ARTICLE IN PRESS

Available online at www.sciencedirect.com

ScienceDirect

Discrete Applied Mathematics ▮ (▮▮▮▮) ▮▮▮–▮▮▮

DISCRETE APPLIED MATHEMATICS

www.elsevier.com/locate/dam

Minimum cycle cover and Chinese postman problems on mixed graphs with bounded tree-width

Cristina G. Fernandes[a,*], Orlando Lee[b], Yoshiko Wakabayashi[a]

[a] *Universidade de São Paulo, Brazil*
[b] *Universidade Estadual de Campinas, Brazil*

Received 25 November 2003; received in revised form 5 July 2007; accepted 22 October 2007

Abstract

Let $M = (V, E, A)$ be a mixed graph with vertex set V, edge set E and arc set A. A cycle cover of M is a family $\mathcal{C} = \{C_1, \ldots, C_k\}$ of cycles of M such that each edge/arc of M belongs to at least one cycle in $\mathcal{C}$. The weight of $\mathcal{C}$ is $\sum_{i=1}^{k} |C_i|$. The *minimum cycle cover problem* is the following: given a strongly connected mixed graph M without bridges, find a cycle cover of M with weight as small as possible. The *Chinese postman problem* is: given a strongly connected mixed graph M, find a minimum length closed walk using all edges and arcs of M. These problems are NP-hard. We show that they can be solved in polynomial time if M has bounded tree-width.

Keywords: Tree-width; Polynomial algorithms; Cycle cover; Chinese postman problem; Mixed graphs

1. Introduction

Mixed graphs generalize the notion of digraphs and graphs in the sense that they may contain (undirected) edges as well as (directed) arcs. A **mixed graph** is a triple $M = (V, E, A)$ where V is a finite set of vertices, E is a finite set of edges and A is a finite set of arcs. When $E = \emptyset$ we say that M is a digraph, and when $A = \emptyset$ we say that M is a graph. Note that these consist of *simple* mixed graphs, that is, loops and parallel arcs/edges are not allowed.

A **cycle cover** of a mixed graph $M = (V, E, A)$ is a family $\mathcal{C} = \{C_1, \ldots, C_k\}$ of cycles of M such that each edge or arc of M belongs to at least one cycle in $\mathcal{C}$. We define the **weight** of $\mathcal{C}$ as the sum of the lengths of the cycles in $\mathcal{C}$, that is, $\sum_{i=1}^{k} |C_i|$.

The MINIMUM CYCLE COVER PROBLEM (MCCP) consists of the following: given a bridgeless strongly connected mixed graph M, find a cycle cover of M of minimum weight.

This problem is NP-hard if M is an arbitrary planar mixed graph [15]. It is well-studied when M is a graph: Thomassen [19] showed that this case is NP-hard. Also, this case is related to the well-known cycle double cover conjecture [14] and the Chinese postman problem [10].

* Corresponding address: Institute of Mathematics and Statistics, University of Sao Paulo, Rua do Matao 1010, 05508-090 Sao Paulo, SP, Brazil. Tel.: +55 11 3091 5709; fax: +55 11 3091 6134.
E-mail addresses: cris@ime.usp.br (C.G. Fernandes), lee@ic.unicamp.br (O. Lee), yw@ime.usp.br (Y. Wakabayashi).

0166-218X/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.dam.2007.10.032

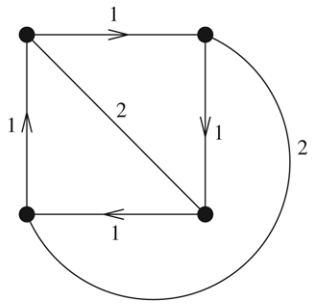

Fig. 1. The numbers denote the number of times each edge/arc is used on a postman walk. This walk cannot be decomposed into a cycle cover of same weight. This postman walk has minimum length, while the weight of a minimum cycle cover is 9.

Let M be a strongly connected mixed graph. A **postman walk** in M is a closed walk that contains all edges and arcs of M. Any cycle cover of weight k can be converted into a postman walk of length k, but the converse is not true. (See Fig. 1.)

The CHINESE POSTMAN PROBLEM (CPP) in mixed graphs is a variant of MCCP: given a strongly connected mixed graph M, find a postman walk in M of minimum length.

We know that CPP can be solved in polynomial time in graphs [10] and in digraphs [11]. On the other hand, Papadimitriou [17] proved that CPP is NP-hard on planar mixed graphs.

In this paper, we study the complexity of MCCP and CPP on mixed graphs with a certain structural parameter – tree-width – bounded by a constant. The tree-width of a graph G can be defined using the notion of a tree-decomposition of G. The study of tree-decompositions started in the eighties and had several consequences in two areas: graph theory and design of algorithms. In the first, tree-decompositions were fundamental in the proof of Robertson and Seymour's result on graph minors: in every infinite sequence of graphs, there are two graphs such that one is a minor of the other. In the second, this concept was successfully explored in the design of polynomial-time algorithms for several NP-hard problems on graphs with bounded tree-width. We show a polynomial-time algorithm for MCCP and for CPP on mixed graphs with bounded tree-width.

It is known that any graph problem that can be expressed as a formula in (specific) extensions of monadic second order (MSO) logic can be solved in polynomial (many times, linear) time on graphs with bounded tree-width [2,3, 7]. Many graph problems can be formulated as such formulas. Indeed, Arnborg, Lagergren, and Seese [2, Sec. 3] explicitly considered many well-known graph problems taken from Garey and Johnson [13] and proved they can be formulated as such. We believe the two problems we address cannot be expressed in these (specific) extensions of MSO logic. Next we present some reasons for this. We refer to MCCP only but the same observations hold for CPP.

In Fig. 2, we show a family of graphs with bounded tree-width, where the weight of a minimum cycle cover is quadratic on the size of the graph. In these graphs, both the number of cycles and the length of each cycle in any minimum cycle cover is not bounded by a fixed constant, even though the graphs have bounded tree-width. The structure we are searching for in the graph is a collection of subsets of edges (the cycles), each of the subsets having size possibly dependent on the size of the graph, and with the collection possibly also of size dependent on the size of the graph. In most of the problems that can be expressed in MSO logic, there is a fixed bound on some parameter of the searched structure (number of sets of the structure, size of the sets of the structure, etc) [2]. However, because of the previous, we did not find a way to express MCCP in (such extensions of) MSO logic. Courcelle and Mosbah [8, page 64] also gave some examples of problems that cannot be expressed in MSO logic, that have some elements in common with MCCP, reinforcing this impression we have.

Many of the polynomial-time algorithms for graphs with bounded tree-width follow a straightforward dynamic programming approach [5]. Although ours is also a dynamic programming algorithm, it is not a straightforward application of the general framework. These two problems seem to require a more elaborate application of the framework. Maybe this is related to the fact that these problems might not be expressible in MSO logic. For them, one has to show that the partial solutions, which are more complex structures, have some nontrivial properties. Also, one has to make a careful choice of the data to be carried along the dynamic programming, to guarantee the polynomial bound on the running time of the resulting algorithm. We elaborate more on this when the involved concepts are introduced.

Please cite this article in press as: C.G. Fernandes, et al., Minimum cycle cover and Chinese postman problems on mixed graphs with bounded tree-width, Discrete Applied Mathematics (2008), doi:10.1016/j.dam.2007.10.032

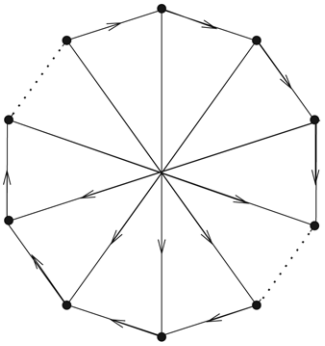

Fig. 2. A family of graphs of tree-width bounded by 4: for each even n, there is a graph in the family consisting of a directed cycle C on n vertices and arcs between antipodal vertices in C. There are only two different types of directed cycles in such graphs: the cycle C and the cycles that contain exactly one chord of C.

Specifically, in this paper we prove the following theorems.

Theorem 1. CPP *is solvable in polynomial time on mixed graphs with bounded tree-width.*

Theorem 2. MCCP *is solvable in polynomial time on mixed graphs with bounded tree-width.*

The paper is organized as follows. Next we introduce some notation and give some definitions. In Section 2 we prove Theorem 1 and in Section 3 we prove Theorem 2.

1.1. Definitions and notation

Most of the concepts defined for graphs and digraphs (see [6]) can be extended in a natural way to mixed graphs. We assume the reader is familiar with them. We present here just a few concepts, to establish the notation.

A **walk** in a mixed graph is a sequence $W := (v_1, l_1, \ldots, v_{k-1}, l_{k-1}, v_k)$, where each v_i is a vertex and either $l_i = v_{i-1}v_i$ is an edge or $l_i = (v_{i-1}, v_i)$ is an arc. Sometimes W is denoted simply by $(v_1, \ldots, v_k)$. The **length** of W is $k-1$. We say that W is a walk **from** v_1 **to** v_k or that W **starts** at v_1 and **ends** at v_k. Also, we say that v_1 and v_k are the **ends** of W. If a walk starts and ends at the same vertex, we say it is **closed**. A **cycle** is a closed walk $(v_1, \ldots, v_k)$, where $v_i \neq v_j$, for all $1 \leq i < j < k$. A **path** is a walk $(v_1, \ldots, v_k)$, where $v_i \neq v_j$, for all $1 \leq i < j \leq k$.

A mixed graph is **strongly connected** if for any ordered pair of vertices (u, v), there is a path from u to v.

Given a mixed graph M, the **underlying undirected graph** of M is the graph obtained from M by replacing every arc in M by an edge with the same ends.

Let G be a graph and $\Theta := \left(T, (W_t)_{t \in V(T)}\right)$ be a pair consisting of a tree T and a multiset whose elements W_t, indexed by the vertices of T, are subsets of $V(G)$. For a vertex v of G, we denote by F_v the subgraph of T induced by those vertices t of T for which W_t contains v. Then Θ is called a **tree-decomposition** of G if the following two conditions hold:

(1) For every edge $e = xy$ of G, there is a vertex t of T such that $\{x, y\} \subseteq W_t$;
(2) For every vertex v of G, the subgraph F_v of T is a tree.

The **width** of Θ is the maximum, over all vertices t of T, of $|W_t| - 1$; and the **tree-width** of G is the least width of any tree-decomposition of G. We refer to [9,18] for properties and other results on tree-decompositions.

We may assume that the tree T in any tree-decomposition is binary. If this is not the case, one can modify T, introducing new vertices in such a way that the modified tree-decomposition preserves the width of the original one.

For a mixed graph M, the concept of tree-decomposition refers to the tree-decomposition of its underlying undirected graph G. Let $\Theta := \left(T, (W_t)_{t \in V(T)}\right)$ be a tree-decomposition of G. For each vertex t of T, choose a set $E(W_t)$ of edges of M and a set $A(W_t)$ of arcs of M such that all edges and arcs in $E(W_t)$ and $A(W_t)$ have their two ends in W_t and every edge or arc in M appears in exactly one of these sets. These sets form a partition of $E(M) \cup A(M)$, which we call an **associated partition** of Θ. (Note that it is not unique.)

Given a mixed graph $M = (V, E, A)$ and a subset X of V, we denote by $\delta(X)$ the set of edges of M with exactly one end in X. We denote by $\delta^+(X)$ the set of arcs of M starting at a vertex in X and ending at a vertex not in X. Also,

Please cite this article in press as: C.G. Fernandes, et al., Minimum cycle cover and Chinese postman problems on mixed graphs with bounded tree-width, Discrete Applied Mathematics (2008), doi:10.1016/j.dam.2007.10.032

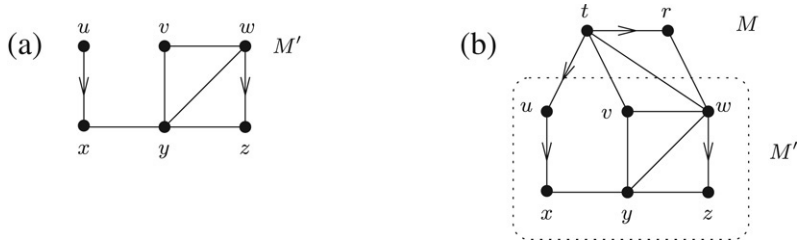

Fig. 3. (a) For $S = \{u, v, w\}$, $d(u) = d(w) = 1$, and $d(v) = -2$, the pair $(\mathcal{P}, \mathcal{C})$ is a d-realization in M' for $\mathcal{P} = \{(u, x, y, v), (w, z, y, v)\}$ and $\mathcal{C} = \{(v, w, y)\}$. (b) The collection $\mathcal{C} = \{(u, x, y, v, t), (w, z, y, v, t, w), (v, w, y), (r, w, t)\}$ is a cycle cover of M whose restriction to M' is the pair $(\mathcal{P}, \mathcal{C})$.

let $\delta^-(X) := \delta^+(V \setminus X)$. A **bridge** in M is an edge e of M such that $\delta(X) \cup \delta^+(X) \cup \delta^-(X) = \{e\}$, for some subset X of V. We say M is **Eulerian** if $|\delta(X)| - \|\delta^+(X)| - |\delta^-(X)\|$ is an even non-negative number, for any subset X of V. The following theorem [4,12] generalizes the results known for graphs and digraphs.

Theorem 3. *Any Eulerian mixed graph can be partitioned into a set of edge and arc disjoint cycles.* ■

2. The Chinese postman problem

In this section, we prove Theorem 1, that is, we present a polynomial-time algorithm for CPP on mixed graphs with bounded tree-width. To simplify the description of the algorithm, we represent a solution of CPP for a mixed graph $M = (V, E, A)$ in an unusual form. Let P be a postman walk in M. Consider the supergraph $H = (V_H, E_H, A_H)$ of M for which $V_H := V$ and, for each edge (arc) e of M, if e appears t times in P, then there are t copies of e in E_H (A_H). Clearly H is an Eulerian mixed graph. Therefore, by Theorem 3, H can be written as a union of edge and arc disjoint cycles, say, $C_1, \ldots, C_q$. Note that some of these cycles might not correspond to cycles in M. This happens exactly when a cycle C_i consists of two copies of the same edge of M. In this case, we say C_i is a **pseudocycle** in M. On the other hand, if $C_1, \ldots, C_q$ is a collection of cycles and pseudocycles of M whose union contains all edges and arcs of M, then the mixed graph given by $\bigcup_{i=1}^{q} C_i$ is Eulerian and corresponds to a postman walk in M.

Define the **weight** of a collection $C_1, \ldots, C_q$ of cycles and pseudocycles as $\sum_{i=1}^{q} |C_i|$. We can formulate CPP as the problem of finding a minimum-weight collection of cycles and pseudocycles containing all edges and arcs of M. From now on, we consider this new formulation of CPP.

Let $m := |E \cup A|$. Observe that, if $C_1, \ldots, C_q$ is an optimal solution for CPP then $q \leq m$. Indeed, suppose $C_1, \ldots, C_q$ is an optimal solution for CPP. In each C_i, there exists an edge or arc that appears in no other (pseudo)cycle C_j (otherwise, by removing C_i we would get a better solution). As a consequence, no edge or arc of M appears in more than m (pseudo)cycles.

Before describing the algorithm, we need some additional notation. Let S be a subset of V. An **S-configuration** is a function $d : S \to \{-m, \ldots, m\}$ such that $\sum_{v \in S} d(v) = 0$. Let M' be a subgraph of M and d be an S-configuration, for some set S of vertices of M'. If $\mathcal{P}$ is a collection of paths in M' with ends in S and $\mathcal{C}$ is a collection of (pseudo)cycles of M', then the pair $(\mathcal{P}, \mathcal{C})$ is a **d-realization** in M' if

- for each v in S, the number of paths in $\mathcal{P}$ that start at v minus the number of paths in $\mathcal{P}$ that end at v is exactly $d(v)$;
- each edge or arc of M' belongs to some path in $\mathcal{P}$ or to some (pseudo)cycle in $\mathcal{C}$.

We give an example of d-realization in Fig. 3(a).

The intuition behind the concepts of S-configuration and d-realization is the following. Consider a cycle cover $\mathcal{C}$ for M. The part of the cycle cover within a subgraph M' of M is a d-realization in M' for some set S of vertices of M' and some function $d : S \to \{-m, \ldots, m\}$ such that $\sum_{v \in S} d(v) = 0$. (See Fig. 3(b).) Indeed, when we take the restriction of the cycles in $\mathcal{C}$ to M', we obtain a collection of paths and cycles (some paths might be part of the same cycle). The endpoints of the paths define the set S. The difference between the number of paths starting and ending in a vertex defines the function d. The concept of d-realization is a formal definition of what we refered to as partial solution. The use of these concepts in the algorithm is described ahead.

Please cite this article in press as: C.G. Fernandes, et al., Minimum cycle cover and Chinese postman problems on mixed graphs with bounded tree-width, Discrete Applied Mathematics (2008), doi:10.1016/j.dam.2007.10.032

We define the **weight** of a d-realization $(\mathcal{P}, \mathcal{C})$ as the quantity given by $\sum_{P\in\mathcal{P}} |P| + \sum_{C\in\mathcal{C}} |C|$. We say a d-realization in a graph is **optimal** if it has the smallest possible weight.

When M' is a digraph with n' vertices and m' arcs, an optimal d-realization can be found in $O((m' \log n')(m' + n' \log n'))$ time using a minimum cost flow algorithm [1,16].

Let $(T, (W_t)_{t\in V(T)})$ be a tree-decomposition of (the underlying undirected graph of) M with tree-width k. Let $(E(W_t))_{t\in V(T)}$ and $(A(W_t))_{t\in V(T)}$ be an associated partition of Θ. Note that each set in this partition has size bounded by a function which depends only on k.

Root the tree T at an arbitrary vertex r in T. For each vertex t of T, denote by M_t the subgraph of M such that $V(M_t)$, $E(M_t)$ and $A(M_t)$ are the union, over all descendents v of t in T, of the sets W_v, $E(W_v)$ and $A(W_v)$ respectively. Note that the restriction of an optimal solution $\mathcal{C}^*$ of CPP to M_t corresponds to a collection $\mathcal{P}$ of paths and a collection $\mathcal{C}$ of (pseudo)cycles in M_t. It is not difficult to see that $(\mathcal{P}, \mathcal{C})$ corresponds to an optimal d-realization in M_t, for some W_t-configuration d. Moreover, if $(\mathcal{P}', \mathcal{C}')$ is an optimal d-realization in M_t, one can get an optimal solution of CPP in M by combining $(\mathcal{P}', \mathcal{C}')$ with the restriction of $\mathcal{C}^*$ to the graph $M - (E(M_t) \cup A(M_t))$.

The algorithm for CPP is based on the following strategy. For each vertex t of T and each W_t-configuration d, find an optimal d-realization in M_t. The algorithm solves this initially for the leaves of T and goes up in T until it reaches its root r. Note that in the particular case of the root, a d-realization for $d \equiv 0$ is a collection of (pseudo)cycles covering all arcs and edges of M, that is, it is a feasible solution for CPP. Hence, solving CPP is equivalent to finding an optimal d-realization in M for the W_r-configuration $d \equiv 0$.

2.1. Description of the algorithm

From now on, let us assume that M has tree-width at most k, where k is a fixed constant. Also, let $(T, (W_t)_{t\in V(T)})$ be a tree-decomposition of M of width at most k, with T binary. Clearly, as k is a constant, $|E(W_t)|$ and $|A(W_t)|$ are bounded by a constant, for all t.

For each vertex t of T, the algorithm builds a list of partial solutions consisting of triples $(d, \mathcal{P}(d), \mathcal{C}(d))$ where d is a W_t-configuration and $(\mathcal{P}(d), \mathcal{C}(d))$ is an optimal d-realization in M_t. Since $|W_t| \leq k + 1$, there are at most $(2m + 1)^{k+1}$ different W_t-configurations for each t.

The algorithm starts building these lists for the leaves of T. This can be done in polynomial time, because, as $|W_t| \leq k + 1$, one can find an optimal d-realization by enumeration, for each W_t-configuration d. Going up the tree, the algorithm uses the lists of the children of a vertex to build the list for this vertex. This process is repeated and halts only when it reaches the root of the tree T.

Let us describe how the list for a vertex is obtained from the lists of its children. Let t be a vertex of the tree T and d be a W_t-configuration. Let t_1 and t_2 be the two children of t (the case where t has only one child is similar) and let $\mathcal{L}_1$ and $\mathcal{L}_2$ be the lists of partial solutions for t_1 and t_2, respectively. Let $\pi := (d_i, \mathcal{P}(d_i), \mathcal{C}(d_i))$ be a partial solution in $\mathcal{L}_i$, for $i = 1, 2$. A **d-extension** for the pair (π_1, π_2) is a d-realization $(\mathcal{P}, \mathcal{C})$ in M_t whose restriction to the subgraph M_{t_i} corresponds to the partial solution $(\mathcal{P}(d_i), \mathcal{C}(d_i))$, for $i = 1, 2$. If, for each pair (π_1, π_2), we can compute an optimal d-extension for (π_1, π_2) (if one exists), then an optimal d-realization in M_t is simply an optimal d-extension for a pair that gives the smallest possible weight (considering all such pairs).

But how can we compute an optimal d-extension for a specific pair (π_1, π_2)? Each d_i is defined on W_{t_i}. A d-extension for the pair (π_1, π_2) exists only if the support of both d_1 and d_2 is contained in W_t. Extend d_1 and d_2 so that they are defined on W_t. Let $d_0 := d - (d_1 + d_2)$ and let $H := (W_t, E(W_t), A(W_t))$. Thus, to compute an optimal d-extension it suffices to find an optimal d_0-realization in H.

Let $(\mathcal{P}, \mathcal{C})$ be an optimal d_0-realization in H. For each $e = uv$ in $E(W_t)$, exactly one of the following holds:

(a) every element in $\mathcal{P} \cup \mathcal{C}$ containing e traverses e from u to v;
(b) every element in $\mathcal{P} \cup \mathcal{C}$ containing e traverses e from v to u;
(c) there are exactly two elements in $\mathcal{P} \cup \mathcal{C}$ containing e, one of them traverses e from u to v, and the other traverses e from v to u.

In order to compute an optimal d_0-realization in H, we consider all digraphs obtained from H by replacing each edge $e = uv$ of H by either (a) an arc (u, v), or (b) an arc (v, u), or (c) two arcs (u, v) and (v, u). For each one of these digraphs, we compute an optimal d_0-realization (using a minimum cost flow algorithm), and we select a d_0-realization with minimum weight. Note that the number of digraphs considered is $3^{|E(W_t)|} \leq 3^{k(k+1)/2}$. So the

Please cite this article in press as: C.G. Fernandes, et al., Minimum cycle cover and Chinese postman problems on mixed graphs with bounded tree-width, Discrete Applied Mathematics (2008), doi:10.1016/j.dam.2007.10.032

total time used to find an optimal d-extension of a pair (π_1, π_2) is $O(3^{k(k+1)/2}(|E(W_t) \cup A(W_t)| \log |W_t|)(|E(W_t) \cup A(W_t)| + |W_t| \log |W_t|)) = O(3^{k(k+1)/2}k^4 \log k)$.

We conclude that, given d and the lists $\mathcal{L}_1$ and $\mathcal{L}_2$ of partial solutions for t_1 and t_2, one can find an optimal d-extension in M_t in time $O(|\mathcal{L}_1||\mathcal{L}_2|3^{k(k+1)/2}k^4 \log k)$. The length of each list $\mathcal{L}_i$ is at most the number of W_{t_i}-configurations, which is at most $(2m+1)^{k+1}$. Thus, one can find an optimal d-realization in M_t in polynomial time. Furthermore, one can build in polynomial time the list of all partial solutions for a vertex t from the lists for its children.

From the above, CPP in mixed graphs with tree-width bounded by k can be solved in polynomial time, completing the proof of Theorem 1. Unfortunately, due to its time complexity, the algorithm is only useful in practice for small values of k. One can make the algorithm a bit faster by including in the list of partial solutions for a vertex t only the partial solutions with support in $W_p \cap W_t$, where p is the parent of t in the tree T, but this does not reduce the complexity of the algorithm.

3. The minimum cycle cover problem

In this section, we prove Theorem 2. The algorithm for MCCP follows the same lines of the algorithm described above. The difference lies in how we define a partial solution for MCCP. Partial solutions for CPP include pseudocycles and these are not allowed in a solution for MCCP. Unfortunately, it is not enough simply to omit the pseudocycles from the partial solutions, because they appear naturally when we put together two partial solutions. The change has to be more radical. More information is needed so that we can put two partial solutions together without forming pseudocycles.

For the MCCP, an **S-configuration** is a function $d : \{-1, 0, 1\}^S \to \{0, \ldots, m\}$. A **path-family** in M is a set of pairwise vertex-disjoint paths in M. Let $F := \{P_1, \ldots, P_t\}$ be a path-family in M, and assume that, for each $i = 1, \ldots, t$, P_i is a path from u_i to v_i. For z in $\{-1, 0, 1\}^S$, we say F **covers** z if $\{s \in S : z(s) = 1\} = \{u_1, \ldots, u_t\}$, and $\{s \in S : z(s) = -1\} = \{v_1, \ldots, v_t\}$.

Let d be an S-configuration, where S is a set of vertices of a subgraph M' of M. If $\mathcal{F}$ is a collection of path-families in M' (with their paths having ends in S) and $\mathcal{C}$ is a collection of cycles in M', we say $(\mathcal{F}, \mathcal{C})$ is a **d-realization** in M' if

- there exists a partition $\mathcal{F} := \bigcup_{z \in \{-1,0,1\}^S} \mathcal{F}_z$ such that $\mathcal{F}_z$ consists of exactly $d(z)$ path-families in M' that cover z;
- each edge or arc of M' belongs to a path in some path-family in $\mathcal{F}$ or to some cycle in $\mathcal{C}$.

The **weight** of a d-realization $(\mathcal{F}, \mathcal{C})$ is defined as the quantity given by $\sum_{F \in \mathcal{F}} \sum_{P \in F} |P| + \sum_{C \in \mathcal{C}} |C|$. A d-realization in a graph is **optimal** if it has the smallest possible weight.

Note that the restriction of a path-family to a subgraph M_t is also a path-family and the restriction of a cycle to M_t is either a cycle or a path-family. Moreover, if t is an arbitrary vertex of T, then the restriction of an optimal solution $\mathcal{C}^*$ for MCCP to the subgraph M_t corresponds to a collection $\mathcal{F}$ of path-families in M_t together with a collection $\mathcal{C}$ of cycles in M_t. It is not difficult to see that $(\mathcal{F}, \mathcal{C})$ corresponds to an optimal d-realization for some W_t-configuration d. Also, if $(\mathcal{F}', \mathcal{C}')$ is another optimal d-realization in M_t, then the restriction of $\mathcal{C}^*$ to the graph $M - (E(M_t) \cup A(M_t))$ combined with $(\mathcal{F}', \mathcal{C}')$ results in another optimal solution for MCCP in M. Note that, when $t = r$ and $d \equiv 0$, a d-realization in M_t $(= M)$ is a collection of cycles covering all arcs and edges of M, that is, it is a feasible solution of MCCP. Thus, solving MCCP is equivalent to finding an optimal d-realization in M for the W_r-configuration $d \equiv 0$.

The algorithm works as the previous one. Given a tree-decomposition $(T, (W_t)_{t \in V(T)})$ of M, with width at most k, where T is a binary tree, it builds, for each vertex t of T, a list of partial solutions for the MCCP in M_t. Each partial solution consists of a triple $(d, \mathcal{F}(d), \mathcal{C}(d))$, where d is a W_t-configuration and $(\mathcal{F}(d), \mathcal{C}(d))$ is an optimal d-realization. Since $|W_t| \leq k+1$, there are at most $(m+1)^{3^{k+1}}$ different W_t-configutations, for each t. For the leaves of T, the lists are built directly by brute force. For each internal vertex t, let t_1 and t_2 be its two children. (The case where t has only one child is similar.) Let us describe how the list for t is obtained from the lists for t_1 and t_2. More specifically, let us show how to extend a pair of partial solutions (π_1, π_2), with $\pi_i := (d_i, \mathcal{F}(d_i), \mathcal{C}(d_i))$ for $i = 1, 2$, to obtain a d-realization $(\mathcal{F}, \mathcal{C})$ in M_t. Such a d-realization is called a **d-extension** of (π_1, π_2) and is such that its restriction to M_{t_i} is exactly π_i for $i = 1, 2$. Specifically, $(\mathcal{F}, \mathcal{C})$ satisfies

- $\mathcal{C}(d_1) \cup \mathcal{C}(d_2) \subseteq \mathcal{C}$;

Please cite this article in press as: C.G. Fernandes, et al., Minimum cycle cover and Chinese postman problems on mixed graphs with bounded tree-width, Discrete Applied Mathematics (2008), doi:10.1016/j.dam.2007.10.032

– the set of arcs and edges of each path-family in $\mathcal{F}$ and each cycle in $\mathcal{C} \setminus (\mathcal{C}(d_1) \cup \mathcal{C}(d_2))$ can be decomposed into L, F_1 and F_2, where L is a path-family in the graph $(W_t, E(W_t) \cup A(W_t))$ and, for $i = 1, 2$, either $F_i = \emptyset$ or $F_i \in \mathcal{F}(d_i)$;
– each path-family of $\mathcal{F}(d_i)$, $i = 1, 2$, appears in the decomposition of exactly either one path-family in $\mathcal{F}$ or one cycle in $\mathcal{C} \setminus (\mathcal{C}(d_1) \cup \mathcal{C}(d_2))$.

For $i = 1, 2$, let z_i in $\{-1, 0, 1\}^{W_t}$ be such that $d_i(z_i) > 0$, and let F_i be a path-family in $\mathcal{F}(d_i)$ that covers z_i. Let L be a path-family in $(W_t, E(W_t), A(W_t))$. Then L, F_1 and F_2 determine, besides possibly some cycles, a path-family in M_t covering some z in $\{-1, 0, 1\}^{W_t}$. Note that z depends only on the choice of L, z_1 and z_2. (Here we used the fact that the paths in the path-families are disjoint and F_1 and F_2 are contained in M_{t_1} and M_{t_2}, respectively.) So we say simply that L, z_1 and z_2 (instead of L, F_1 and F_2) determine a path-family in M_t and we call the triple (L, z_1, z_2) a **candidate path-family in** M_t. Similarly, for each z in $\{-1, 0, 1\}^{W_t}$ such that $d_i(z) > 0$, for $i = 1$ or 2, we can do the same with only L and z, and we also say (L, z) is an candidate path-family in M_t. A candidate path-family in M_t **covers** z if it determines a path-family in M_t that covers z. We can do the same with a cycle (instead of a path-family) in M_t. In this case, we say (L, z_1, z_2) (or (L, z)) is a **candidate cycle in** M_t. Each candidate path-family (cycle) represents several path-families (cycles) in M_t.

We reduce the problem of finding an optimal d-extension of (π_1, π_2) to the problem of enumerating all feasible solutions of an integer linear program of constant size. We use candidate path-families (cycles) to avoid enumerating all possible path-families and cycles in M_t. In this way, we need only to consider triples or pairs of subsets of a bounded-size set.

Let us describe the above mentioned integer linear program.

Let D_1 and D_2 be two disjoint copies of $\{-1, 0, 1\}^{W_t}$. Let $\mathbf{P}$ be a matrix with rows indexed by $R := E(W_t) \cup A(W_t) \cup D_1 \cup D_2$, whose columns are the incidence vectors of candidate path-families in M_t. Analogously, let $\mathbf{C}$ be a matrix with rows indexed by R, whose columns are the candidate cycles in M_t. For z in R, denote by $\mathbf{P}_z$ (respectively $\mathbf{C}_z$) the row of $\mathbf{P}$ (respectively $\mathbf{C}$) corresponding to z. Note that the size of R is $|E(W_t)| + |A(W_t)| + 2(3^{|W_t|}) \leq k(k+1)/2 + k(k+1) + 2(3^{k+1})$. Also, the number of columns of these two matrices is bounded by $2^{|E(W_t) \cup A(W_t)|}|D_1||D_2| \leq 2^{3k(k+1)/2} + 3^{2(k+1)}$. So, $\mathbf{P}$ and $\mathbf{C}$ have constant size.

For z in $\{-1, 0, 1\}^{W_t}$, denote by $\mathcal{K}(z)$ the set of candidate path-families in M_t that cover z. Consider pairs (x, y), where x is a non-negative integer vector indexed by candidate path-families in M_t and y is a non-negative integer vector indexed by candidate cycles in M_t, satisfying the following conditions:

$$\begin{aligned}
&\sum_{F \in \mathcal{K}(z)} x_F = d(z) \quad \forall z \in \{-1, 0, 1\}^{W_t};\\
&\mathbf{P}_e x + \mathbf{C}_e y \geq 1 \qquad \forall e \in E(W_t) \cup A(W_t);\\
&\mathbf{P}_z x + \mathbf{C}_z y = d_1(z) \quad \forall z \in D_1;\\
&\mathbf{P}_z x + \mathbf{C}_z y = d_2(z) \quad \forall z \in D_2.
\end{aligned}$$

Let us show that there is a correspondence between integer solutions (x, y) and extensions of (π_1, π_2). Clearly, an extension of (π_1, π_2) corresponds to an integer solution (x, y). Now, suppose that (x, y) is an integer vector satisfying the integer linear program above. Let $\mathcal{F}$ be the collection of candidate path-families in M_t, with each candidate path-family F appearing exactly x_F times, and let $\mathcal{C}$ be the collection of candidate cycles in M_i, with each candidate cycle C appearing exactly y_C times. Recall that the pair $(\mathcal{F}(d_i), \mathcal{C}(d_i))$ is a d_i-realization in M_{t_i}, for $i = 1, 2$. The pair (x, y) forces the occurrence in $\mathcal{F} \cup \mathcal{C}$ of exactly $d_i(z)$ candidate path-families covering z in D_i. Replacing these candidate path-families in $\mathcal{F}, \mathcal{C}$ by path-families in $\mathcal{F}(d_1) \cup \mathcal{F}(d_2)$ for each z, we obtain a collection of path-families and cycles in M_t, which we call $\hat{\mathcal{F}}$ and $\hat{\mathcal{C}}$ respectively. It is easy to see that $(\hat{\mathcal{F}}, \hat{\mathcal{C}} \cup \mathcal{C}(d_1) \cup \mathcal{C}(d_2))$ is a d-realization in M_t.

Thus, to find an optimal extension, it is enough to find a pair (x, y) satisfying the system above and that minimizes the weight of the corresponding realization. Clearly, we may assume that the components of x and y lie in $\{0, \ldots, m\}$. So, an upper bound for the number of feasible solutions of the system is $(m+1)^{g(k)}$, where $g(k) = O(2^{3k(k+1)/2} + 3^{2(k+1)})$ is the sum of the number of columns of $\mathbf{P}$ and $\mathbf{C}$.

Therefore, given d and $\pi_i := (d_i, \mathcal{F}(d_i), \mathcal{C}(d_i))$, $i = 1, 2$, one can find an optimal d-realization (π_1, π_2) in time $O(g(k) + m^{g(k)})$. Given d and the lists $\mathcal{L}_1, \mathcal{L}_2$ of partial solutions for t_1,t_2, one can compute an optimal d-realization in M_t in time $|\mathcal{L}_1||\mathcal{L}_2|O(g(k) + m^{g(k)})$. The length of each list $\mathcal{L}_i$ is the number of W_{t_i}-configurations, which is at

Please cite this article in press as: C.G. Fernandes, et al., Minimum cycle cover and Chinese postman problems on mixed graphs with bounded tree-width, Discrete Applied Mathematics (2008), doi:10.1016/j.dam.2007.10.032

most $(m+1)^{3^{k+1}}$. Hence, one can find an optimal d-realization in M_t in polynomial time. Furthermore, one can build in polynomial time a list of partial solutions for a vertex t from the lists for its children repeating the process for all possible W_t-configurations. This concludes the proof of Theorem 2.

4. Concluding remarks

The algorithm we obtained is more of theoretical interest because of its complexity. It would be interesting to find a simpler dynamic programming algorithm, though this does not seem to be an easy task.

It would also be interesting to establish whether these two problems are in the classes of problems addressed by one of the metaresults for bounded tree-width graphs [2,8]. If the answer to this is positive, it would lead to another algorithm for these problems. If it is negative, maybe the algorithms we described inspire possible extensions of these metaresults to incorporate the two problems and others of similar nature.

We consider a relevant contribution of this paper the detailed presentation of a far from straightforward dynamic programming algorithm for two well-known problems on bounded tree-width graphs. Most of the dynamic programming algorithms for bounded tree-width graphs are not so intricate to derive. We hope the approach we described in this paper may help in the design of algorithms for other problems on bounded tree-width graphs that have a similar nature and require a more sophisticated approach.

Acknowledgments

The authors would like to thank Robin Thomas for helpful discussions, and an anonymous referee for several suggestions that helped to improve this paper. This research was partially supported by ProNEx - FAPESP/CNPq Proc. No. 2003/09925-5 (Brazil), CNPq (Proc. 490333/04-4, 478329/04-0, 478470/06-1, 307011/03-8, 308138/04-0, 301310/05-0).

References

[1] R.K. Ahuja, T.L. Magnantie, J.B. Orlin, Network Flows, Prentice-Hall, 1993.
[2] S. Arnborg, J. Lagergren, D. Seese, Easy problems for tree-decomposable graphs, J. Algorithms 12 (1991) 308–340.
[3] S. Arnborg, A. Proskurowski, Linear time algorithms for NP-hard problems restricted to partial k-trees, Discrete Appl. Math. 23 (1989) 11–24.
[4] V. Batagelj, T. Pisanski, On partially directed Eulerian multigraphs, Publ. de l'Inst. Math. Soc. 25 (1979) 16–24.
[5] H.L. Bodlaender, Dynamic programming on graphs of bounded treewidth, in: Proc. 15th International Colloquium on Automata, Languages and Programming, in: Lecture Notes in Computer Science, vol. 317, Springer-Verlag, Berlin, 1988, pp. 631–643.
[6] J.A. Bondy, U.S.R. Murty, Graph Theory with Applications, MacMillan Press, 1976.
[7] B. Courcelle, The monadic second order logic of graphs. I. Recognizable sets of finite graphs, Inform. Comput. 85 (1990) 12–75.
[8] B. Courcelle, M. Mosbah, Monadic 2nd-order evaluations on tree-decomposable graphs, Theoret. Comput. Sci. 109 (1–2) (1993) 49–82.
[9] R. Diestel, Graph Theory, second edn, in: Graduate Texts in Mathematics, vol. 173, Springer-Verlag, New York, 2000.
[10] J. Edmonds, The Chinese postman problem, Oper. Res. 13 (1965) B73–B77.
[11] J. Edmonds, E.L. Johnson, Matching, Euler tours and the Chinese postman problem, Math. Program. 5 (1973) 88–124.
[12] L.R. Ford, D.R. Fulkerson, Flows in Networks, Princeton U. Press, 1973.
[13] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, Freeman, 1979.
[14] U. Jamshy, M. Tarsi, Shortest cycle covers and the cycle double cover conjecture, J. Combin. Theory Ser. B 56 (1992) 197–204.
[15] O. Lee, Y. Wakabayashi, On the circuit cover problem for mixed graphs, Combin. Probab. Comput. 11 (2002) 43–59.
[16] J.B. Orlin, A faster strongly polynomial minimum cost flow algorithm, Oper. Res. 41 (2) (1993) 338–350.
[17] C.H. Papadimitriou, On the complexity of edge traversing, J. ACM 23 (1976) 544–554.
[18] B. Reed, Tree width and tangles: A new connectivity measure and some applications, in: Surveys in Combinatorics, 1997, London, in: London Math. Soc. Lecture Note Ser., vol. 241, Cambridge Univ. Press, Cambridge, 1997, pp. 87–162.
[19] C. Thomassen, On the complexity of finding a minimum cycle cover of a graph, SIAM J. Comput. 26 (1997) 675–677.

Please cite this article in press as: C.G. Fernandes, et al., Minimum cycle cover and Chinese postman problems on mixed graphs with bounded tree-width, Discrete Applied Mathematics (2008), doi:10.1016/j.dam.2007.10.032