Deleting edges to restrict the size of an epidemic: a new application for treewidth

Jessica Enright[1] and Kitty Meeks[2]

[1] Computing Science and Mathematics, University of Stirling jae@cs.stir.ac.uk
[2] School of Mathematics and Statistics, University of Glasgow
kitty.meeks@glasgow.ac.uk

Abstract. Motivated by applications in network epidemiology, we consider the problem of determining whether it is possible to delete at most k edges from a given input graph (of small treewidth) so that the maximum component size in the resulting graph is at most h. While this problem is NP-complete in general, we provide evidence that many of the real-world networks of interest are likely to have small treewidth, and we describe an algorithm which solves the problem in time $O((wh)^{2w}n)$ on an input graph having n vertices and whose treewidth is bounded by a fixed constant w.

1 Introduction

Network epidemiology seeks to understand the dynamics of disease spreading over a network or graph, and is an increasingly popular method of modelling real-world disease. The rise of network epidemiology corresponds to a rapid increase in the availability of contact network datasets that can be encoded as networks or graphs: typically, the vertices of the graph represent agents that can be infected and infectious, such as individual humans or animals, or appropriate groupings of these, such as cities, households, or farms. The edges are then the potentially infectious contacts between those agents. Considering the contacts within a population as the edges of a graph can give a large improvement in disease modelling accuracy over mass action models, which assume that a population is homogeneously mixing. For example, if we consider a sexual contact network in which the vertices are people and the edges are sexual contacts, the heterogeneity in contacts is very important for explaining the pattern and magnitude of an AIDS epidemic [1].

Our work has been especially motivated by the idea of controlling diseases of livestock by preventing disease spread over livestock trading networks. As required by European law, individual cattle movements between agricultural holdings in Great Britain are recorded by the British Cattle Movement Service (BCMS) [20]; in early 2014, this dataset contained just under 300 million trades and just over 133,000 agricultural holdings. For modelling disease spread across the British cattle industry, it is common to create vertices from farms, and edges from trades of cattle between those farms: a disease incursion starting at a single

farm could spread across this graph through animal trades, as is thought to have happened during the economically-damaging 2001 British foot-and-mouth disease crisis [15].

We are interested in controlling or limiting the spread of disease on this sort of network, and so have focussed our attention on edge deletion, which might correspond to forbidden trade patterns or, more reasonably, extra vaccination or disease surveillance along certain trade routes. Introducing extra controls of this kind is costly, so it is important to ensure that this is done as effectively as possible. Our target graph class is also informed by our disease motivation: when a contagion spreads over the edges of a graph, the maximum component size is an upper bound on the maximum number of vertices infected from a single initially infected vertex. To this end, we consider the problem of determining whether a given graph can be modified, using only up to k edge-deletion operations, so that the resulting graph has maximum component size at most h. We also discuss a number of relevant extensions:

- assigning different weights to different vertices (e.g. corresponding to the number of animals in a particular animal holding), and seeking to bound the total weight of each connected component;
- associating different costs with the deletion of different edges;
- imposing different limits on the size of components containing individual vertices (for example, we might want to enforce a smaller size limit for components containing certain vertices considered to be particularly high risk).

This problem is intractable in general, so in order to develop useful algorithms for real-world applications we need to exploit structural properties of the input network. In Section 2 we provide evidence that many animal trade networks of interest are likely to have small treewidth. In Section 3 we then go on to describe an algorithm to solve the problem whose running time on an n-vertex graph of treewidth w is bounded by $O((wh)^{2w}n)$; this algorithm is easily adapted to output an optimal solution. Many problems that are thought to be intractable in general are known to admit polynomial-time algorithms when restricted to graphs of bounded treewidth, often by means of a dynamic programming strategy similar to that used to attack the problem considered here; however, to the best of the authors' knowledge, the usefulness of such algorithms for solving real-world network problems has yet to be investigated thoroughly.

In reality, policy decisions about where to introduce controls are likely to be influenced by a range of factors, which cannot all be captured adequately in a network model. Thus, the main application of our algorithm will be in comparing any proposed strategy with the theoretical optimum: a policy-maker can determine whether there is a solution with the same total cost that results in a smaller maximum component size; extensive experiments on real animal movement data are left as a task for future work.

In the remainder of this section, we begin by reviewing previous related work in Section 1.1 before introducing some important notation in Section 1.2 and reviewing the key features of tree decompositions in Section 1.3. A discussion of

the treewidth of real-world networks is given in Section 2, and our algorithm is described in Section 3.

1.1 Review of Previous Work

The problem of modifying a graph to bound the maximum component size has previously been studied both in the setting of epidemiology [18] and in the study of network vulnerability [13, 7]. The edge-modification version we consider here appears in the literature under various names, including the *component order edge connectivity problem* [13] and the *minimum worst contamination problem* [18]. Li and Tang [18] show that it is NP-hard to approximate the minimisation version of the problem to within $2 - \epsilon$, while Gross et. al.[13] describe a polynomial-time algorithm to solve the problem when the input graph is a tree.

From a combinatorial perspective, this problem belongs to the more general family of *edge-deletion problems*. An edge-deletion problem asks if there is a set of at most k edges that can be deleted from an input graph to produce a graph in some target class. In contrast to the related well-characterised vertex-deletion problems [17], there is not yet a complete characterisation of the hardness of edge-deletion problems by target graph class.

Yannakakis [24] gave early results in edge-deletion problems, showing that edge-deletion to planar graphs, outer-planar graphs, line graphs, and transitive digraphs is NP-complete. Subsequently, Watanabe, Ae and Nakamura [23] showed that edge-deletion problems are NP-complete if the target graph class can be finitely characterised by 3-connected graphs. There are a number of further hardness results known for edge-deletion to well-studied graph classes, including for interval and unit interval graphs [11], cographs [8], and threshold graphs [19] and, as noted in [21], hardness of edge-deletion to bipartite graphs follows from the hardness of a MAX-CUT problem. Natanzon, Shamir and Sharan [21] further showed NP-completeness of edge-deletion to disjoint unions of cliques, and perfect, chain, chordal, split, and asteroidal-triple-tree graphs, but also give polynomial-time algorithms, in the special case of the input graph having bounded degree, for edge-deletion to chain, split, and threshold graphs.

Given the large number of hardness results in the literature, it is natural to consider the parameterised complexity of these problems. Cai [5] initiated this investigation, showing that edge-deletion to a graph class characterisable by a finite set of forbidden induced subgraphs is fixed-parameter tractable when parameterised by k (the number of edges to delete): he gave an algorithm to solve the problem in time $O(d^{2k} \cdot n^{d+1})$, where n is the number of vertices in the input graph and d is the maximum number of vertices in a forbidden induced subgraph. Further fpt-algorithms have been obtained for edge-deletion to split graphs [10] and to chain, split, threshold, and co-trivially perfect graphs [14].

Considering the problem of deleting edges to obtain a graph with restricted maximum componet size, restricted to graphs of small treewidth, the algorithm we describe in this paper represents a significant improvement on Cai's result [5] above, which implies the existence of an algorithm running in time $O(h^{2k} \cdot n^h)$ (on arbitrary input graphs). While the fixed parameter tractability of this

problem (parameterised by the maximum component size h) restricted to graphs of bounded treewidth does follow from the optimization version of Courcelle's Theorem [3, 6], this does not lead to a practical algorithm for addressing real-world problems.

1.2 Notation and Problem Definition

Unless otherwise stated, all graphs are simple, undirected, and loopless. For graph $G = (V, E)$, $V = V(G)$ is the vertex set of G, and $E = E(G)$ the edge set of G. We denote the sizes of the edge and vertex sets of G as $e(G) = |E(G)|$ and $v(G) = |V(G)|$. For further general graph notation, we direct the reader to [12].

A *partition* $\mathcal{P}$ of a set X is a collection of disjoint, non-empty sets whose union is X. We call each set in the partition a *block* of the partition, and every partition corresponds to a unique equivalence relation on X where $x \sim y$ if and only if x and y belong to the same block of X.

In this paper, we consider the following problem, were $\mathcal{C}_h$ is the set of all connected graphs on h vertices.

$\mathcal{C}_h$-Free Edge Deletion
Input: A Graph $G = (V, E)$ and an integer k.
Question: Does there exist $E' \subseteq E$ with $|E'| = k$ such that $G \setminus E'$ does not contain any $H \in \mathcal{C}_h$ as an induced subgraph?

This problem is NP-complete even for $h = 4$: in [9] we outline an easy proof of this result, by means of a reduction from Perfect Triangle Cover, which relies on the observation that the maximum number of edges in a graph having maximum component size h is obtained if the graph is a disjoint union of h-cliques. In particular, this indicates that parameterisation by h alone will not be sufficient to give an fpt-algorithm.

1.3 Tree Decompositions

In this section we review the concept of a tree decomposition (introduced by Robertson and Seymour in [22]) and introduce some of the key notation we will use throughout the rest of the paper.

Given any tree T, we will assume that it contains some distinguished vertex $r(T)$, which we will call the *root* of T. For any vertex $v \in V(T) \setminus r(T)$, the *parent* of v is the neighbour of v on the unique path from v to $r(T)$; the set of *children* of v is the set of all vertices $u \in V(T)$ such that v is the parent of u. The *leaves* of T are the vertices of T whose set of children is empty. We say that a vertex u is a *descendant* of the vertex v if v lies somewhere on the unique path from u to $r(T)$ (note therefore that every vertex is a descendant of the root). Additionally, for any vertex v, we will denote by T_v the subtree induced by v together with the descendants of v.

We say that $(T, \mathcal{D})$ is a *tree decomposition* of G if T is a tree and $\mathcal{D} = \{\mathcal{D}(t) : t \in V(T)\}$ is a collection of non-empty subsets of $V(G)$ (or *bags*), indexed by the nodes of T, satisfying:

1. $V(G) = \bigcup_{t \in V(T)} \mathcal{D}(t)$,
2. for every $e = uv \in E(G)$, there exists $t \in V(T)$ such that $u, v \in \mathcal{D}(t)$,
3. for every $v \in V(G)$, if $T(v)$ is defined to be the subgraph of T induced by nodes t with $v \in \mathcal{D}(t)$, then $T(v)$ is connected.

The *width* of the tree decomposition $(T, \mathcal{D})$ is defined to be $\max_{t \in V(T)} |\mathcal{D}(t)| - 1$, and the *treewidth* of G is the minimum width over all tree decompositions of G.

We will denote by V_t the set of vertices in G that occur in bags indexed by the descendants of t in T. Thus, $V_t = \bigcup_{t' \in V(T_t)} \mathcal{D}(t')$.

Although it is NP-hard to determine the treewidth of an arbitrary graph [2], it is shown in [4] that the problem of determining whether a graph has treewidth at most w, and if so computing a tree-decomposition of width at most w, can be solved in linear time for any constant w.

Theorem 1 ([4]). *For each $w \in N$, there exists a linear time algorithm, that tests whether a given graph $G = (V, E)$ has treewidth at most w, and if so, outputs a tree decomposition of G with treewidth at most w.*

A special kind of tree decomposition, known as a *nice tree decomposition*, was introduced in [16]. The nodes in such a decomposition can be partitioned into four types (examples in Figure 1):

Leaf nodes: t is a leaf in T.
Introduce nodes: t has one child t', such that $\mathcal{D}(t') \subset \mathcal{D}(t)$ and $|\mathcal{D}(t)| = |\mathcal{D}(t')| + 1$.
Forget nodes: t has one child t', such that $\mathcal{D}(t') \supset \mathcal{D}(t)$ and $|\mathcal{D}(t)| = |\mathcal{D}(t')| - 1$.
Join nodes: t has two children, t_1 and t_2, with $\mathcal{D}(t_1) = \mathcal{D}(t_2) = \mathcal{D}(t)$.

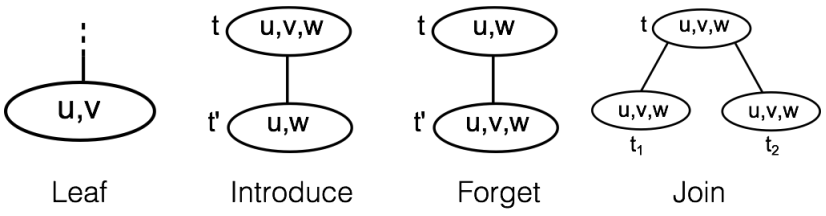

Fig. 1. The four types of node in a nice tree decomposition. From left to right: a leaf, an introduce node, a forget node, and a join node.

Any tree decomposition can be transformed into a nice tree decomposition in linear time:

Lemma 1 ([16]). *For constant k, given a tree decomposition of a graph G of width w and $O(n)$ nodes, where n is the number of vertices of G, one can find a nice tree decomposition of G of width w and with at most $4n$ nodes in $O(n)$ time.*

2 Treewidth of real networks

While the overall graph of cattle trades in Great Britain from 2001 to 2014 is fairly dense, many of the edges are repeated or parallel trades: that is, a farm sending animals over time to the same place, or many individual animals being moved at the same time; when we restrict our attention to a limited time frame, and ignore movements that would generate multiple edges (that is, we require our graph to be simple), the graph is quite sparse. When considering an epidemic, it is much more relevant only to consider trades occurring within some restricted time frame (whose precise duration depends on the disease under consideration).

Moreover, the networks that are obtained by considering shorter time frames typically have an approximately hub-and-spoke or tree-like structure, which results in small treewidth. This can be explained to some extent by considering the structure of the industry and the directionality of farm management styles. For example, beef cattle are likely to flow through dealers or markets, and lead quite short lives, which is likely to result in a hub-and-spoke network. Additionally, farms can sometimes be characterised by "type", with breeders producing calves who then might be grown at one or two other farms before eventual slaughter: this means that cycles are unlikely to occur frequently in the network.

These anecdotal observations about the treewidth of livestock trade networks have been supported by computational calculations on some examples of real cattle trading graphs. First of all, for each year from 2009 to 2014, we generated a graph from a type of persistent trade link recorded by BCMS in Scotland. The largest of these is derived from the trades in 2013, and includes approximately 7,000 nodes and 6,000 edges (this lower density is typical when considering only persistent trade links, or trades over a restricted time period). None of these six graphs has treewidth more than four.

Secondly, in addition to these persistent trade links, we have computed an upper bound of the treewidth of the largest component of an aggregated, undirected version of the overall network of cattle trades in Scotland in 2009 over a variety of time windows 2. The treewidths of these components remains low even for large time windows: for an aggregation of all movements in a 200-day window the treewidth is below 10, and for all movements over the year it is below 18. It is unlikely to be necessary to include a full year of movements in the analysis of any single epidemic, as the time scale of most exotic epidemics is much shorter.

While we have by no means completed an exhaustive study of the structural properties of real-world livestock trade networks, the evidence given here seems sufficient to suggest that algorithms which achieve a good running time on graphs of bounded treewidth will be useful for this application in practice.

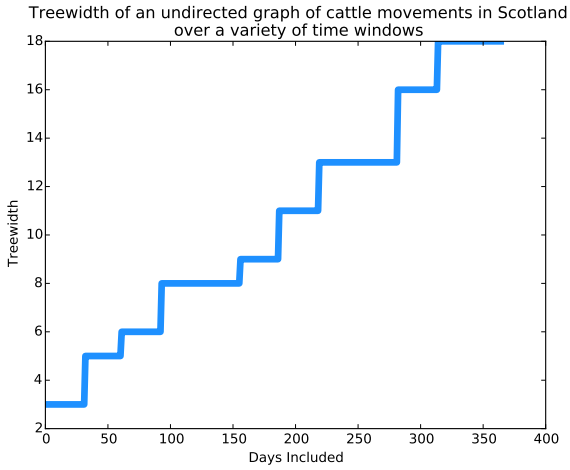

Fig. 2. A plot of an upper bound treewidth of the largest component in an undirected version of the cattle movement graph in Scotland in 2009 over a number of different days included: all day sets start on January 1, 2009. Treewidths below eight are exact, treewidths over eight are upper bounds of the true treewidth.

3 The Algorithm

In this section, we describe an algorithm which, given a graph G together with a nice tree decomposition $(T, \mathcal{D})$ of G of width at most w, determines whether or not it is possible to delete at most k edges from G so that the resulting graph has no component on more than h vertices. Since there exist linear-time algorithms both to compute a tree-decomposition of any graph G of fixed treewidth w, and to transform an arbitrary tree-decomposition into a nice tree decomposition, this in fact gives an algorithm which takes as input just a graph G of treewidth at most w. Thus, we prove the following theorem.

Theorem 2. *There exists an algorithm to solve* $\mathcal{C}_h$-Free Edge Deletion *in time* $O((wh)^{2w}n)$ *on an input graph with* n *vertices whose treewidth is at most* w*.*

As with many algorithms that use tree decompositions, our algorithm works by recursively carrying out computations for each node of the tree, using the results of the same computation carried out on any children of the node in question. In this case, we recursively compute the *signature* of each node: we define the signature of a node in Section 3.1. It is then possible to determine whether we have a yes- or no-instance to the problem by examining the signature of the root of T.

The techniques used to calculate each node's signature from those of its children (which differ slightly depending on which of the four types of node in the nice tree decomposition is being considered) are fairly standard in the

literature, and are omitted here due to space constraints. Full details of the algorithm, together with a mathematical proof of its correctness, are given in [9]. The running time of the algorithm is justified in Section 3.2, and several extensions are also discussed.

3.1 The Signature of a Node

In this section, we describe the information we compute for each node, and define the *signature* of a node.

Throughout the algorithm, we need to record the possible states corresponding to a given bag. A valid *state* of a bag $\mathcal{D}(t)$ is a triple consisting of:

1. a partition $\mathcal{P}$ of $\mathcal{D}(t)$ into disjoint, non-empty subsets or *blocks* of size at most h, and
2. a function $c : \mathcal{P} \to [h]$ such that, for each $X \in \mathcal{P}$, $|X| \leq c(X)$.

We will write $u \sim_{\mathcal{P}} v$ to indicate that u and v belong to the same block of $\mathcal{P}$.

Intuitively, $\mathcal{P}$ tells us which vertices are allowed to belong to the same component of the graph we obtain after deleting edges and c tells us the maximum number of vertices which are permitted in components corresponding to a given block of the partition.

For any bag $\mathcal{D}(t)$, we denote by $\mathrm{st}(t)$ the set of possible states of $\mathcal{D}(t)$. Note that there are at most B_w partitions of a set of size w (where B_w is the w^{th} Bell number) and at most h^w functions from a set of size at most w to $[h]$; thus the total number of valid states for $\mathcal{D}(t)$ is at most $B_w h^w < (wh)^w$ (although not all possible combinations of a partition and a function will give rise to a valid state).

For any given state $\sigma = (\mathcal{P}, c) \in \mathrm{st}(t)$, we set $\mathcal{E}(t, \sigma)$ to be the set of edge-sets $E' \subset E(G[V_t])$ such that $\widetilde{G_t} = G[V_t] \setminus E'$ has the following properties:

1. for each connected component C of $\widetilde{G_t}$:
   (a) $|V(C)| \leq h$, and
   (b) if $C_t = V(C) \cap \mathcal{D}(t) \neq \emptyset$, then C_t is contained in a single block X_C of $\mathcal{P}$,
2. for each block X in $\mathcal{P}$, the total number of vertices in connected components of $\widetilde{G_t}$ that intersect X is at most $c(X)$.

Note that, whenever σ is a valid state for t, the set $\mathcal{E}(t, \sigma)$ will be non-empty: setting $E' = E(G[V_t])$ will always satisfy both conditions. Since we are interested in determining whether it is possible to delete at most k edges to obtain a graph with maximum component size h, we will primarily be interested in a subset of $\mathcal{E}(t, \sigma)$: for any node t and $\sigma \in \mathrm{st}(t)$ we define this subset as

$$\mathcal{E}_k(t, \sigma) = \{E' \in \mathcal{E}(t, \sigma) : |E'| \leq k\}.$$

We then define

$$\mathrm{del}_k(t, \sigma) = \min_{E' \in \mathcal{E}_k(t,\sigma)} |E'|,$$

adopting the convention that the minimum, taken over an empty set, is equal to infinity. To simplify notation, given any $a, b \in \mathbb{N}$, we define $[a]_{\leq b}$ to be equal to a if $a \leq b$, and equal to ∞ otherwise. Finally, we define the *signature* of a node t to be the function $\mathrm{sig}_t : \mathrm{st}(t) \to \{0, 1, \ldots, k, \infty\}$ such that $\mathrm{sig}_t(\sigma) = \mathrm{del}_k(t, \sigma)$.

Observe that, with this definition, our input graph is a yes-instance to $\mathcal{C}_h$-Free Edge Deletion if and only if there exists some $\sigma \in \mathrm{st}(r)$ such that $\mathrm{sig}_r(\sigma) \leq k$, where r is the root of the tree indexing the decomposition.

3.2 Running time and extensions

At each of the $O(n)$ nodes of the nice tree decomposition, we will generate, and then iterate over, fewer than $(wh)^w$ states for that node. For each of those states, we will need to consider a collection of *inherited states* for the node's children; there are at most $(wh)^w$ such states (or pairs of states, in the case of a join node) that need to be considered for each state of the parent node. In the algorithm, we first generate each of the states for a given node, and the corresponding set of inherited states for its children, then iterate over each relevant combination of states, performing various constant-time operations. Thus, at each of $O(n)$ nodes we do $O\left((wh)^{2w}\right)$ work, giving an overall time complexity of $O((wh)^{2w}n)$.

For simplicity, we have only described the most basic version of the algorithm; however, it is straightforward to extend it to deal with more complicated situations, involving any or all of the following.

Deleting edges so that the sum of weights of vertices in any component is at most h, where a weight function $w : V(G) \to \mathbb{N}$ is given: change condition 1(a) in the definition of $\mathcal{E}(t, \sigma)$ to $\sum_{v \in V(C)} w(v) \leq h$, and add to the definition of the set of valid states for a node the condition that, for each block X of $\mathcal{P}$, we have $\sum_{v \in X} w(X) \leq c(X)$.

Determining if it is possible to delete a set of edges whose total cost is at most k, where a cost function $f : E(G) \to \mathbb{N}$ is given: define $\mathrm{del}_k(t, \sigma)$ to be $\min_{E' \in \mathcal{E}(\sigma, t)} \sum_{e \in E'} f(e)$.

Deleting edges so that each vertex v belongs to a component containing at most $\ell(v)$ vertices, where a limit function $\ell : V(G) \to \mathbb{N}$ is given: change condition 1(a) in the definition of $\mathcal{E}(t, \sigma)$ to $|V(C)| \leq \min_{v \in V(C)} \ell(v)$, and add to the definition of the set of valid states for a node the condition that, for each block X of $\mathcal{P}$, we have $c(X) \leq \min_{v \in X} \ell(v)$.

None of these adaptations changes the asymptotic running time of the algorithm.

Additionally, if we wish to output an optimal set of edges to delete in any of the variants (note that in general there may be many such optimal sets), we can simply record, for each node t and each state $\sigma \in \mathrm{st}(t)$, a set of edges $E' \in \mathcal{E}_k(t, \sigma)$ such that $|E'| = \mathrm{del}_k(t, \sigma)$; computing such a set from the relevant sets for the node's children requires only basic set operations. An element of $\mathcal{E}(r, \sigma)$, where r is the root of the tree decomposition and $\mathrm{del}_k(r, \sigma) = \min_{\sigma \in \mathrm{st}(r)}$ is then an optimal solution for the problem.

4 Conclusions and Open Problems

We have investigated the relevance of the well-studied graph parameter treewidth to the structure of real-world animal trade networks, and have provided evidence that this parameter is likely to be small for many networks of interest for epidemiological applications. Motivated by this observation, we have derived an algorithm to solve $\mathcal{C}_h$-Free Edge Deletion on input graphs having n vertices and treewidth bounded by some fixed constant w in time $O((wh)^{2w}n)$. It is straightforward to adapt this algorithm to deal with more complicated situations likely to arise in the application.

An implementation of our approach and its application to real livestock data sets will be one of our next steps; this presents an unusual opportunity to apply a treewidth-based optimisation algorithm to a real-life problem.

Many open questions remain concerning the complexity of this problem more generally, as we are far from having a complete complexity classification. We know that useful structure in the input graph is required to give an fpt-algorithm: we demonstrated that it is not sufficient to parameterise by the maximum component size h alone (unless P=NP). However, it remains open whether the problem might belong to FPT when parameterised only by the treewidth w; we conjecture that treewidth alone is *not* enough, and that the problem is W[1]-hard with respect to this parameterisation. Considering other potentially useful structural properties of input graphs, one question of particular relevance to epidemiology would be the complexity of the problem on planar graphs: this would be relevant for considering the spread of a disease based on the geographic location of animal holdings (in situations where a disease is likely to be transmitted between animals in adjacent fields).

Furthermore, animal movement networks can capture more information on real-world activity when considered as *directed* graphs, and the natural generalisation of the problem to directed graphs in this context would be to consider whether it is possible to delete at most k edges from a given directed graph so that the maximum number of vertices *reachable* from any given starting vertex is at most h. Exploiting information on the direction of movements might allow more efficient algorithms for this problem when the underlying undirected graph does not have very low treewidth; a natural first question would be to consider whether there exists an efficient algorithm to solve this problem on directed acyclic graphs.

Finally, based on our investigation of the treewidth of real-world animal trade networks, it is natural to ask what other relevant problems can be solved efficiently on graphs of bounded treewidth. For example, we might wish to delete edges to achieve membership in a more complicated graph class (for example, some class of graphs on which intervention strategies used in the event of a disease outbreak are likely to be effective); alternatively, if deleting edges to achieve a small component size is too costly to be practical in some situations, we might wish to consider more relaxed criteria that nevertheless retain some desirable properties.

Acknowledgements

We are very grateful to Ivaylo Valkov for his assistance in implementing this algorithm as part of a summer research project, and to EPIC: Scotland's Centre of Expertise on Animal Disease Outbreaks, which supported JE for part of her work on this project.

References

1. R.M. Anderson, S. Gupta, and W. Ng, *The significance of sexual partner contact networks for the transmission dynamics of HIV*, Journal of Acquired Immune Deficiency Syndromes and Human Retrovirology **3** (1990), 417–429.
2. S. Arnborg, D. G. Corneil, and A. Proskurowski, *Complexity of finding embeddings in a k-tree*, SIAM J. Alg. Disc. Meth. **8** (1987), 277–284.
3. S. Arnborg, J. Lagergren, and D. Sesse, *Easy problems for tree-decomposable graphs*, Journal of Algorithms **12** (1991), 308–340.
4. Hans L. Bodlaender, *A linear time algorithm for finding tree-decompositions of small treewidth*, Proceedings of the Twenty-fifth Annual ACM Symposium on Theory of Computing (New York, NY, USA), STOC '93, ACM, 1993, pp. 226–234.
5. Leizhen Cai, *Fixed-parameter tractability of graph modification problems for hereditary properties*, Inf. Process. Lett. **58** (1996), no. 4, 171–176.
6. B. Courcelle and M. Mosbah, *Monadic second-order evaluations on tree-decomposable graphs*, Theoretical Computer Science **109** (1993), no. 12, 49 – 82.
7. Pål Grønås Drange, Markus Sortland Dregi, and Pim van 't Hof, *On the computational complexity of vertex integrity and component order connectivity*, Algorithms and Computation (Hee-Kap Ahn and Chan-Su Shin, eds.), Lecture Notes in Computer Science, vol. 8889, Springer International Publishing, 2014, pp. 285–297 (English).
8. Ehab S. El-Mallah and Charles J Colbourn, *The complexity of some edge deletion problems*, IEEE Trans. Circuits and Systems **3** (1988), 354–362.
9. Jessica Enright and Kitty Meeks, *Deleting edges to restrict the size of an epidemic*, arXiv:1504.05773 [cs.DS], 2015.
10. Esha Ghosh, Sudeshna Kolay, Mrinal Kumar, Pranabendu Misra, Fahad Panolan, Ashutosh Rai, and M.S. Ramanujan, *Faster parameterized algorithms for deletion to split graphs*, Algorithm Theory SWAT 2012 (FedorV. Fomin and Petteri Kaski, eds.), Lecture Notes in Computer Science, vol. 7357, Springer Berlin Heidelberg, 2012, pp. 107–118 (English).
11. Paul W. Goldberg, Martin C. Golumbic, Haim Kaplan, and Ron Shamir, *Four strikes against physical mapping of DNA*, Journal of Computational Biology **2** (1993), 139–152.
12. Martin Charles Golumbic, *Algorithmic graph theory and perfect graphs*, vol. 57, Elsevier, 2004.
13. Daniel Gross, Monika Heinig, Lakshmi Iswara, L. William Kazmierczak, Kristi Luttrell, John T. Saccoman, and Charles Suffel, *A survey of component order connectivity models of graph theoretic networks*, SWEAS Trans. Math. **12** (2013), no. 9.
14. Jiong Guo, *Problem kernels for NP-complete edge deletion problems: Split and related graphs*, Algorithms and Computation (Takeshi Tokuyama, ed.), Lecture Notes in Computer Science, vol. 4835, Springer Berlin Heidelberg, 2007, pp. 915–926 (English).

15. Rowland R. Kao, Darren M. Green, Jethro Johnson, and Istvan Z. Kiss, *Disease dynamics over very different time-scales: foot-and-mouth disease and scrapie on the network of livestock movements in the UK*, Journal of The Royal Society Interface **4** (2007), no. 16, 907–916.
16. Ton Kloks, *Treewidth*, Lecture Notes in Computer Science, vol. 842, Springer-Verlag, Berlin, 1994, Computations and approximations.
17. John M. Lewis and Mihalis Yannakakis, *The node-deletion problem for hereditary properties is NP-complete*, Journal of Computer and System Sciences **20** (1980), no. 2, 219 – 230.
18. Angsheng Li and Linqing Tang, *The complexity and approximability of minimum contamination problems*, Theory and Applications of Models of Computation (Mitsunori Ogihara and Jun Tarui, eds.), Lecture Notes in Computer Science, vol. 6648, Springer Berlin Heidelberg, 2011, pp. 298–307 (English).
19. F. Margot, *Some complexity results about threshold graphs*, Discrete Applied Mathematics **49** (1994), no. 1:3, 299 – 308, Special Volume Viewpoints on Optimization.
20. A. Mitchell, D. Bourn, J. Mawdsley, W. Wint, R. Clifton-Hadley, and M. Gilbert, *Characteristics of cattle movements in britain: an analysis of records from the cattle tracing system*, Animal Science **80** (2005), 265–273.
21. Assaf Natanzon, Ron Shamir, and Roded Sharan, *Complexity classification of some edge modification problems*, Discrete Applied Mathematics **113** (2001), no. 1, 109 – 128, Selected Papers: 12th Workshop on Graph-Theoretic Concepts in Computer Science.
22. Neil Robertson and P.D Seymour, *Graph minors. III. planar tree-width*, Journal of Combinatorial Theory, Series B **36** (1984), no. 1, 49 – 64.
23. Toshimasa Watanabe, Tadashi Ae, and Akira Nakamura, *On the NP-hardness of edge-deletion and -contraction problems*, Discrete Applied Mathematics **6** (1983), no. 1, 63 – 78.
24. Mihalis Yannakakis, *Node-and edge-deletion NP-complete problems*, Proceedings of the Tenth Annual ACM Symposium on Theory of Computing (New York, NY, USA), STOC '78, ACM, 1978, pp. 253–264.