Parameterized Complexity of the k-Arc Chinese Postman Problem

Gregory Gutin[1], Mark Jones[1], and Bin Sheng[1]

Royal Holloway, University of London
Egham, Surrey TW20 0EX, UK

Abstract. In the Mixed Chinese Postman Problem (MCPP), given an edge-weighted mixed graph G (G may have both edges and arcs), our aim is to find a minimum weight closed walk traversing each edge and arc at least once. The MCPP parameterized by the number of edges was known to be fixed-parameter tractable using a simple argument. Solving an open question of van Bevern et al., we prove that the MCPP parameterized by the number of arcs is also fixed-parameter tractable. Our proof is more involved and, in particular, uses a very useful result of Marx, O'Sullivan and Razgon (2013) on the treewidth of torso graphs with respect to small separators.

1 Introduction

A *mixed graph* is a graph that may contain both edges and arcs (i.e., directed edges). A mixed graph G is *strongly connected* if for each ordered pair x, y of vertices in G there is a path from x to y that traverses each arc in its direction. We provide further definitions and notation on (mainly) directed graphs in the next section.

In this paper, we will study the following problem.

Mixed Chinese Postman Problem (MCPP)
Instance: A strongly connected mixed graph $G = (V, E \cup A)$, with vertex set V, set E of edges and set A of arcs; a weight function $w : E \cup A \to \mathbb{N}_0$.
Output: A closed walk of G that traverses each edge and arc at least once, of minimum weight.

There is numerous literature on various algorithms and heuristics for MCPP; for informative surveys, see [2, 4, 8, 12, 14]. When $A = \emptyset$, we call the problem the Undirected Chinese Postman Problem (UCPP), and when $E = \emptyset$, we call the problem the Directed Chinese Postman Problem (DCPP). It is well-known that UCPP is polynomial-time solvable [7] and so is DCPP [3, 5, 7], but MCPP is NP-complete, even when G is planar with each vertex having total degree 3 and all edges and arcs having weight 1 [13]. It is therefore reasonable to believe that MCPP may become easier the closer it gets to UCPP or DCPP.

Van Bevern *et al.* [2] considered two natural parameters for MCPP: the number of edges and the number of arcs. They showed that MCPP is fixed-parameter tractable[1] (FPT) when parameterized by the number k of edges. Their algorithm is as follows. Replace every undirected edge uv by either the arc $\overrightarrow{uv}$ or arc $\overrightarrow{vu}$ or the pair $\overrightarrow{uv}$ and $\overrightarrow{vu}$ (all arc have the same weight as uv) and solve the resulting DCPP. Thus, the MCPP can be solved in time $O(3^k n^3)$, where n is the number of the number of vertices in G. We describe a faster algorithm in the full version of this paper [10].

Van Bevern *et al.* [2] and Sorge [15] left it as an open question whether the MCPP is fixed-parameter tractable when parameterized by the number of arcs.

> k-ARC CHINESE POSTMAN PROBLEM (k-ARC CPP)
> *Instance:* A strongly connected weighted mixed graph $G = (V, E \cup A)$, with vertex set V, set E of edges and set A of arcs; a weight function $w : E \cup A \to \mathbb{N}_0$.
> *Parameter:* $k = |A|$.
> *Output:* A closed walk of G that traverses each edge and arc at least once, of minimum weight.

This parameterized problem is of interest, for example, if we view the mixed graph as a network of streets in a city: while edges represent two-way streets, arcs are for one-way streets. Many cities have a relatively small number of one-way streets and so the number of arcs appears to be a good parameter for optimizing, say, police patrol in such cities [2].

We will assume for convenience that the input G of k-ARC CPP is a *simple* graph, i.e. there is at most one edge or one arc (but not both) between any pair of vertices. The multigraph version of the problem may be reduced to the simple graph version by subdividing arcs and edges. As the number of arcs and edges is at most doubled by this reduction, this does not affect the parameterized complexity of the problem.

We will show that k-ARC CPP is fixed-parameter tractable. Our proof is significantly more complicated than the one for the MCPP parameterized by the number of edges as a similar approach will not work. Instead, in FPT time, we reduce the problem to the BALANCED CHINESE POSTMAN PROBLEM (BCPP), in which there are no arcs but instead a demand function on the imbalance of the vertices. Unfortunately this problem is still NP-hard, and so we must use further techniques to solve the problem.

Marx, O'Sullivan and Razgon [11] use the following notion of a graph torso. Let $G = (V, E)$ be a graph and $S \subseteq V$. The graph $\operatorname{torso}(G, S)$ has vertex set S and vertices $a, b \in S$ are connected by an edge ab if $ab \in E$ or there is a path in G connecting a and b whose internal vertices are not in S.

Marx *et al.* [11] show that for a number of graph separation problems, it is possible to derive a graph closely related to a torso graph, which has the

[1] That is, MCPP can be solved in time $f(k)n^{O(1)}$, where f is a function only depending on k, and n is the number of vertices in G. For background and terminology on parameterized complexity we refer the reader to [6].

same separators as the original input graph. The separation problem can then be solved on this new graph, which has bounded treewidth. By contrast, we use the torso graph as a tool to construct a tree decomposition of the original graph, which does not have bounded width, but has enough other structural restrictions to make a dynamic programming algorithm possible. So, our application of Marx *et al.*'s result is quite different from its use in [11], and we believe it may be used for designing fixed-parameter algorithms for other problems on graphs. Note that Marx *et al.* are interested in small separators (i.e. sets of vertices whose removal disconnects a graph), whereas we are interested in small cuts (sets of edges whose removal disconnects a graph). This necessitates an extra step in the construction of our tree decomposition, to ensure that all minimal cuts are covered by minimal separators.

The rest of the paper is organized as follows. The next section contains further terminology and notation. In Section 3, we reduce k-arc CPP to Balanced Chinese Postman Problem (BCPP). In Section 4, we introduce and study two key notions that we use to solve BCPP: t-roads and small t-cuts. In Section 5, we investigate a special tree decomposition of the input graph of BCPP. This decomposition is used in a dynamic programming algorithm given in Section 6. The last section contains some conclusions and open problems.

2 Further Terminology and Notation

For a positive integer p and an integer q, $q < p$, $[q, p]$ will denote the set $\{q, q+1, \ldots, p\}$ and $[p]$ the set $[1, p]$. To avoid confusion, we denote an edge between two vertices u, v as uv, and an arc from u to v as $\overrightarrow{uv}$.

For a mixed multigraph G, let D be a directed multigraph derived from G by replacing each arc $\overrightarrow{uv}$ of G with multiple copies of $\overrightarrow{uv}$ (at least one), and replacing each edge in uv in G with multiple copies of the arcs $\overrightarrow{uv}$ and $\overrightarrow{vu}$ (such that there is at least one copy of $\overrightarrow{uv}$ or at least one copy of $\overrightarrow{vu}$). Then we say D is a *multi-orientation* of G. If D has exactly one copy $\overrightarrow{uv}$ for each arc $\overrightarrow{uv}$ in G, and either exactly one copy of $\overrightarrow{uv}$ or exactly one copy of $\overrightarrow{vu}$ for each edge uv in G, we say D is an *orientation* of G. If D is an orientation of G, we say that G is the *undirected version* of D (if D has parallel arcs then G has parallel edges).

For a mixed multigraph G, $\mu_G(\overrightarrow{uv})$ denotes the number of arcs of the form $\overrightarrow{uv}$ in G, and $\mu_G(uv)$ denotes the number of edges of the form uv. For a weighted graph G and a multi-orientation D of G, the *weight* of D is the sum of the weights of all its arcs, where the weight of an arc in D is the weight of the corresponding edge or arc in G.

For a directed multigraph $D = (V, A)$ and $v \in V$, $d_D^+(v)$ and $d_D^-(v)$ denote the out-degree and in-degree of v in D, respectively. Let $t : V \to \mathbb{Z}$ be a function. We say that a vertex u in D is *t-balanced* if $d_D^+(u) - d_D^-(u) = t(u)$. We say that D is *t-balanced* if every vertex is t-balanced. Note that if D is t-balanced then $\sum_{v \in V} t(v) = 0$. When $t(v) = 0$ for all $v \in V$, we omit t and speak of *balanced* vertices and *balanced* directed multigraphs. Let $V_t^+ = \{v \in V : \ t(v) > 0\}$ and $V_t^- = \{v \in V : \ t(v) < 0\}$.

In directed multigraphs, all walks (in particular, paths and cycles) that we consider are directed. A directed multigraph D is *Eulerian* if there is a closed walk of D traversing every arc exactly once. It is well-known that a directed multigraph D is Eulerian if and only if D is balanced and the undirected version of D is connected [1].

For an undirected graph $G = (V, E)$ and vertices a, b of G, a set S of edges (vertices, respectively) is called an (a, b)-*cut* ((a, b)-*separator*, respectively) if a and b are in different components of $G - S$.

Observe that the following is an equivalent formulation of the k-ARC CPP.

k-ARC CHINESE POSTMAN PROBLEM (k-ARC CPP) *Instance:* A strongly connected mixed graph $G = (V, E \cup A)$, with vertex set V, set E of edges and set A of arcs; weight function $w : E \cup A \to \mathbb{N}_0$. *Parameter:* $k = \|A\|$. *Output:* A directed multigraph D of minimum weight such that D is a multi-orientation of G and D is Eulerian.

3 Reduction to Balanced CPP

Our first step is to reduce k-ARC CPP to a problem on a graph without arcs. Essentially, given a graph $G = (V, E \cup A)$ we will "guess" the number of times each arc in A is traversed in an optimal solution. This then leaves us with a problem on $G' = (V, E)$. Rather than trying to find an Eulerian multi-orientation of G, we now try to find a multi-orientation of G' in which the imbalance between the in- and out-degrees of each vertex depends on the guesses for the arcs in A incident with that vertex.

More formally, we will provide a Turing reduction to the following problem:

BALANCED CHINESE POSTMAN PROBLEM (BCPP) *Instance:* An undirected graph $G = (V, E)$; a weight function $w : E \to \mathbb{N}_0$; a demand function $t : V \to \mathbb{Z}$ such that $\sum_{v \in V} t(v) = 0$. *Parameter:* $p = \sum_{v \in V_t^+} t(v)$. *Output:* A minimum weight t-balanced multi-orientation D of G.

Observe that when $t(v) = 0$ for all $v \in V$, BCPP is equivalent to UCPP. BCPP was studied by Zaragoza Martínez [16] who proved that the problem is NP-hard. We will reduce k-ARC CPP to BCPP by guessing the number of times each arc is traversed. In order to ensure a fixed-parameter aglorithm, we need a bound (in terms of $|A|$) on the number guesses. We will do this by bounding the total number of times any arc can be traversed in an optimal solution.

Lemma 1. *Let $G = (V, A \cup E)$ be a mixed graph, and let $k = |A|$. Then for any optimal solution D to k-ARC CPP on G with minimal number of arcs, we have that $\sum_{\overrightarrow{uv} \in A} \mu_D(\overrightarrow{uv}) \leq k^2/2 + 2k$.*

Proof. Let $A = A_1 \cup A_2$ where $A_1 = \{\overrightarrow{uv} : \overrightarrow{uv} \in A \text{ and } \mu_D(\overrightarrow{uv}) \geq 3\}$ and $A_2 = A \setminus A_1$. Let $|A_1| = p$ and $|A_2| = k - p = q$.

Consider an arc $\overrightarrow{uv} \in A$. Since D is balanced, we have that D has $\mu_D(\overrightarrow{uv})$ arc-disjoint directed cycles, each containing exactly one copy of $\overrightarrow{uv}$. We claim that each such cycle must contain at least one copy of an arc in A_2. Indeed, otherwise, there is a cycle C containing $\overrightarrow{uv}$ that does not contain any arc in A_2, which means that C consists of arcs in A_1 and arcs corresponding to (undirected) edges in G. We may construct a directed multigraph D' as follows: Remove from D two copies of each arc in A_1 that appears in C, and reverse the arcs in C that correspond to undirected edges in G. Observe that D' is Eulerian and is also a multi-orientation of G, and so D' is a solution with smaller weight than D or an optimal solution with fewer arcs than D, contradicting the minimality of D.

So each of the $\mu_D(\overrightarrow{uv})$ cycles contains at least one copy of an arc in A_2. Observe that D has at most $2q$ copies of arcs in A_2, and so $\mu_D(\overrightarrow{uv}) \leq 2q$. Thus, we have $\sum_{\overrightarrow{uv} \in A} \mu_D(\overrightarrow{uv}) = \sum_{\overrightarrow{uv} \in A_1} \mu_D(\overrightarrow{uv}) + \sum_{\overrightarrow{uv} \in A_2} \mu_D(\overrightarrow{uv}) \leq p \cdot 2q + 2q \leq 2 \cdot (\frac{p+q}{2})^2 + 2k = k^2/2 + 2k$. □

Now we may prove the following:

Lemma 2. *If* BCPP *is FPT then so is* k-ARC CPP.

Proof. Let $(G = (V, A \cup E), w)$ be an instance of k-ARC CPP, and let $k = |A|$. Let $\kappa = \lfloor k^2/2 + 2k \rfloor$. By Lemma 1, $\sum_{\overrightarrow{uv} \in A} \mu_D(\overrightarrow{uv}) \leq \kappa$ for any optimal solution D to k-ARC CPP on (G, w) with minimal number of arcs.

Let $G' = (V, E)$ and let w' be w restricted to E. Given a function $\phi : A \to [\kappa]$ such that $\sum_{\overrightarrow{uv} \in A} \phi(\overrightarrow{uv}) \leq \kappa$, let $t_\phi : V \to [-\kappa, \kappa]$ be the function such that $t_\phi(v) = \sum_{\overrightarrow{uv} \in A} \phi(\overrightarrow{uv}) - \sum_{\overrightarrow{vu} \in A} \phi(\overrightarrow{vu})$ for all $v \in V$. Observe that $\sum_{v \in V^+_{t_\phi}} t_\phi(v) \leq \sum_{\overrightarrow{uv} \in A} \phi(\overrightarrow{uv}) \leq \kappa$, and thus BCPP on (G', w', t_ϕ) has parameter $p_\phi \leq \kappa$.

Observe that given a solution D_ϕ to BCPP on (G', w', t_ϕ), if we add $\phi(\overrightarrow{uv})$ copies of each arc $\overrightarrow{uv} \in A$ to D_ϕ, then the resulting graph D is a solution to k-ARC CPP on (G, w) with weight $w'(D_\phi) + \sum_{\overrightarrow{uv} \in A} \phi(\overrightarrow{uv}) w(\overrightarrow{uv})$. Furthermore for any solution D to k-ARC CPP on (G, w), let $\phi(\overrightarrow{uv}) = \mu_D(\overrightarrow{uv})$ for each $\overrightarrow{uv} \in A$ and let D_ϕ be D restricted to E. Then D_ϕ is a solution to BCPP on (G', w', t_ϕ) and D has weight $w'(D_\phi) + \sum_{\overrightarrow{uv} \in A} \phi(\overrightarrow{uv}) w(\overrightarrow{uv})$.

Suppose that there exists an algorithm which finds the optimal solution to an instance of BCPP on (G', w', t') with parameter p in time $f(p)|V|^{O(1)}$. There are at most $\binom{q}{k}$ ways of choosing positive integers $x_1, \ldots, x_k$ such that $\sum_{i \in [k]} x_i \leq q$. Indeed, each $i \in [k]$ let $y_i = \sum_{j=1}^{i} x_j$. Then $y_i < y_j$ for $i < j$ and $y_i \in [q]$ for all i, and for any such choice of $y_1, \ldots, y_k$ there is corresponding choice of $x_1, \ldots, x_k$ satisfying $\sum_{i=1}^{k} x_i \leq q$. Therefore the number of valid choices for $x_1, \ldots, x_k$ is the number of ways of choosing $y_1, \ldots, y_k$, which is the number of ways of choosing k elements from a set of q elements.

Therefore there are at most $\binom{\kappa}{k}$ choices for a function $\phi : A \to [\kappa]$ such that $\sum_{\overrightarrow{uv} \in A} \phi(\overrightarrow{uv}) \leq \kappa$. Each choice leads to an instance of BCPP with parameter at most κ. Therefore in time $\binom{\kappa}{k} f(\kappa)|V|^{O(1)}$ we can find the optinal solution D_ϕ to BCPP on (G', w', t_ϕ) for every valid choice of ϕ.

It then remains to choose the function ϕ that minimizes $w'(D_\phi) + \sum_{\overrightarrow{uv} \in A} \phi(\overrightarrow{uv}) w(\overrightarrow{uv})$, and return the graph D_ϕ together with $\phi(\overrightarrow{uv})$ copies of each arc $\overrightarrow{uv} \in A$. □

Due to Lemma 2, we may now focus on BCPP.

4 *t*-roads and *t*-cuts

Lemma 3. *Let (G, w, t) be an instance of* BCPP*, with $p = \sum_{v \in V_t^+} t(v)$. Then for any optimal solution D to* BCPP *on (G, w, t) with minimal number of arcs, we have that $\mu_D(\overrightarrow{uv}) + \mu_D(\overrightarrow{vu}) \leq \max\{p, 2\}$ for each edge uv in G.*

Proof. Suppose that $\mu_D(\overrightarrow{uv}) + \mu_D(\overrightarrow{vu}) > \max\{p, 2\}$ for some edge uv in G. Observe that if $\mu_D(\overrightarrow{uv}) \geq 1$ and $\mu_D(\overrightarrow{vu}) \geq 1$, then by removing one copy of $\overrightarrow{uv}$ and one copy of $\overrightarrow{vu}$, we obtain a solution to BCPP on (G, w, t) with weight at most that of D but with fewer arcs. Therefore, we may assume that $\mu_D(\overrightarrow{uv}) > \max\{p, 2\}$ and $\mu_D(\overrightarrow{vu}) = 0$.

We now show that there must exist a cycle in D containing a copy of $\overrightarrow{uv}$.

Modify D by adding a new vertex x, with $t(v)$ arcs from x to v for each $v \in V_t^+$, and $-t(v)$ arcs from v to x for each $v \in V_t^-$. Let D^* be the resulting directed graph. Then observe that D^* is balanced, and therefore D^* has $\mu_D(\overrightarrow{uv})$ arc-disjoint cycles, each containing exactly one copy of $\overrightarrow{uv}$. At most p of these cycles can pass through x. Therefore there is at least one cycle containing $\overrightarrow{uv}$ which is a cycle in D.

So now let $v = v_1, v_2, \ldots, v_l = u$ be a sequence of vertices such that $\mu_D(\overrightarrow{v_i v_{i+1}}) \geq 1$ for each $i \in [l-1]$. Replace one copy of each arc $\overrightarrow{v_i v_{i+1}}$ with a copy of $\overrightarrow{v_{i+1} v_i}$ and remove 2 copies of $\overrightarrow{uv}$. Observe that the resulting graph covers every edge of G, and the imbalance of each vertex is the same as in D. Therefore, we have a solution to BCPP on (G, w, t) with weight at most that of D but with fewer arcs. This contradiction proves the lemma. □

Definition 1. *Let $H = (V, E)$ be an undirected multigraph and t a function $V \to \mathbb{Z}$. A* t-road *is a directed multigraph T such that for each vertex v, $d_T^+(v) - d_T^-(v) = t(v)$. We say H* has a t-road T *if there is a subgraph H' of H such that T is an orientation of H'.*

Observe that given a solution D to the BCPP on (G, w, t), the undirected version of D has a t-road.

Definition 2. *Let $H = (V, E(H))$ be an undirected multigraph and $t : V \to \mathbb{Z}$ a function such that $\sum_{v \in V} t(v) = 0$. Let H^* be the multigraph derived from H by creating two new vertices a, b, with $t(v)$ edges between a and v for each $v \in V_t^+$, and $-t(v)$ edges between b and v for each $v \in V_t^-$. Let $p = \sum_{v \in V_t^+} t(v)$. Then a* small t-cut *is a set of edges $F \subseteq E(H)$ such that $F = E(H) \cap F'$ for some minimal (a, b)-cut F' of H^* and $|F'| < p$.*

Note that a small t-cut can be the empty set. A t-road, if one exists, can be found in polynomial time by computing a flow of value p from a to b in the unit capacity network N with underlying multigraph H^*. The next lemma follows from the well-known max-flow-min-cut theorem for N.

Lemma 4. *An undirected multigraph H has a t-road if and only if H does not have a small t-cut.*

Let $(G = (V, E), w, t)$ be an instance of BCPP, and let F be the union of all small t-cuts in G. We say a t-road T is *well-behaved* if $\mu_T(\overrightarrow{uv}) + \mu_T(\overrightarrow{vu}) \leq 1$ for all $uv \in E \setminus F$.

Lemma 5. *Let D be an optimal solution to* BCPP *on $(G = (V, E), w, t)$, and let H be the undirected version of D. Then H has a well-behaved t-road.*

Proof. Let $F \subseteq E$ be the union of all small t-cuts in G. Let J be the undirected multigraph derived from H by removing all but one copy of every edge in $E \setminus F$. Observe that every t-road in J is also a t-road in H and every t-road in J is well-behaved. So, it is sufficient to show that J has a t-road.

Note that if J does not have a t-road, then by Lemma 4 J has a small t-cut. Note also that by construction H has a t-road and therefore does not have a small t-cut. Consider a small t-cut S in J and suppose that every edge in S is a copy of an edge in F. As S is not a small t-cut in H, there are vertices $u \in V_t^+$ and $v \in V_t^-$ such that $H \setminus S$ contains a path $v_1 v_2 \dots v_l$, where $v_1 = u$ and $v_l = v$. Note that $v_1 \dots v_l$ is also a path in $J \setminus S$, unless all copies of the edge $v_i v_{i+1}$ are in S for some $i \in [l-1]$. However, as $S \subseteq F$, if all copies of $v_i v_{i+1}$ in J are in S, then all copies of $v_i v_{i+1}$ in H are in S (as $\mu_H(v_i v_{i+1}) = \mu_J(v_i v_{i+1})$), and $v_1 \dots v_l$ is not a path in $H \setminus S$, a contradiction. Therefore $v_1 \dots v_l$ is a path in $J \setminus S$, and so S is not a small t-cut in J, a contradiction. Therefore every small t-cut in J contains a copy of an edge not in F. If J has a small t-cut, then as every small t-cut in J is also a small t-cut in G, it follows that there is a small t-cut in G containing edges not in F. This is a contradiction by definition of F. Therefore we may conclude that J does not have a small t-cut, and so J has a t-road, as required. □

If $|F|$ is bounded by a function on p then, using Lemma 5 we can solve BCPP in FPT time by guessing the multiplicities of each edge in F for an optimal solution D. Unfortunately, $|F|$ may be larger than any function of p in general. It is also possible to solve the problem on graphs of bounded treewidth using dynamic programming techniques, but in general the treewidth may be unbounded. In Section 5 we give a tree decomposition of G in which the number of edges from F in each bag is bounded by a function of p. This allows us to combine both techniques. In Section 6 we give a dynamic programming algorithm utilizing Lemma 5 that runs in FPT time.

5 Tree Decomposition

In this section, we provide a tree decomposition of G which we will use for our dynamic programming algorithm. The tree decomposition does not have

bounded treewidth (i.e. the bags do not have bounded size), but the intersection between bags is small, and each bag has a bounded number of vertices from small t-cuts. This will turn out to be enough to develop a fixed-parameter algorithm, as in some sense the hardness of BCPP comes from the small t-cuts.

Our tree decomposition is based on a result by Marx, O'Sullivan and Razgon [11], in which they show that the minimal small separators of a graph "live in a part of the graph that has bounded treewidth"[11].

Definition 3. *Given an undirected graph $G = (V, E)$, a* tree decomposition *of G is a pair $(\mathcal{T}, \beta)$, where $\mathcal{T}$ is a tree and $\beta : V(\mathcal{T}) \to 2^V$ such that $\bigcup_{x \in V(\mathcal{T})} \beta(x) = V$, for each edge $uv \in E$, there exists a node $x \in V(\mathcal{T})$ such that $u, v \in \beta(x)$, and for each $v \in V$, the set $\beta^{-1}(v)$ of nodes form a connected subgraph in $\mathcal{T}$.*

The width *of $(\mathcal{T}, \beta)$ is $\max_{x \in V(\mathcal{T})}(|\beta(x)| - 1)$. The* treewidth *of G (denoted $tw(G)$) is the minimum width of all tree decompositions of G.*

Lemma 6. *[11, Lemma 2.11] Let a, b be vertices of a graph $G = (V, E)$ and let l be the minimum size of an (a, b)-separator. For some $e \geq 0$, let S be the union of all minimal (a, b)-separators of size at most $l + e$. Then there is an $f(l, e) \cdot (|E| + |V|)$ time algorithm that returns a set $S' \supseteq S$ disjoint from $\{a, b\}$ such that $tw(torso(G, S')) \leq g(l, e)$, for some functions f and g depending only on l and e.*

Definition 4. *Given an undirected graph $G = (V, E)$, a* nice tree decomposition *$(\mathcal{T}, \beta)$ is a tree decomposition such that $\mathcal{T}$ is a rooted tree, and each of the nodes $x \in V(\mathcal{T})$ falls under one of the following classes:*

- *x* **is a Leaf node:** *Then x has no children in $\mathcal{T}$;*
- *x* **is an Introduce node:** *Then x has a single child y in $\mathcal{T}$, and there exists a vertex $v \notin \beta(y)$ such that $\beta(x) = \beta(y) \cup \{v\}$;*
- *x* **is a Forget node:** *Then x has a single child y in $\mathcal{T}$, and there exists a vertex $v \in \beta(y)$ such that $\beta(x) = \beta(y) \setminus \{v\}$;*
- *x* **is a Join node:** *Then x has two children y and z, and $\beta(x) = \beta(y) = \beta(z)$.*

It is well-known that given a tree decomposition of a graph, it can be transformed into a nice tree decomposition of the same width in polynomial time.

Our tree decomposition will be similar but not identical to a nice tree decomposition. We are now ready to give our tree decomposition, which is the main result of this section. Lemma 7 is proved in the Appendix.

Lemma 7. *Let $(G = (V, E), w, t)$ be an instance of* BCPP*, let C be the non-empty set of vertices appearing in edges in small t-cuts. Then there is an $f(p) \cdot (n+m)^{O(1)}$ time algorithm that returns a set S' and a (binary) tree decomposition $(\mathcal{T}, \beta)$ of G such that:*

1. *$C \subseteq S'$;*
2. *For any nodes $x \neq y$ in $\mathcal{T}$, $\beta(x) \cap \beta(y) \subseteq S'$;*
3. *For any node x in $\mathcal{T}$, $|\beta(x) \cap S'| \leq g(p)$*

for some functions f and g depending only on p. In addition, $\mathcal{T}$ is a rooted tree, and each of the nodes $\mathcal{T}$ falls under one of the following classes:

1. *x* **is a Leaf node:** *Then x has no children;*
2. *x* **is an Introduce node:** *Then x has a single child y in $\mathcal{T}$, $\beta(x) \subseteq S'$, and there exists a vertex $v \notin \beta(y)$ such that $\beta(x) = (\beta(y) \cap S') \cup \{v\}$;*
3. *x* **is a Forget node:** *Then x has a single child y in $\mathcal{T}$, $\beta(x) \subseteq S'$, and there exists a vertex $v \in \beta(y) \cap S'$ such that $\beta(x) = (\beta(y) \cap S') \setminus \{v\}$;*
4. *x* **is a Join node:** *Then x has two children y and z, $\beta(x) \subseteq S'$, and $\beta(x) = \beta(y) \cap S' = \beta(z) \cap S'$.*

Note that in our tree decomposition, the only nodes of $\mathcal{T}$ with bags not contained in S' are the Leaf nodes.

6 Dynamic Programming

Let (G, w, t) be an instance of BCPP. Let $(\mathcal{T}, \beta)$ be the tree decomposition of G and S' the set of vertices containing all vertices of every small t-cut given by Lemma 7. In this section we give a dynamic programming algorithm based on this decomposition.

Unlike the usual use of dynamic programming tree decompositions, we will not construct the restrictions of potential solutions to each bag when that bag is processed. Instead, we will produce undirected multigraphs corresponding to the undirected versions of potential solutions. When we have the optimal undirected multigraph for the whole problem, we then find its orientation as a last step. We do this because trying to find an optimal orientation at each bag $\beta(x)$ would involve making the bag t'-balanced for an arbitrary function $t' : \beta(x) \to [-p, p]$. On the other hand, to find the undirected version of a solution it is enough to guess the parity of the degree of each vertex within each bag.

To this end we introduce a function $h : V(G) \to \{\text{ODD}, \text{EVEN}\}$, where $h(v) = \text{ODD}$ if $t(v)$ is odd and $h(v) = \text{EVEN}$ if $t(v)$ is even. Observe that in the undirected version of any solution to BCPP on (G, w, t), each vertex v will have odd degree if and only if $t(v)$ is odd. Thus, h and similar functions will be used to tell us whether a vertex should have odd or even degree.

To simplify some expressions, we adopt the convention that $\text{ODD} + \text{ODD} = \text{EVEN}, \text{EVEN} + \text{EVEN} = \text{EVEN}$, and $\text{ODD} + \text{EVEN} = \text{ODD}$. We say a vertex v is *h-balanced* if it has odd degree if and only if $h(v) = \text{ODD}$. An undirected multigraph H is *h-balanced* if every vertex is h-balanced.

Let $\alpha(x) = \beta(x) \cap S'$. Thus $\beta(x) \cap \beta(y) \subseteq \alpha(x)$ for all nodes $x \neq y$, and $\alpha(x) = \beta(x)$ for every non-leaf x. Furthermore, for any Join node x with two children y and z, we have that $\alpha(x) = \alpha(y) = \alpha(z)$, even if one or both of the children of x is a Leaf node whose bag contains vertices not in S'.

Let $\gamma(x)$ be the union of the bags of all predecessors of x including x itself. Thus, if r is the root node of $\mathcal{T}$, then $\gamma(r) = V(G)$.

We now define the set of graphs constructed in our dynamic programming algorithm. Let x be a node of $\mathcal{T}$, let H' be an undirected multigraph with

underlying graph $G[\alpha(x)]$, such that $\mu_{H'}(uv) \leq \max\{p, 2\}$ for all edges uv. Let T' be a directed graph with vertex set $\alpha(x)$, such that $\mu_{T'}(\overrightarrow{uv}) + \mu_{T'}(\overrightarrow{vu}) \leq \mu_{H'}(uv)$ for all edges uv. Let t' be a function $\alpha(x) \to [-p, p]$ and let h' be a function $\alpha(x) \to \{\text{ODD}, \text{EVEN}\}$. Then let $\psi(x, H', T', t', h')$ be an undirected multigraph H with underlying graph $G[\gamma(x)]$, of minimum weight such that

1. $H[\alpha(x)] = H'$.
2. H has a well-behaved t^*-road T such that T restricted to $\alpha(x)$ is T', where $t^* : \gamma(x) \to [-p, p]$ is the function such that $t^*(v) = t'(v)$ for $v \in \alpha(x)$ and $t^*(v) = t(v)$, otherwise.
3. H is h^*-balanced, where $h^* : \gamma(x) \to \{\text{ODD}, \text{EVEN}\}$ is the function such that $h^*(v) = h'(v)$ if $v \in \alpha(x)$ and $h^*(v) = h(v)$, otherwise.

Lemma 8. *Let r be the root node of $\mathcal{T}$. Let t' be t restricted to $\alpha(r)$, and let h' be h restricted to $\alpha(r)$. Let H' and T' be chosen such that the weight of $H = \psi(r, H', T', t', h')$ is minimized. Then the weight of H is the weight of an optimal solution to* BCPP *on (G, w, t), and given H we may construct an optimal solution to* BCPP *on (G, w, t) in polynomial time.*

Proof. Observe that by construction of t' and h', t^* and h^* in the definition of $\psi(r, H', T', t', h')$ are t and h, respectively. Let $H = \psi(r, H', T', t', h')$ for some choice of H' and T'. Then by definition H has a well-behaved t-road T and H is h-balanced. For each arc $\overrightarrow{uv}$ in T, orient a copy of the edge uv in H from u to v. Let D' be the resulting mixed multigraph. Then for every vertex v in D', we have $d^+_{D'}(v) - d^-_{D'}(v) = t(v)$. By definition of h and the fact that H was h-balanced, every v has an even number of edges incident to it.

Thus, the undirected edges can be partitioned into a set of cycles. By orienting each of these cycles to make a directed cycle, we get a directed multigraph D which is a solution to BCPP on (G, w, t). This shows that for every choice of H' and T', the graph $\psi(r, H', T', t', h')$ can be oriented to produce a solution to BCPP on (G, w, t). We will now show that an optimal solution D to BCPP on (G, w, t) is an orientation of $H = \psi(r, H', T', t', h')$ for some choice of H', T'.

Let H' be the undirected version of D restricted to $\alpha(r)$. Given a t-road T in D, let T' be T restricted to $\alpha(r)$. By Lemma 5, we may assume that T is well-behaved. Observe that H satisfies the conditions of $\psi(r, H', T', t', h')$. □

Given an undirected graph F and a set X of vertices of F of even size, a set J of edges of F is an *X-Join* if $d_{F[J]}(v)$ is odd if and only if $v \in X$. When F has weights on its edges, we can speak of the MINIMUM WEIGHT X-JOIN PROBLEM; this problem can be solved in time $O(|V(F)|^3)$ [7]. (Traditionally, the MINIMUM WEIGHT X-JOIN PROBLEM is called the MINIMUM WEIGHT T-JOIN PROBLEM, but we use T for t-roads.)

Lemma 9. *$\psi(x, H', T', t', h')$ can be calculated in FPT time, for all choices of x, H', T', t', h'.*

Proof. Consider some node x, and assume that we have already calculated $\psi(y, H'', T'', t'', h'')$, for all descendants y of x and all choices of H'', T'', t'', h''. We consider the possible types of nodes separately.

x **is a Leaf node:** If $\beta(x) \subseteq S'$, then the only possible graph is H'. So return H' if H' is a solution, and return NULL, otherwise.

If $\beta(x) \setminus S' \neq \emptyset$, proceed as follows. Let $G_x = (\beta(x), E(G[\beta(x)]) \setminus E(G[\alpha(x)]))$. For each $v \in \beta(x)$, let $t''(v) = t'(v) - \sum_u \mu_{T'}(\overrightarrow{vu}) + \sum_u \mu_{T'}(\overrightarrow{uv})$. Then for any t'-road T^* that agrees with T' on $\alpha(x)$, T^* is the union of T' and a t''-road T'' on G_x. Furthermore, if T^* is well-behaved then $\mu_{T''}(\overrightarrow{uv}) + \mu_{T''}(\overrightarrow{vu}) \leq 1$ for any u, v. Thus, if $\psi(x, H', T', t', h') \neq$ NULL, then G_x has a t''-road. So we may proceed as follows. Check if G_x has a t''-road. If it does not, then return NULL. Otherwise, let $h^{**} : \beta(x) \to \{\text{ODD}, \text{EVEN}\}$ be such that if $v \in \alpha(x)$ has odd degree in H', then $h^{**}(v) = h^*(v) + \text{ODD}$, and otherwise $h^{**}(v) = h^*(v)$. Observe that the restriction of $\psi(x, H', T', t', h')$ to G_x will be h^{**}-balanced. Then to find $\psi(x, H', T', t', h')$, it suffices to find a minimum weight (multi)set of edges to add to G_x to make it h^{**}-balanced. This can be done by solving the MINIMUM WEIGHT X-JOIN PROBLEM, where X is the set of all vertices in $\beta(x)$ that are not h^{**}-balanced in G_x.

x **is an Introduce node:** Let y be the child node of x, and let v be the single vertex in $\beta(x) \setminus \alpha(y)$. Then no vertices in $\gamma(x)$ are adjacent with v, except for those in $\alpha(x)$. Therefore if v is not h'-balanced in H' or is not t'-balanced in T', we may return NULL. Otherwise, let H'' be H' restricted to $\alpha(y)$. Let T'' be T' restricted to $\alpha(y)$. Let $t'' : \alpha(y) \to [-p, p]$ be such that $t''(u) = t'(u) - \mu_{T'}(\overrightarrow{uv}) + \mu_{T'}(\overrightarrow{vu})$. Let $h'' : \alpha(y) \to \{\text{ODD}, \text{EVEN}\}$ be such that if $\mu_{H'}(uv)$ is odd then $h''(u) = h'(u) + \text{ODD}$, and otherwise $h''(u) = h'(u)$. Then $\psi(x, H', T', t', h')$ is $\psi(y, H'', T'', t'', h'')$ together with the edges of H' incident with v.

x **is a Forget node:** Let y be the child node of x, and let v be the single vertex in $\alpha(y) \setminus \beta(x)$. Let $t'' : \alpha(y) \to [-p, p]$ be the function that extends t' and assigns v to $t(v)$. Let $h'' : \alpha(y) \to \{\text{ODD}, \text{EVEN}\}$ be the function that extends h' and assings v to $h(v)$. Then $\psi(x, H', T', t', h')$ is $\psi(y, H'', T'', t'', h'')$, for some choice of H'' and T'' minimizing the weight of $\psi(y, H'', T'', t'', h'')$ such that H'' restricted to $\alpha(x)$ is H', and T'' restricted to $\alpha(x)$ is T'.

The proof of the case when x **is a Join node** and the analysis of algorithm's running time are in the Appendix. □

Lemmas 8 and 9 imply the following:

Theorem 1. BCPP *is fixed-parameter tractable.*

Theorem 1 and Lemma 2 imply the following:

Theorem 2. k-ARC CPP *is fixed-parameter tractable.*

7 Conclusions and Open Problems

We have solved an open problem in [2] by showing that MCPP parameterized by the number of arcs is fixed-parameter tractable. To prove this result we reduced MCPP to a generalization of UCPP and applied a very useful lemma of Marx *et al.* [11] on treewidth of the torso graph with respect to small separators. Note

that our application of the lemma is significantly different from those in [11] and we believe that our approach will be of interest in designing fixed-parameter algorithms for other problems.

Van Bevern *et al.* [2] mention two other parameterizations of MCPP. One of them is by $\mathrm{tw}(G)$. It was proved by Fernandes *et al.* [9] that this parameterisation of MCPP is in XP, but it is unknown whether it is FPT [2]. A vertex v of G is called even if the number of arcs and edges incident to v is even. Edmonds and Johnson [7] proved that if all vertices of G are even then MCPP is polynomial time solvable. So, the number of odd (not even) vertices is a natural parameter. It is unknown whether the corresponding parameterization of MCPP is FPT [2].

Acknowledgement Research of GG was supported by Royal Society Wolfson Research Merit Award.

References

1. J. Bang-Jensen and G. Gutin, Digraphs: Theory, Algorithms and Applications, 2nd Ed., Springer, 2009.
2. R. van Bevern, R. Niedermeier, M. Sorge, and M. Weller, Complexity of Arc Rooting Problems. Chapter 2 in A. Corberán and G. Laporte (eds.), Arc Routing: Problems, Methods and Applications, SIAM, Phil., in press.
3. E. J. Beltrami and L. D. Bodin. Networks and vehicle routing for municipal waste collection. Networks, 4(1):65–94, 1974.
4. P. Brucker. The Chinese postman problem for mixed graphs. Lect. Notes Comput. Sci. 100 (1981) 354–366.
5. N. Christofides. The optimum traversal of a graph. Omega 1(1973) 719–732.
6. R. G. Downey and M. R. Fellows, *Fundamentals of Parameterized Complexity*, Springer, 2013.
7. J. Edmonds and E. L. Johnson. Matching, Euler tours and the Chinese postman. Mathematical Programming, 5 (1973) 88–124.
8. H. A. Eiselt, M. Gendreau, and G. Laporte. Arc routing problems. I. The Chinese postman problem. Oper. Res. 43 (1995) 231–242.
9. C. G. Fernandes, O. Lee, and Y. Wakabayashi. Minimum cycle cover and Chinese postman problems on mixed graphs with bounded tree-width. Discrete Applied Mathematics, 157(2):272–279, 2009.
10. G. Gutin, M. Jones and B. Sheng, Parameterized Complexity of the k-Arc Chinese Postman Problem. `http://arxiv.org/abs/1403.1512`
11. D. Marx, B. O'Sullivan and I. Razgon, Finding small separators in linear time via treewidth reduction. ACM Trans. Algorithms 9 (2013) article 30.
12. E. Minieka. The Chinese postman problem for mixed networks. Management Sci. 25 (1979/80) 643–648.
13. C. H. Papadimitriou. On the complexity of edge traversing. J. ACM 23 (1976) 544–554.
14. Y. Peng. Approximation algorithms for some postman problems over mixed graphs. Chinese J. Oper. Res. 8 (1989) 76–80.
15. M. Sorge, Some Algorithmic Challenges in Arc Routing. Talk at NII Shonan Seminar no. 18, May 2013.
16. F.J. Zaragoza Martínez, Postman Problems on Mixed Graphs. PhD thesis, University of Waterloo, 2003.