined by hourglasses is computable in linear time. (Note that each level contains at most d vertices.)

Summarizing, we have shown how to construct for every graph G a k-clustering $\mathcal{P}'$ with at most $d \cdot \ell$ clusters, assuming that a d-octopus T of G is part of the input. Combined with a well-known lower bound (see e.g. [16]) we obtain

$$\tfrac{1}{k+1}(\mathrm{diam}(G)+1) \le \mathrm{cl}_k(G) \le |\mathcal{P}'| \le d \cdot \lceil \tfrac{1}{k-1}(\mathrm{ecc}(r,T)+1) \rceil \,.$$

The definition of T implies $\mathrm{ecc}(r,T) \le \mathrm{ecc}(r,G)$. Consequently $\mathrm{ecc}(r,T)+1 \le \mathrm{ecc}(r,G)+1 \le \mathrm{diam}(G)+1 \le (k+1)\cdot \mathrm{cl}_k(G)$. Together with the upper bound for $|\mathcal{P}'|$ this yields

$$|\mathcal{P}'| \le d \cdot \lceil \tfrac{k+1}{k-1} \cdot \mathrm{cl}_k(G) \rceil .$$

Thus the approximation ratio of our algorithm is bounded above by

$$d \cdot \max\{ \tfrac{1}{t} \lceil t \cdot \tfrac{k+1}{k-1} \rceil \;:\; t = 1, 2, \dots, n \} \,.$$

□

4.2 Bandwidth

A *layout* of a graph $G = (V, E)$ is a one-to-one mapping $L\colon V \to \{1, \dots, |V|\}$. The *width* of a graph G and a layout L of G is

$$\mathrm{b}(G, L) := \max\{ |L(u) - L(v)| \colon (u, v) \in E \} \,,$$

and the bandwidth of a graph G is

$$\mathrm{bw}(G) := \min\{ \mathrm{b}(G, L) \colon L \text{ is a layout of } G \} \,.$$

The bandwidth problem has a long history and a number of applications (see [10,11] for a survey). Bandwidth is known to be NP-complete for many graph classes, including cobipartite and thus AT-free graphs [29]. Hence Bandwidth does not become polynomial by adding even a 1-octopus as advice to the input graph (unless P=NP), since for connected AT-free graphs a 1-octopus can be computed in linear time while the bandwidth problem is NP-complete on connected AT-free graphs.

Unger has shown that there is no polynomial time algorithm to approximate the bandwidth of graphs within a constant factor unless P=NP even on a small subclass of trees [37]. The best known approximation algorithm for general graphs has a polylogarithmic performance ratio [20]. Approximation algorithms with constant factor guarantee on AT-free graphs and graphs with dominating pair (i.e. a subclass of graphs with 1-octopus) were obtained in [29].

We present an algorithm to approximate the bandwidth of graphs within $8d$ assuming that a d-octopus of the graph is part of the input. The following partition of the vertex set of G into levels will be useful.

Let $T = (W, F)$ be a d-octopus of the graph $G = (V, E)$ and r be the root of T. First the vertices of T are partitioned into levels $L_0, L_1, \ldots, L_m$, where L_i is the set of vertices at distance i from r in T. Notice that the number of vertices in each level is at most d. We extend this level structure to all vertices of G by assigning vertex $v \in V \setminus W$ to level L_i iff v is adjacent to a vertex of T from level L_i and v is not adjacent to a vertex of T from a level L_j with $j < i$. Since the vertices of T form a dominating set in G, every vertex of G is assigned to one of the levels. We call such a partition of the vertex set of G the (unique) *partition of V associated to T*.

Lemma 8 *Let T be a d-octopus of the graph $G = (V, E)$ and $L_0, \ldots, L_m$ be the partition of V associated to T. Then $|L_i| \le \Delta(G)d$ for all $0 < i \le m$, and for any two vertices $u \in L_i$ and $w \in L_j$, $\{u, w\} \in E$ implies $|i - j| \le 3$.*

PROOF. The first statement follows from the fact that every L_i is dominated by at most d vertices. To prove the second statement we assume w.l.o.g. $u \in L_i$, $w \in L_j$ and $i \le j$. By the construction of the partition there are vertices $u' \in L_i$ and $w' \in L_j$ of T that are adjacent or equal to u and w, respectively, in G.

Suppose $|i - j| > 3$. Let w' be a vertex of the shortest r, l_k-path P of G being a path of T. Consider the path P' of G: $r, \ldots, u', u, w, w', \ldots, l_k$, where $r, \ldots, u'$ is a shortest path in T (of length i) and $w', \ldots, l_k$ is a subpath of P. Consequently P' is an r, l_k-path of G being shorter than P, a contradiction. □

Theorem 9 *Let G be a graph with d-octopus. Then* $\mathrm{bw}(G) \le 4d\Delta(G) - 1$.

PROOF. Let T be a d-octopus of a graph $G = (V, E)$ and let $L_0, L_1, \ldots, L_m$ be the partition of V associated to T. Consider the following layout $L\colon V \to \{1, 2, \ldots, n\}$ of G. The vertices are labeled according to the levels, i.e., for all vertices $u \in L_i$ and $w \in L_j$, $L(u) < L(v)$ implies $i \le j$. In each level L_i the vertices of T are labeled first. By Lemma 8 the width of L is at most $4d\Delta(G) - 1$. □

Theorem 9 and the well-known inequality $\mathrm{bw}(G) \ge \lceil \Delta(G)/2 \rceil$ (see e.g. [10]) imply that a linear-time algorithm computing a layout L as described in the proof of Theorem 9 has worst case approximation ratio $8d$.

Corollary 10 *There is a linear time algorithm to approximate the bandwidth of graphs within a factor of $8d$, if the input graph is given with a d-octopus.*

The following theorem is an improvement of Theorem 9 for AT-free graphs (i.e. $d = 1$). However with this approach we only obtain a 6-approximation algorithm which does not beat the 2-approximation algorithm for bandwidth of AT-free graphs given in [29].

Lemma 11 *For any AT-free graph G,* $\mathrm{bw}(G) \le 3(\Delta(G) - 1)$.

PROOF. The idea of the proof is as follows. Construct by 2LexBFS a dominating pair (a_0, a_m), a partition of $V(G)$ into LexBFS-levels $L_0, L_1, \ldots L_m$ such that $a_0 \in L_0$, and a dominating shortest path $a_0, a_1, \ldots, a_m$ such that $a_i \in L_i$ for all $i \in \{0, 1, \ldots m\}$ as in [29]. Then every vertex of level L_i is adjacent to vertex a_i or a_{i-1}. Therefore, the number of vertices in two consecutive levels is at most $3\Delta(G) - 2$. $\square$

Theorem 9 also provides an upper bound for the topological bandwidth of a graph G, denoted $\mathrm{tbw}(G)$, which is defined as the minimum bandwidth of any subdivision of the graph G. Since for every graph G, $\mathrm{bw}(G) \ge \mathrm{tbw}(G) \ge \lceil \Delta(G)/2 \rceil$ we observe that there is a linear time algorithm to approximate the topological bandwidth of graphs within a factor of $8d$, if the input graph is given with a d-octopus.

Furthermore Theorem 9 can be used to obtain an amazing upper bound on the ratio of bandwidth and topological bandwidth of a graph.

Corollary 12 $\mathrm{bw}(G)/\,\mathrm{tbw}(G) \le 8d$ *for every graph G with a d-octopus.*

Thus it seems interesting to study the ratio of bandwidth and topological bandwidth on AT-free graphs and some of their well-known subclasses like interval and permutation graphs. We mention that the equality of both parameters is known for cographs [31], i.e. a subclass of AT-free graphs and thus a class of graphs with 1-octopus.

4.3 $\boldsymbol{\lambda}$-coloring

Radio frequency assignment is the task to assign radio frequencies to transmitters at different locations without causing interference. This important practical problem can be formulated as a graph coloring problem. The *λ-coloring* problem arises from a certain restricted type of frequency assignment.

Definition 13 *Let d_1 and d_2 be two non-negative integers. A* λ-coloring *of a graph $G = (V, E)$ is a mapping $f\colon V \to \{0, 1, \ldots, \lambda\}$ (an assignment of colors from a set $\{0, 1, \ldots, \lambda\}$ to the vertices of G). The λ-coloring* satisfies the $L(d_1, d_2)$-constraint, *$d_1 \ge d_2$, if for every pair $u, v \in V$, $u \ne v$, the condition $d(u, v) = i$, $1 \le i \le 2$, implies $|f(u) - f(v)| \ge d_i$. The minimum value λ for* *which G admits a λ-coloring satisfying the $L(d_1, d_2)$-constraint is denoted by $\lambda_{d_1,d_2}(G)$.*

Notice that $\lambda_{1,0}(G) + 1$ is equal to the chromatic number of G and that $\lambda_{1,1}(G) + 1$ is the chromatic number of G^2.

λ-colorings satisfying the $L(2,1)$-constraint have been studied extensively (see e.g. [9,26]). There are polynomial time algorithms to compute $\lambda_{2,1}$ for trees and cographs [9]. Approximation algorithms for different graph classes like planar, strongly chordal and permutation graphs are presented in [6,9,24]. $\lambda_{2,1}$-coloring is NP-complete on planar graphs as well as on split graphs [6,24].

We observe that $\lambda_{2,1}$-coloring is also NP-complete for cobipartite graphs, i.e. those graphs that are complement of a bipartite graph. Notice that cobipartite graphs form a very restricted subclass of AT-free graphs.

Theorem 14 *$\lambda_{2,1}$-coloring is NP-complete for cobipartite graphs.*

PROOF. The proof is an immediate corollary of the following result in [6] (which in turn is a slight modification of a proof by Griggs and Yeh [26]).

Let $\mathcal{G}$ be a class of graphs satisfying the following two properties:

- for each graph $G = (V, E) \in \mathcal{G}$, the graph obtained by adding a new vertex v and making it adjacent to every vertex in V also belongs to $\mathcal{G}$.
- Hamiltonian Path is NP-complete for the class of those graphs whose complement belongs to $\mathcal{G}$.

Then the problem to decide for a given graph $G = (V, E) \in \mathcal{G}$ whether $\lambda_{2,1}(G) \leq |V|$ is NP-complete.

Cobipartite graphs satisfy both conditions since Hamiltonian Path is NP-complete for bipartite graphs [25], hence the theorem follows. $\square$

We need the following modification of Lemma 8.

Lemma 15 *Let T be a d-octopus of the graph $G = (V, E)$ and $L_0, L_1, \ldots, L_m$ be the partition of $V(G)$ associated to T. Then for any two vertices $u \in L_i$ and $w \in L_j$, $d(u, w) = 2$ implies $|i - j| \leq 4$.*

PROOF. Let us assume that $u \in L_i$, $w \in L_j$, $i \leq j$ and $x \in N[u] \cap N[w]$. By the construction of the partition there are vertices $u' \in L_i$ and $w' \in L_j$ of T that are adjacent or equal to u and w, respectively, in G.

Suppose $|i-j| > 4$. Let w' be a vertex of the shortest r, l_k-path P of G being a path of T. Consider the path P' of G: $r, \dots, u', u, x, w, w', \dots, l_k$, where $r, \dots, u'$ is a shortest path in T (of length i) and $w', \dots, l_k$ is a subpath of P. Consequently P' is an r, l_k-path of G being shorter than P, a contradiction. $\square$

Theorem 16 *Let $G = (V, E)$ be a graph and let T be a d-octopus of G. Then for any integer $p \geq 1$, $\lambda_{p,1}(G) \leq 4dp\Delta(G) - p + 1$.*

PROOF. Let T be a d-octopus of a graph $G = (V, E)$ and let $L_0, L_1, \dots, L_m$ be the partition of $V(G)$ associated to T.

By Lemma 8, the number of vertices in four consecutive levels $L_i, L_{i+1}, L_{i+2}, L_{i+3}$ is at most $4d\Delta(G)$ for every $i \in \{0, 1, \dots, m-3\}$.

We construct a coloring $f\colon V \to \mathbb{N}$ as follows. For $j \in \{0, 1, \dots, \lfloor m/2 \rfloor\}$ we color each four levels $L_{2j+1}, L_{2j+2}, L_{2j+3}, L_{2j+4}$ by colors from the set $\{0, p, 2p, \dots, 4dp\Delta(G) - p\}$. Moreover, for each $i \in \{1, 2, 3, 4\}$, coloring f assigns to vertices of level L_{2j+i} colors from $\{(i-1)dp\Delta(G), \dots, idp\Delta(G) - p\}$ using each color for at most one vertex in these four levels.

Similarly, we color each of the four levels $L_{2(j+1)+1}, L_{2(j+1)+2}, L_{2(j+1)+3}$, and $L_{2(j+1)+4}$ by colors from $\{1, p+1, 2p+1, \dots, 4dp\Delta(G) - p + 1\}$.

Therefore, if two vertices $u \in L_x$ and $w \in L_y$, obtain the same color, i.e. $f(u) = f(v)$, then $|x - y| \geq 7$ and if $0 < |f(u) - f(v)| \leq p$, then $|x - y| \geq 4$.

Notice that the coloring f does not assign a color larger than $4dp\Delta(G) - p + 1$ to a vertex. Finally, let us verify that f satisfies the $L(p,1)$-constraint. If two vertices $u \in L_x$ and $w \in L_y$ are adjacent, then by Lemma 8, $|x - y| \leq 3$; hence $|f(u) - f(v)| \geq p$. If $d(u, v) = 2$ then by Lemma 15, $|x - y| \leq 4$ and $|f(u) - f(v)| \geq 1$. $\square$

The λ-coloring described in the proof of Theorem 16 can be computed by a linear time algorithm. Since $\lambda_{p,1}(G) \geq \Delta(G) - 1 + p$, for every graph G and every integer $p \geq 1$, Theorem 16 implies the following

Corollary 17 *For any $p \geq 1$, there is linear time algorithm to approximate a λ-coloring with $L(p,1)$-constraints of a graph $G = (V, E)$ within a factor of $4dp$, if the input graph G is given with a d-octopus T of G.*

The following theorem is an improvement of Theorem 16 for AT-free graphs.

Lemma 18 *For any AT-free graph $G = (V, E)$,*

$$\lambda_{p,1}(G) \leq 3p(\Delta(G) - 1) + 1.$$

PROOF. The idea of the proof is as follows. Find by 2LexBFS a dominating pair (a_0, a_m) of G, a partition of $V(G)$ into LexBFS-levels $L_0, L_1, \dots, L_m$ and a dominating shortest path $a_0, a_1, \dots, a_m$ such that $a_i \in L_i$ for all $i \in \{0, 1, \dots m\}$, as in the proof of Lemma 11 (see also [29]). Then every vertex of level L_i is adjacent to vertex a_i or a_{i-1}. Thus the number of vertices in two consecutive levels is at most $3\Delta(G) - 2$ and we can color vertices in levels L_1, L_2 by using colors from the set $\{0, p, 2p, \dots, p(3\Delta(G)-3)\}$. Then we color vertices from L_3, L_4 with colors from the set $\{1, p+1, 2p+1, \dots, p(3\Delta(G)-3)+1\}$. Then we color vertices in L_5, L_6 by colors from $\{0, p, 2p, \dots, p(3\Delta(G) - 3)\}$ and so on.

Finally it is easy to check that in such a coloring adjacent vertices obtain colors at least p apart and that vertices at distance two have different colors. $\Box$

4.4 Bilateral orientations

An orientation of an undirected graph G is an assignment of directions to the edges of G. An orientation H of G is strongly connected if every two vertices in H are mutually reachable in H. If a graph G is thought as the plan of the system of two-way streets then the orientations of the graph can be viewed as arrangements of one-way streets. Some variants of one-way street assignments were studied in [36]. The orientation problem has other applications in different network routing, broadcasting and gossip problems. See e.g. [27].

Chvátal and Thomassen have initiated in [14] the study of the following quantitative variation of the orientation problem: For a given connected bridgeless graph G, find an orientation of G with the smallest diameter, i.e. the maximum distance one may have to travel in the one-way system should be as small as possible. It was shown in [14] that the problem Bilaterial Orientation: 'Given a graph G and an integer t, decide whether G has an orientation of diameter at most t' is NP-complete. The problem remains NP-complete on cobipartite graphs (a subclass of graphs with 1-octopus) and on split graphs [23].

To prove the main result of this section we need some auxiliary results. In 1939, Robbins [35] proved that an undirected graph G admits a strongly connected orientation if and only if G is connected and bridgeless. This results implies the following.

Corollary 19 *Every connected bridgeless graph $G = (V, E)$ has an orientation H such that* $\mathrm{diam}(H) \leq |V| - 1$.

The next lemma was given in [22]. We provide its proof since it is essential to our algorithm.

Lemma 20 *Let $G_D = (V_D, E_D)$ be a connected bridgeless subgraph of a graph $G = (V, E)$ such that the vertex set of G_D is a dominating set of G. Then for every strongly connected orientation H_D of G_D there is an orientation H of G such that H is an extension of H_D, i.e. each edge of E_D has the same orientation in H as in H_D, and* $\mathrm{diam}(H) \leq \mathrm{diam}(H_D) + 4$. *Furthermore, such an orientation H can be computed by a linear time algorithm.*

PROOF. For every connected component Q of $G - V_D$ we direct the edges being incident to a vertex of Q as follows. If Q consists of one vertex x then x is adjacent to at least two vertices of V_D, say u and v since G is bridgeless. We direct one edge from (x, u) and a second edge (v, x). Since H_D is a strongly connected orientation of G_D, $d_H(x, w) \leq 1 + \mathrm{diam}(H_D)$ and and $d_H(w, x) \leq \mathrm{diam}(H_D)$ for each vertex $w \in V_D$.

Suppose that there are at least two vertices in Q. Every vertex of Q is adjacent in G to a vertex of V_D. Choose a spanning tree of Q with a vertex v as root. We orient the edges of this tree as follows: If a vertex x of the tree has odd distance to v, then we orient all tree edges incident to x towards x. If x has even distance to v then we orient all tree edges incident to x such that x becomes the head. Furthermore for every such vertex x we orient each edge between x and a vertex of G_D towards x if the distance from v on the tree is even, and towards the vertex of G_D otherwise. (See Fig. 2.) Consequently, for every vertex $x \in Q$ and every vertex $w \in V_D$ we have $d_H(x, w) \leq 2 + \mathrm{diam}(H_D)$ and $d_H(w, x) \leq 2 + \mathrm{diam}(H_D)$.

All edges not oriented by the algorithm so far will be oriented arbitrarily.

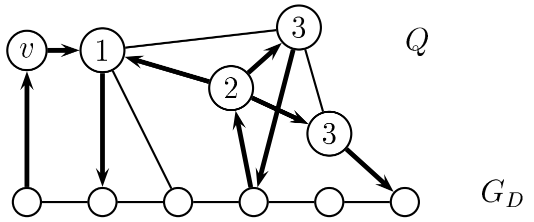

Fig. 2. Orientation of edges in Q

In such an orientation H, for every component Q of $G - V_D$, every vertex $x \in Q$ and every vertex $w \in V_D$ we have $d_H(x, w) \leq 2 + \mathrm{diam}(H_D)$ and $d_H(w, x) \leq 2 + \mathrm{diam}(H_D)$. Therefore, for every $x, y \in V \setminus V_D$ the distance between x and y in H is at most $\mathrm{diam}(H_D) + 4$. Altogether, $\mathrm{diam}(H) \leq \mathrm{diam}(H_D) + 4$.

The existence of a linear time algorithm to compute such an orientation H follows immediately from the proof. □

Theorem 21 *Every connected bridgeless graph G with d-octopus and diameter D has an orientation H such that* $\mathrm{diam}(H) \leq 3dD + 3$.

PROOF. Let $T = (W, F)$ be a d-octopus of the graph G. Then the number of edges in T is at most dD. Let us show how to transform T to a bridgeless graph $\hat{T}$ by adding at most $3dD$ edges.

The graph $\hat{T}$ is obtained by the following procedure

- $\hat{T} := T$
- **While** there is an edge $e = \{u, w\}$ of T such that $\hat{T} - e$ is disconnected **do**
  - Let P be a shortest path among all paths in G having vertices in both components of $\hat{T} - e$ and having no edges of $\hat{T}$. (In other words, P connects two components of $\hat{T} - e$ and is edge disjoint from $\hat{T}$.) Add the edges and vertices of P to $\hat{T}$.

Notice that we can always find such a shortest path P since G is bridgeless. The number of steps in the algorithm is at most $|F| \leq dD$. Since W is a dominating set of G, at every step of the algorithm the length of the path P is at most three. Thus at every step of the algorithm at most two new vertices are added to $\hat{T}$. Therefore, the number of vertices in $\hat{T}$ is at most $|W| + 2dD \leq 3dD$ and by Corollary 19, $\hat{T}$ has an orientation of diameter at most $3dD - 1$. The graph $\hat{T}$ has the octopus T as subgraph, thus its vertex set is a dominating set of G. By Lemma 20, G has an orientation of diameter at most $3dD + 3$. $\square$

Chung, Garey and Tarjan provide a linear-time algorithm to test whether a graph has a strong orientation and finding one if it does [12]. Using this result, Lemma 20 and Theorem 21 we deduce the following

Corollary 22 *There is an $O(nm)$ time algorithm to approximate the oriented diameter of a graph within a factor of $3(d+1)$, if the input graph $G = (V, E)$ is given with a d-octopus T of G.*

4.5 Chordal and interval max-degree

The results of Section 4.2 and in particular Theorem 9 can also be used to obtain approximation algorithms for other graph parameters.

A graph $G = (V, E)$ is called an *interval graph* provided we can assign to each $v \in V$ an interval I_v of the real line such that $\{u, v\} \in E$ if and only if $I_u \cap I_v \neq \varnothing$. A *chord* of a cycle C is an edge joining two non consecutive vertices of C. A *chordless cycle* in G is a cycle of length more than three in G that has no chord. A graph G is *chordal* if it does not contain a chordless cycle. A *chordal completion* of a graph G is a chordal supergraph of G.

Chung and Mumford in [13] initiated the study of chordal completions with smallest maximum degree. They have shown that a chordal completion of the n by n grid must contain a vertex of degree at least cn.

The *chordal max-degree* of a graph G is

$$\operatorname{cd}(G) := \min\{\Delta(G') \colon G' \text{ is a chordal supergraph of } G\}\,.$$

This parameter has some important applications in artificial intelligence, see [13] for further references.

Another related parameter, the interval max-degree of a graph, was studied by Fomin and Golovach in [21]. The *interval max-degree* of a graph G is

$$\operatorname{id}(G) := \min\{\Delta(G') \colon G' \text{ is an interval supergraph of } G\}\,.$$

Since every interval graph is chordal (see, e.g. [7]) we have that for any graph G, $\Delta(G) \leq \operatorname{cd}(G) \leq \operatorname{id}(G)$. Fomin and Golovach have proved in [21] that for any graph G, $\operatorname{bw}(G) \leq \operatorname{id}(G) \leq 2\operatorname{bw}(G)$. Combining the latter with Theorem 9 we obtain the following theorem.

Theorem 23 *Let G be a graph with a d-octopus. Then $\Delta(G) \leq \operatorname{cd}(G) \leq \operatorname{id}(G) \leq 8d\Delta(G) - 2$.*

Notice that the results of Unger [37] on hardness of bandwidth approximation imply that there is no polynomial time algorithm to approximate the interval max-degree of graphs within a constant factor unless P=NP. On the other hand, for graphs with small octopus Corollary 10 and Theorem 23 imply the following

Corollary 24 *There is a linear time algorithm to approximate the interval and the chordal max-degree of a graph within a factor of $8d$ if the input graph is given with a d-octopus.*

Corollary 25 *Let G be a graph with a d-octopus. Then $\operatorname{id}(G)/\operatorname{cd}(G) \leq 8d$.*

4.6 Domino pathwidth and domino treewidth

Theorem 9 can be also used to approximate another graph parameter, namely the domino-treewidth introduced in [5].

A pair $(\mathcal{X}, T)$ is a *domino-tree-decomposition* of a graph $G = (V, E)$ if $\mathcal{X} = \{X_i : i \in I\}$ is a set of subsets $X_i \subseteq V$ and $T = (I, F)$ is a tree such that

- $1 \leq |\{i : v \in X_i\}| \leq 2$ for each vertex $v \in V$,
- for each edge $e \in E$ there is an index i such that $e \subseteq X_i$,
- if j is a vertex on the path from i to k in T, then $X_i \cap X_k \subseteq X_j$.

The *width* of a decomposition $(\mathcal{X}, T)$ is $\max\{|X_i| : i \in I\} - 1$. The *domino-treewidth* of G, denoted by $\mathrm{dtw}(G)$, is the minimum width of a domino-tree-decomposition of G. A *domino-path-decomposition* of a graph G is a domino-tree-decomposition $(\mathcal{X}, T)$ of G where T is a path. The *domino-pathwidth* of G, denoted by $\mathrm{dpw}(G)$, is the minimum width of a domino-path-decomposition of G. Notice that $\mathrm{dpw}(G) \geq \mathrm{dtw}(G) \geq \Delta(G)/2$ for any graph G.

Lemma 26 *For any graph G, $\frac{1}{2}(\mathrm{dpw}(G) + 1) \leq \mathrm{bw}(G)$.*

PROOF.

Let L be an optimal bandwidth layout of G of width k. We construct a domino-path-decomposition $(\mathcal{X}, T)$ of $G = (V, E)$ with $T = (X_1, \ldots, X_{l-1})$ of width $2k - 1$ (we may think that $|V| = lk$) by setting $X_i := \{v : (i-1)k < L(v) \leq (i+1)k\}$ for each $i \in \{1, 2, \ldots, l-1\}$. Since $X_i \cap X_{i+2} = \varnothing$ we have that the pair $(\mathcal{X}, T)$ satisfies the first and the third property of a domino-tree-decomposition. For every edge the difference between its endpoints in the layout L is at most k. Since $|X_i \cap X_{i+1}| = k$, we have that every edge is in some X_i, thus the second property of a domino-tree-decomposition is also fulfilled. $\square$

Clearly, the domino-treewidth of a graph is at most its domino-pathwidth. Thus by Theorem 9 we obtain the following result.

Theorem 27 *Let G be a graph with a d-octopus. Then* $\mathrm{dtw}(G) \leq \mathrm{dpw}(G) \leq 8d\Delta(G) - 3$.

Combining Lemma 26 with the hardness of bandwidth approximation implies that there is no polynomial time algorithm to approximate the domino-pathwidth of graphs within a constant factor unless P=NP. It is worth mentioning that no lower bound for the approximation of the domino-treewidth is known.

Since $\mathrm{dtw}(G) \geq \Delta(G)/2$ for any graph G, Corollary 10 and Theorem 27 imply

Corollary 28 *There is a linear time algorithm to approximate the domino-treewidth and the domino-pathwidth of graphs within a factor of $16d$ if the* *input graph is given with a d-octopus.*

Corollary 29 *Let G be a graph with a d-octopus. Then* $\operatorname{dpw}(G)/\operatorname{dtw}(G) \leq 16d$.

5 Domination

We already mentioned that there is a constant $c > 0$ such that there is no approximation algorithm with factor $c \log n$ for the Domination problem unless P=NP [34]. Similar statements hold for the problems Total Domination and Connected Domination as a consequence of theorems on the ratios of domination parameters:

$$\begin{aligned}
\gamma(G) \leq \gamma_{\text{total}}(G) &\leq \gamma_{\text{conn}}(G) \\
\gamma_{\text{total}}(G) &\leq 2\gamma(G) \\
\gamma_{\text{conn}}(G) &\leq 2\gamma_{\text{total}}(G) - 2 && \text{([19])} \\
\gamma_{\text{conn}}(G) &\leq 3\gamma(G) - 2 && \text{([18])}
\end{aligned}$$

(The first equation is not true if $\gamma(G) = \gamma_{\text{conn}}(G) = 1$ since then $\gamma_{\text{total}}(G) = 2$. Furthermore only graphs are considered for which the corresponding parameters exist.)

The following lemma shows that any d-octopus can already be used as output of an approximation algorithm.

Lemma 30 *Let $T = (W, F)$ be a d-octopus of a graph $G = (V, E)$. Then*

- *W is a dominating set of G with $|W| \leq 3d\gamma(G)$,*
- *W is a total dominating set of G with $|W| \leq 2d\gamma_{\text{total}}(G)$, and*
- *W is a connected dominating set of G with $|W| \leq d(\gamma_{\text{conn}}(G) + 2)$.*

PROOF. By definition, W is a connected dominating set of G, and thus it is also a total dominating set a and dominating set of G.

Let r be the root of the d-octopus T and let q be the length of a longest r, l_i-path in T. Let $L_0 = \{r\}, L_1 = N(r), \ldots, L_s$ be the levels of a breadth-first-search on G with start vertex r. Clearly $s \geq q$ since each r, l_i-path of T is a shortest path in G.

Each level L_i, $1 \leq i \leq q$, contains at most d vertices of T, thus $|W| \leq dq + 1$. Since each dominating set of G must contain a vertex of every third BFS-level, we obtain $\gamma(G) \geq (q+1)/3$. Since each total dominating set of G must contain at least two vertices of every four BFS-levels, we obtain $\gamma_{\text{total}}(G) \geq (q+1)/2$. Since each connected dominating set of G must contain one vertex of each BFS-level, except possibly L_0 and L_s, we obtain $\gamma_{\text{conn}}(G) \geq q-1$.

Combining all inequalities we obtain $|W| \leq 3d\gamma(G)$, $|W| \leq 2d\gamma_{\text{total}}(G)$ and $|W| \leq d(\gamma_{\text{conn}}(G)+2)$. $\square$

To obtain exact algorithms for computing a minimum dominating set and a minimum total dominating for graphs having a d-octopus, we shall rely on algorithms and proof techniques developed in [32] to obtain exact algorithms for the domination and total domination problem on AT-free graphs and on graphs with a dominating shortest path, i.e. graphs with a 1-octopus. In particular, the algorithms $mcds_w(G)$ and $mctds_w(G)$ were introduced and the following properties of these algorithms were proved in [32].

Theorem 31 ([32], Theorem 6) *The algorithm* $\mathrm{mcds}_w(G)$ *computes in time* $O(n^{w+2})$ *a minimum cardinality dominating set of a given connected graph* $G = (V, E)$*, if* G *has a minimum cardinality dominating set* D *and a vertex* $x \in V$ *such that at most* w *vertices of* D *belong to any three consecutive* BFS*-levels when* x *is the start vertex of the breadth-first-search.*

Theorem 32 ([32], Theorem 9) *Algorithm* $\mathrm{mctds}_w(G)$ *computes in time* $O(n^{w+2})$ *a minimum cardinality total dominating set of a given connected graph* $G = (V, E)$*, if* G *has a minimum cardinality total dominating set* D' *and a vertex* $x \in V$ *such that at most* w *vertices of* D' *belong to any three consecutive* BFS*-levels when* x *is the start vertex of the breadth-first-search.*

The following lemma is an extension of Theorem 3 of [32] to graphs with a d-octopus. In fact we shall relax this requirement a bit. Instead of a d-octopus with root r we only require the existence of a dominating set S such that $|\{v \in S : d_G(r,v) = i\}| \leq d$ for all $i \geq 0$.

Lemma 33 *Let* $G = (V, E)$ *be a graph and* x *a vertex of* G*. Let* $L_0 = \{x\}, L_1 = N(x), \ldots, L_i = \{w \in V : d_G(x,w) = i\}, \ldots, L_q = \{w \in V : d_G(x,w) = q\}$ *be the* BFS*-levels of* x*. Suppose* G *has a (total) dominating set* S*, such that* S *has at most* d *vertices in common with each* BFS*-level* L_i*. Then* G *has a minimum cardinality (total) dominating set* D *such that*

$$\bigwedge_{i \in \{0,1,\ldots,\ell\}} \; \bigwedge_{j \in \{0,1,\ldots,\ell-i\}} \quad |D \cap \bigcup_{s=i}^{i+j} L_s| \leq (j+5)d - 1. \qquad (*)$$

PROOF. We prove the part on total domination only, the proof of the other part (on domination) follows the same lines. Starting with any minimum cardinality total dominating set D_0 we construct sets $D_1, D_2, \ldots$ until we reach a set D with property $(*)$. In each step we replace some vertices in D_r by certain vertices in S such that the resulting set D_{r+1} is again a minimum cardinality total dominating set.

For $0 \leq i \leq q$ and $0 \leq j \leq q-i$ let $L_i^j = \bigcup_{s=i}^{i+j} L_s$ and

$$Q_r = \{(i,j) \ : \ |D_r \cap L_i^j| \geq (j+5)d\}\,.$$

If D_r does not fulfill property $(*)$ then $Q_r \neq \varnothing$. We choose a pair $(i_r, j_r) \in Q_r$ such that first $i_r = \min\{i \ : \ (i,j) \in Q_r\}$ and then $j_r = \max\{j \ : \ (i_r, j) \in Q_r\}$. Now we define $D_{r+1} = (D_r \setminus L_{i_r}^{j_r}) \cup (S \cap L_{i_r-2}^{j_r+2})$.

Since both D_r and S are total dominating sets of G, D_{r+1} is a total dominating set of G because every vertex in L_s is adjacent to a vertex in $S \cap L_{i_r-2}^{j_r+2}$ (for $i_r - 1 \leq s \leq i_r + j_r + 1$) or $D_r \setminus L_{i_r}^{j_r}$ (otherwise). Furthermore D_{r+1} is a minimum total dominating set since $|D_r \cap L_{i_r}^{j_r}| \geq (j_r+5)d$ and $|S \cap L_{i_r-2}^{j_r+2}| \leq (j_r+5)d$. Especially this means that both sets contain exactly $(j_r+5)d$ vertices and the boundary cases $i_r \in \{0,1\}$ or $i_r + j_r \in \{q-1, q\}$ are impossible because $L_s = \varnothing$ for $s < 0$ or $s > q$.

It remains to show that the sequence of sets $D_0, D_1, \ldots$ which do not fulfill property $(*)$ is finite. To do this we prove $i_r + j_r < i_{r+1}$ for all steps of our construction with $Q_{r+1} \neq \varnothing$.

Assume $i_{r+1} \leq i_r + j_r + 2$. By our choice of i_r and j_r this implies $i_{r+1} + j_{r+1} \geq i_r$ because $D_r \cap L_0^{i_r-3} = D_{r+1} \cap L_0^{i_r-3}$. Next we have $i_{r+1} + j_{r+1} \geq i_r + j_r + 2$ because $|D_{r+1} \cap L_s| \geq d$ for all indices s with $i_r - 2 \leq s \leq i_r + j_r + 2$. But this implies $|D_{r+1} \cap L_{i_{r+1}}^{j_{r+1}}| = |D_r \cap L_{i_{r+1}}^{j_{r+1}}|$ contradicting our choice of i_r and j_r again. Consequently $i_{r+1} > i_r + j_r + 2$ holds. $\square$

Combining Lemma 33 with Theorems 31 and 32, we obtain

Theorem 34 *There are $O(n^{7d+2})$ time algorithms taking as input a graph G that is known to have a d-octopus and computing a minimum dominating set and a minimum total dominating set of G.*

Remark 35 *To guarantee the existence of the $O(n^{7d+2})$ time algorithms it is enough to require that the input graph G has* BFS*-levels $L_0, L_1, \ldots, L_q$ (for some vertex) and a dominating set S and a total dominating set S', respectively, such that S and S', respectively, have at most d vertices in common with each* BFS*-level.*
If the root of a d-octopus is known we shall gain a factor of n in the running time.

6 Conclusions

We have shown how approximation algorithms can use a small octopus as advice. Efficient constant-factor approximation algorithms for graphs with small octopus have been obtained for the problems k-Clustering, Bandwidth, λ-Coloring, Domino Treewidth, Domino Pathwidth, Bilateral Orientation, Interval Max-Degree and Chordal Max-Degree. These algorithms approximate the corresponding graph parameters within a factor of $c \cdot d$, if the input graph is given with a d-octopus. Thereby c is a small constant depending on the problem. As a byproduct we obtained exact polynomial time algorithms for the problems Domination and Total Domination for graphs with d-octopus, where we do not require a d-octopus to be part of the input.

Unfortunately, there are problems for which a d-octopus as advice does not help. Let G' be the graph obtained from G by adding a new vertex adjacent to all vertices of G. Whatever graph G is, G' has a 1-octopus consisting of the additional vertex only. By this simple construction we observe that the problems Clique, Independent Set, Coloring, Partition into Cliques, Treewidth and Pathwidth are NP-complete for graphs with 1-octopus. Furthermore, all these problems are as difficult to approximate for graphs in general as for graphs having a 1-octopus. Thus, k-clustering is hard to approximate for $k = 1$ even when the inputs are supposed to be a graph and a 1-octopus of the graph; on the other hand, for each $k \geq 2$ there are linear time constant factor approximation algorithms for the k-clustering problem.

Acknowledgments

We are grateful to an anonymous referee for pointing out a flaw in a previous version of the proof of Theorem 3.

References

[1] N. Abbas and L. Stewart, *Clustering bipartite and chordal graphs: Complexity, sequential and parallel algorithms*, Discrete Appl. Math. 91 (1999), pp. 11-23.

[2] R. Allan, R. Laskar and S.T. Hedetniemi, *A note on total domination*, Discrete Math. 49 (1984), pp. 7–13.

[3] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela and M. Protasi, *Complexity and approximation*, Springer, 1999.

[4] H.L. Bodlaender, *Treewidth: Algorithmic techniques and results*, in Proceedings of Mathematical Foundations of Computer Science (MFCS 1997), 19–36, Lecture Notes in Comput. Sci., 1295, Springer, Berlin, 1997.

[5] H. L. Bodlaender and J. Engelfriet, *Domino treewidth*, J. Algorithms 24 (1997), pp. 94–123.

[6] H.L. Bodlaender, T. Kloks, R.B. Tan and J. van Leeuwen, *λ-coloring of graphs*, in Proceedings of Symposium on Theoretical Aspects of Computer Science (STACS 2000), 395–406, Lecture Notes in Comput. Sci., 1770, Springer, Berlin, 2000.

[7] A. Brandstädt, V.B. Le and J.P. Spinrad, *Graph classes: a survey*, SIAM Monographs on Discrete Mathematics and Applications, Philadelphia, 1999.

[8] H. Broersma, T. Kloks, D. Kratsch and H. Müller, *Independent sets in asteroidal triple-free graphs*, SIAM J. Discrete Math. 12 (1999), pp. 267–287.

[9] G.J. Chang and D. Kuo, *The $L(2,1)$-labeling problem on graphs*, SIAM J. Discrete Math. 9 (1996), pp. 309–316.

[10] P.Z. Chinn, J. Chvátalová, A.K. Dewdney and N.E. Gibbs, *The bandwidth problem for graphs and matrices—a survey*, J. Graph Theory 6 (1982), pp. 223–254.

[11] F.R.K. Chung, *Labelings of graphs*, in Selected Topics in Graph Theory, Academic Press, New York, 1988, pp. 151–168.

[12] F.R.K. Chung, M.R. Garey and R.E. Tarjan, *Strongly connected orientations of mixed multigraphs*, Networks 15 (1985), pp. 477–484.

[13] F.R.K. Chung and D. Mumford, *Chordal completions of planar graphs*, J. Combin. Theory Ser. B 62 (1994), pp. 96–106.

[14] V. Chvátal and C. Thomassen, *Distances in orientations of graphs*, J. Combin. Theory Ser. B 24 (1978), pp. 61–75.

[15] D.G. Corneil, S. Olariu and L. Stewart, *A linear time algorithm to compute dominating pairs in asteroidal triple-free graphs*, SIAM J. Comput. 28 (1999), pp. 1284–1297.

[16] J.S. Deogun, D. Kratsch and G. Steiner, *An approximation algorithm for clustering graphs with dominating diametral path*, Inform. Proc. Letters 61 (1997), pp. 121–127.

[17] R.G. Downey and M.R. Fellows, *Parameterized Complexity*, Springer, Berlin, 1999.

[18] P. Duchet and H. Meyniel, *On Hadwiger's number and the stability number*, Annals of Discrete Mathematics 13 (1982), pp. 71–74.

[19] O. Favaron and D. Kratsch, *Ratios of domination parameters*, in Advances in Graph Theory, V. Kulli, (ed.), Vishwa International, 1991, pp. 173–182.

[20] U. Feige, *Approximating the bandwidth via volume respecting embeddings*, J. Comput. Syst. Sci. 60 (2000), pp. 510–539.

[21] F.V. Fomin and P.A. Golovach, *Interval completion with the smallest max-degree*, in Proceedings of Graph-theoretic concepts in computer science (WG 1998), 359–371, Lecture Notes in Comput. Sci., 1517, Springer, Berlin, 1998.

[22] F.V. Fomin, M. Matamala, E. Prisner and I. Rapaport, *Bilateral orientations and domination*, manuscript, 2001.

[23] F. V. Fomin, M. Matamala and I. Rapaport, *The complexity of approximating the oriented diameter of chordal graphs*, to appear in Proceedings of Graph-Theoretic Concepts in Computer Science (WG 2002), Lecture Notes in Comput. Sci., Springer, Berlin, 2002.

[24] D.A. Fotakis, S.E. Nikoletseas, V.G. Papadopoulou and P.G. Spirakis, *NP-completeness results and efficient approximations for radiocoloring in planar graphs*, in Proceedings of Mathematical Foundations of Computer Science (MFCS 2000), 363–372, Lecture Notes in Comput. Sci., 1893, Springer, Berlin, 2000.

[25] M.R. Garey and D.S. Johnson, *Computers and Intractability: A guide to the Theory of NP-completeness*, Freeman, New York, 1979.

[26] J.R. Griggs and R.K. Yeh, *Labelling graphs with a condition at distance 2*, SIAM J. Discrete Math. 5 (1992), pp. 586–595.

[27] S.M. Hedetniemi, S.T. Hedetniemi and A.L. Liestman, *A survey of gossiping and broadcasting in communication networks*, Networks 18 (1988), pp. 319–349.

[28] D.S. Johnson, *Approximation algorithms for combinatorial problems*, J. Comput. System Sci. 9 (1974), pp. 256–278.

[29] T. Kloks, D. Kratsch and H. Müller, *Approximating the bandwidth for AT-free graphs*, J. Algorithms 32 (1999), pp. 41–57.

[30] T. Kloks, D. Kratsch and H. Müller, *On the structure of graphs with bounded asteroidal number*, Graphs and Combinatorics 17 (2001), pp. 295–306.

[31] T. Kloks and R.B. Tan, *Bandwidth and topological bandwidth of graphs with few P_4s*, Discrete Appl. Math. 115 (2001), pp. 117–133.

[32] D. Kratsch, *Domination and total domination on asteroidal triple-free graphs*, Discrete Appl. Math. 99 (2000), pp. 111–123.

[33] L. Lovász, *On the ratio of optimal integral and fractional covers*, Discrete Math. 13 (1975), pp. 383–390.

[34] R. Raz and S. Safra, *A sub-constant error-probability low-degree test, and sub-constant error-probability PCP characterization of NP*, in Proceedings of the 29*th* Annual ACM Symposium on Theory of Computing (STOC'97), 475–484, ACM, 1997.

[35] H.E. Robbins, *A theorem on graphs with an application to a problem of traffic control*, Amer. Math. Monthly 46 (1939), pp. 281–283.

[36] F.S. Roberts and Y. Xu, *On the optimal strongly connected orientations of city street graphs, I. Large grids*, SIAM J. Discrete Math. 1 (1988), pp. 199–222.

[37] W. Unger, *The complexity of the approximation of the bandwidth problem*, in Proceedings of the 39*th* Annual IEEE Symposium on Foundations of Computer Science (FOCS'98), 82–91, IEEE, 1998.