View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Royal Holloway - Pure

The Mixed Chinese Postman Problem Parameterized by Pathwidth and Treedepth*

Gregory Gutin, Mark Jones, and Magnus Wahlström
Royal Holloway, University of London
Egham, Surrey TW20 0EX, UK

Abstract

In the Mixed Chinese Postman Problem (MCPP), given a weighted mixed graph G (it may have both edges and arcs), our aim is to find a closed walk of minimum weight traversing each edge and arc at least once. The MCPP parameterized by the number of edges in G or the number of arcs in G is fixed-parameter tractable as proved by van Bevern *et al.* (2014) and Gutin, Jones and Sheng (2014), respectively. Solving an open question of van Bevern *et al.* (2014), we show that somewhat unexpectedly MCPP parameterized by the (undirected) treewidth of G is W[1]-hard. In fact, we prove that even the unweighted MCPP parameterized by the pathwidth of G is W[1]-hard. On the positive side, we show that MCPP parameterized by treedepth is fixed-parameter tractable (even with arbitrary integer weights). We are unaware of any widely studied graph parameters between pathwidth and treedepth and so our results provide a close characterization of the complexity of MCPP.

1 Introduction

Let us formally introduce the Mixed Chinese Postman Problem. A *mixed graph* $G = (V, E \cup A)$ may contain both edges (set E) and arcs (set A).[1] A mixed graph G is *strongly connected* if for each ordered pair x, y of vertices in G there is a walk from x to y that traverses each arc in its direction. In this paper, we will deal with *simple* mixed graphs (where for each pair of vertices u, v, at most one of the edge uv, the arc uv and the arc vu exist)[2] and (possibly non-simple) directed multigraphs (with multiple arcs between each pair of vertices). If every edge and arc e of a mixed graph G is assigned a weight $w(e)$, the *weight* of a walk Q in G is $\sum_{e \in E \cup A} t_Q(e)w(e)$, where $t_Q(e)$ is the number of occurrences of e in Q. Whenever we refer to the treewidth (pathwidth, treedepth) of a graph, we mean the treewidth (pathwidth, treedepth) of the underlying undirected graph.

> 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 H of G that traverses each edge and arc at least once, of minimum weight.

*An extended abstract [17] of this paper has appeared in ESA 2015.

[1] Undefined terms in graph theory and parameterized complexity used in this section, are defined in the next section.

[2] This assumption is for convenience only. We can relax it in our algorithm by subdividing parallel arcs and edges, as this increases treedepth by at most one.

In what follows, we will consider a solution H of MCPP as both a walk in G and a strongly connected directed multigraph G^*. Every arc of G^* corresponds to either an arc of G or an edge of G together with one of the two directions to traverse the edge. The multiplicity of an arc xy of G^* equals the number times arc xy of G is traversed by the walk or the number of times edge xy of G is traversed from x to y by the walk. Clearly, G^* is an Euler directed multigraph in which each vertex is *balanced*, i.e., the out-degree and in-degree of each vertex are equal. Every Euler trail of G^* corresponds to a closed walk of G that traverses each arc and edge of G at least once. We say that G^* *covers* an arc (edge) in G if the multiplicity of that arc in G^* (the combined multiplicity of both orientations of that edge in G^*) is at least 1. Thus, G^* must cover every edge and arc in G.

There is numerous literature on various algorithms and heuristics for MCPP; for informative surveys, see [2,5,11,20]. 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 [10] and so is DCPP [1,6,10], 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 [19]. It is therefore reasonable to believe that MCPP may become easier the closer it gets to UCPP or DCPP. Indeed, when parameterized by the number of edges in G or the number of arcs in G, MCPP is proved to be fixed-parameter tractable (FPT) by van Bevern *et al.* [2] and Gutin, Jones and Sheng [16], respectively.

Van Bevern *et al.* [2] noted that Fernandes, Lee and Wakabayashi [13] proved that MCPP parameterized by the treewidth of G is in XP (when all edges and arcs have weight 1), and asked whether this parameterization of MCPP is FPT. It is well-known that many graph problems are FPT when parameterized by the treewidth of the input graph (only a few such problems are W[1]-hard; see, e.g., [8,12,15]). In this paper, we show that somewhat unexpectedly the MCPP parameterized by treewidth belongs to a small minority of problems, i.e., it is W[1]-hard. In fact, we prove a stronger result by (i) replacing treewidth with pathwidth, and (ii) assuming that all edges and arcs have weight 1.

To complement this, we obtain a positive result for the parameter treedepth. We design an algorithm which uses an "iterative improvement" strategy, i.e., given a solution to MCPP for a graph of treedepth k, it finds a solution of lower cost in FPT time (or conclude that we have found the optimum). The algorithm searches for a feasible improvement step by applying dynamic programming on the treedepth decomposition of the graph, and after iterating the process a sufficient number of times, an optimal solution will have been produced. We show that if there exists such an improvement, then there also exists an improvement step where the edit distance between the two solutions is bounded by a function of the treedepth. Note that the bound on treedepth is used here in two different manners: to prove a structural result about the space of the solutions, and to run the final dynamic programming algorithm.

To make our presentation easier to understand and following [13], we initially assume that all weights equal 1. However, we also show how to generalize our result to the integer-weighted case.

Our paper is organized as follows. Some basic terminology and notation in graph theory and parameterized complexity is defined in the next section. In Section 3 we introduce an intermediate problem Properly Balanced Subgraph (PBS), and give a W[1]-hardness proof for a restricted variant of it. In Section 4 we reduce this variant of PBS to MCPP parameterized by pathwidth, showing that the latter is also W[1]-hard. In Section 5 we show that PBS is FPT with respect to treedepth, as outlined above, and in Section 6 we reduce MCPP parameterized by treedepth to PBS parameterized by treedepth, showing that this parameterization of MCPP is FPT. In Section 7 we discuss the generalization to arbitrary integer weights. We conclude the paper with Section 8, where, in particular, we mention an open question from van Bevern *et al.* [2] on another parameterization of MCPP.

2 Preliminaries

Directed and Mixed Graphs. In this paper, all walks in mixed and directed graphs are *directed*, i.e., they traverse arcs in their directions. A walk is *closed* if it starts and ends at the same vertex. For an arc uv, u is its *tail* and v is its *head*. For a vertex x in a directed multigraph D, the *out-degree* $d^+_D(x)$ of x (*in-degree* $d^-_D(x)$ of x) is the number of arcs whose tails (heads) are x. We will usually drop the subscript when D can be understood from context. A *trail* in a directed multigraph D is a walk without repeated arcs. A closed trail in D is *Euler* if it traverses every arc just once. It is well-known that D has an Euler trail if and only if D is strongly connected and every vertex x of D is *balanced*, i.e., $d^+(x) = d^-(x)$.

Treewidth, Pathwidth and Treedepth. For 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 $x \in V(\mathcal{T})$ with $u, v \in \beta(x)$, and
- for each $v \in V$ the set of nodes $\beta^{-1}(v)$ induces 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 is the minimum width of all tree decompositions of G.

The *pathwidth* of a graph is the minimum width of all tree decompositions $(\mathcal{T}, \beta)$ for which $\mathcal{T}$ is a path. For a directed multigraph H, we will use $\mathrm{pw}(H)$ to denote the pathwidth of the underlying undirected graph of H.

The *treedepth* of a connected graph G is defined as follows. Let T be a rooted tree with vertex set $V(G)$, such that if xy is an edge in G then x is either an ancestor or a descendant of y in T. Then we say that G *is embedded in* T. The *depth* of T is the number of vertices in a longest path in T from the root to a leaf. The *treedepth* of G is the minimum t such that G is embedded in a tree of depth t. Thus, for example, a star $K_{1,r}$ has treedepth 2. It is well-known that a path of length n has treedepth $O(\log n)$ and a graph of treedepth k has pathwidth at most $k - 1$ [3].

Parameterized Complexity. A *parameterized problem* is a subset $L \subseteq \Sigma^* \times \mathbb{N}$ over a finite alphabet Σ. L is *fixed-parameter tractable (FPT)* if the membership of an instance (x, k) in L can be decided in time $f(k)|x|^{O(1)}$ (called *FPT time* and the corresponding algorithm an *FPT algorithm*), and in XP if it can be decided in time $|x|^{f(k)}$, where f is a computable function of the *parameter* k only. Let P and Q be parameterized problems, a function $\phi : P \to Q$ is a *parameterized reduction from* P *to* Q if

- $x \in P$ if and only if $\phi(x) \in Q$,
- for $x \in P$, $\phi(x)$ can be computed in FPT time (in the parameter k of x), and
- there is a function $g(k)$ such that if k is the parameter of $x \in P$ and k' is the parameter of $\phi(x)$, then $k' \leq g(k)$.

It is not hard to see that if there is a parameterized reduction from P to Q and Q is FPT, then so is P.

There exists a collection of parameterized complexity classes, $\mathrm{W}[1], \mathrm{W}[2], \ldots$ such that $\mathrm{FPT} \subseteq \mathrm{W}[1] \subseteq \mathrm{W}[2] \subseteq \cdots \subseteq \mathrm{XP}$. Informally, a parameterized problem belongs to the complexity class $\mathrm{W}[i]$ if there exists an FPT algorithm that transforms every instance of the problem into an

instance of Weighted Circuit Satisfiability for a circuit of weft i: FPT is the class W[0]. It is widely-believed and generally assumed that FPT $\neq$ W[1]. Consider κ-Clique, Clique parameterized by the number κ of vertices of the required clique. κ-Clique was proved to be W[1]-complete and so it is widely-believed that the problem is not FPT. For more information on parameterized algorithms and complexity, see [7, 9].

Other Notation. Henceforth, for each positive integer n, $[n] = \{1, 2, \ldots, n\}$.

3 Properly Balanced Subgraph Problem

In this section, we introduce the problem Properly Balanced Subgraph (PBS). As discussed at the end of Section 1, we use PBS as an intermediate problem towards deriving results for MCPP. The intuition behind the problem is that PBS captures an "improvement step" of MCPP, i.e., given a (possibly suboptimal) solution to MCPP, the task of finding a solution of lower cost can be cast as an instance of PBS. In this section we show that PBS is W[1]-hard parameterized by pathwidth, and in Section 5 we show that a special case of PBS with restricted weights is fixed-parameter tractable with respect to treedepth. Both of these results are extended to MCPP via parameterized reductions, in Sections 4 and 6.

A directed multigraph is called *balanced* if the in-degree of each vertex coincides with its out-degree. A *double arc* is a pair of arcs (a, a') such that a and a' have the same heads and tails. We will say that a subgraph D' of D *respects double arcs* if $|A(D') \cap \{a, a'\}| \neq 1$ for every double arc (a, a'). A *forbidden pair* is a specified pair of arcs (b, b') such that b is the reverse of b'. We say that D' *respects forbidden pairs* if $|A(D') \cap \{b, b'\}| \neq 2$ for every forbidden pair (b, b'). We will say that a subgraph D' of D is *properly balanced* if D' is balanced and respects double arcs and forbidden pairs. PBS is then defined as follows.

> Properly Balanced Subgraph (PBS)
> *Instance:* A directed multigraph $D = (V, A)$; a weight function $w : A \to \mathbb{Z}$; a set $X = \{(a_1, a'_1), \ldots, (a_r, a'_r)\}$ of *double arcs* with $a_i, a'_i \in A$ for each $i \in [r]$; a set $Y = \{(b_1, b'_1), \ldots, (b_s, b'_s)\}$ of *forbidden pairs* with $b_i, b'_i \in A$ for each $i \in [s]$. Each arc occurs in at most one pair of $X \cup Y$.
> *Output:* A properly balanced subgraph D' of D of negative weight, if one exists.

Throughout the paper, when we talk about a graph in the context of PBS, we implicitly mean a directed multigraph together with a weight function, a set of double arcs and a set of forbidden pairs, as described above.

3.1 Gadgets for PBS

We now describe three basic gadget graphs used in our proof that PBS is W[1]-hard parameterized by pathwidth (for now we do not assign weights; we will do this later). Each gadget will have some number of *input* and *output* arcs. Later, we will combine these gadgets by joining the input and output arcs of different gadgets together using double arcs.

Definition 1. *A* Duplication gadget *has one input arc and t output arcs, for some positive integer t. The vertex set consists of vertices x, y, and u_i, v_i for each $i \in [t]$. The arcs form a cycle*

$$xy, yu_1, u_1v_1, v_1u_2, u_2v_2, \ldots, u_tv_t, v_tx.$$

The input arc is the arc xy, and the output arcs are the arcs u_iv_i for each $i \in [t]$.

Definition 2. *A* Choice gadget *(see Figure 1) has one input arc xy and t output arcs $u_iv_i : i \in [t]$, for some positive integer t. The vertex set consists of the vertices x, y, z, w and u_i, v_i for each $i \in [t]$. The arcs consist of a path $wxyz$, and the path zu_iv_iw for each $i \in [t]$.*

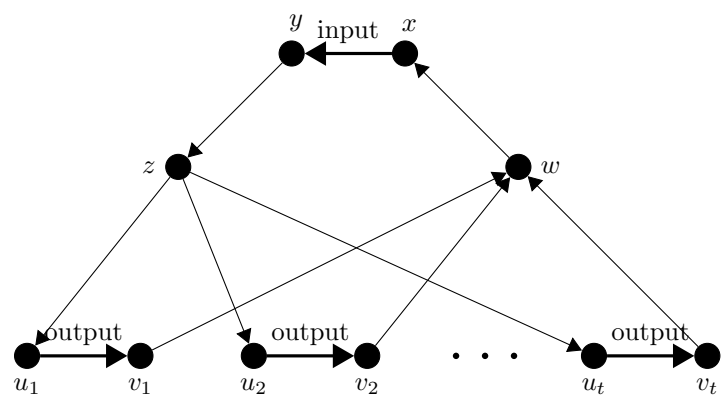


Figure 1: A Choice gadget with t output arcs. A balanced subgraph that contains the input arc will contain exactly one output arc.

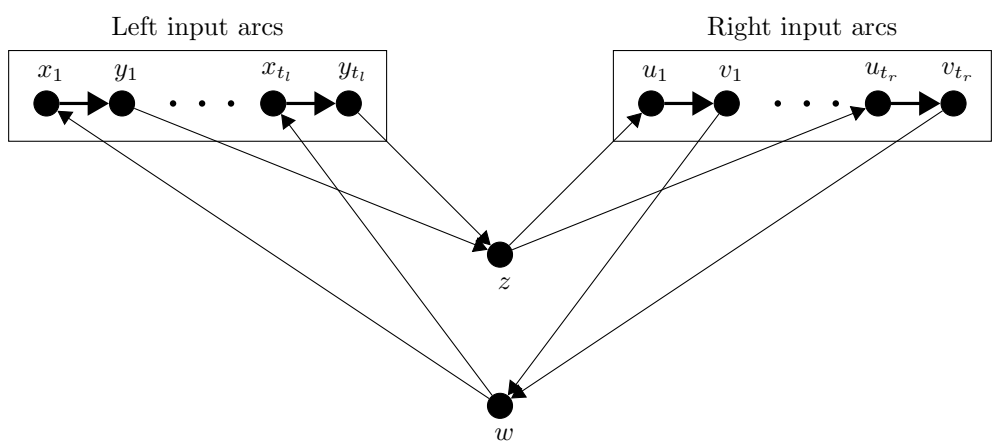


Figure 2: A Checksum gadget with t_l left input arcs and t_r right input arcs. A balanced subgraph will contain the same number of left and right input arcs.

Definition 3. *A* Checksum gadget *(see Figure 2) has t_l* left input *arcs $x_iy_i : i \in [t_l]$ for some positive integer t_l, and t_r* right input *arcs $u_iv_i : i \in [t_r]$, and no output arcs. The vertex set consists of the vertices w, z together with x_i, y_i for each $i \in [t_l]$ and u_i, v_i for each $i \in [t_r]$. The arc set consists of the path wx_iy_iz for each $i \in [t_l]$, and zu_iv_iw for each $i \in [t_r]$.*

Proposition 1 below is easy to prove and thus its proof is omitted.

Proposition 1. *Let D be one of the gadgets of Definitions 1-3, let X be a subset of input arcs in D, and let Y be a subset of output arcs in D. Then there exists a properly balanced subgraph D' of D containing all the arcs from X and Y (and no other input or output arcs) if and only if one of the following cases holds:*

(1) *$X = \emptyset$ and $Y = \emptyset$;*

(2) *D is a Duplication gadget, $|X| = 1$ and Y contains all the output arcs of D;*

(3) *D is a Choice gadget, $|X| = 1$ and $|Y| = 1$; or*

(4) *D is a Checksum gadget, and X contains an equal number of left input arcs and right input arcs.*

We next describe how to combine these gadgets. For two vertex-disjoint arcs uv and xy (possibly in disjoint graphs), the operation of *joining* uv and xy is as follows: Identify u and x, and identify v and y. Keep both uv and xy, and add (uv, xy) as a double arc.

Observe that in all of our gadgets, the vertices in input or output arcs all have in-degree and out-degree 1. Thus, the following technical lemma will be useful for showing the correctness of our constructions.

Lemma 1. *Let D_1 and D_2 be disjoint directed multigraphs. Let $u_1v_1, \ldots, u_tv_t$ be arcs in D_1, and let $x_1y_1, \ldots, x_ty_t$ be arcs in D_2, such that u_i and v_i both have in-degree and out-degree 1 in D_1, and x_i and y_i both have in-degree and out-degree 1 in D_2 for each $i \in [t]$. Let D be the graph formed by joining u_iv_i and x_iy_i for each $i \in [t]$. Then a subgraph D' of D is a properly balanced graph if and only if (1) $|A(D') \cap \{u_iv_i, x_iy_i\}| \neq 1$ for each $i \in [t]$; and (2) D' restricted to D_1 is a properly balanced subgraph of D_1, and D' restricted to D_2 is a properly balanced subgraph of D_2.*

Proof. Suppose first that D' is a properly balanced subgraph of D. The first condition holds by definition, since (u_iv_i, x_iy_i) is a double arc in D for each $i \in [t]$. For the second condition, let D'_1 be D' restricted to D_1. It is clear that D'_1 respects double and forbidden arcs in D_1. Furthermore, every vertex except u_i or v_i, $i \in [t]$, has the same in- and out-degree in D'_1 as in D', hence D'_1 is balanced for these vertices. Finally, for a vertex u_i or v_i, $i \in [t]$, note that such a vertex has in- and out-degree 2 in D, and by the double arc (u_iv_i, x_iy_i) either both these arcs are used in D', or neither. In both cases, we find that the restriction D'_1 is balanced. Hence D'_1 is balanced at every vertex, and respects both double and forbidden arcs, i.e., D'_1 is a properly balanced subgraph of D_1. An analogous argument holds for D_2.

Conversely, suppose that $|A(D') \cap \{u_iv_i, x_iy_i\}| \neq 1$ for each $i \in [t]$, D' restricted to D_1 is a properly balanced subgraph of D_1, and D' restricted to D_2 is a properly balanced subgraph of D_2. Then, by definition, D' respects double arcs and forbidden arcs of D. As D_1, D_2 partition the arcs of D, and D' is balanced when restricted to either of these graphs, we have that D' is balanced. Thus, D' is a properly balanced subgraph of D, as required. □

3.2 W[1]-hardness of PBS

We now use the gadget behavior, as described in Proposition 1, to construct a W[1]-hardness proof for PBS (in Theorem 2). By joining an output arc of one gadget to the input arc of another gadget, we have that a solution will only pass through the second gadget if it uses the corresponding output arc of the first gadget. Thus for example, if a Duplication gadget has k output arcs, each of which is joined to the input arc of a Choice gadget, then any solution that uses the input arc of the Duplication gadget has to use exactly one output arc from each of the Choice gadgets. By combining gadgets in this way, we can create "circuits" that represent instances of other problems.

We will use this idea to represent the following W[1]-hard problem. In k-Multicolored Clique, we are given a graph $G = (V_1 \cup V_2 \cdots \cup V_k, E)$, such that for each $i \in [k]$, V_i forms an independent set, and asked to decide whether G contains a clique with k vertices, where k is the parameter.

Theorem 1. *[9] k-Multicolored Clique is W[1]-hard.*

Our reduction is similar in structure to that of Dom *et. al* [8] for Capacitated Dominating Set parameterized by treewidth and solution size, although the details are rather different. We produce a large graph of constant pathwidth that represents a choice of one vertex v_i from each class V_i. In order to ensure that these vertices form a clique, the graph also requires that for each chosen vertex $v_i \in V_i$, we choose an edge between v_i and V_j, for each $j \neq i$. Finally, we add $O(k^2)$ vertices to check that for each $i \neq j$, the choice of edge between v_i and V_j is the same as the choice of edge between v_j and V_i.

The following technical lemma will also be useful for proving that PBS is W[1]-hard parameterized by pathwidth.

Lemma 2. *Let $D', D_1, D_2, \ldots, D_l$ be disjoint directed multigraphs, let $a_1, \ldots, a_l$ be distinct arcs in D', and let D be a graph formed by joining the arc a_i to an arc in D_i, for each $i \in [l]$. Then $pw(D) \leq pw(D') + \max_i(pw(D_i)) + 1$.*

Proof. Consider a minimum width path decomposition of D'. For each $i \in [l]$, let x_i be the bag in the path decomposition of D' that contains both vertices of a_i (if there is a choice of bags, let x_i be the bag of smallest size). Now replace x_i with two identical bags x_i', x_i'', and in between x_i' and x_i'' add a sequence of bags formed by taking a minimum width path decomposition of D_i and adding all the vertices of x_i to each bag. Do this for each $i \in [l]$. The resulting decomposition is a path decomposition of D. By construction and by choice of x_i, the width of this decomposition is at most $\mathrm{pw}(D') + \max_i(\mathrm{pw}(D_i)) + 1$. □

We are now ready to give a full proof of Theorem 2.

Definition 4. *We say that an instance of* PBS *is* semi-simple *if replacing each double arc and each forbidden pair by an edge yields a simple mixed graph (i.e., for every pair of vertices u, v, there is either one double arc, one forbidden pair, a single arc, or neither).*

Theorem 2. PBS *is W[1]-hard parameterized by pathwidth, even when the instance is semi-simple, there are no forbidden arcs, there is a single arc a^* of weight -1 and a^* is not part of a double arc, and all other arcs have weight* 0.

Proof. Given an instance $G = (V_1 \cup V_2 \cdots \cup V_k, E)$ of k-Multicolored Clique, let $e_1, \ldots e_m$ be an arbitrary enumeration of the edges of E. For each unordered pair $\{i, j\} \subseteq [k]$ with $i \neq j$, let $E_{\{i,j\}}$ be the subset of edges in E with one vertex in V_i and the other in V_j. Note that any k-clique in G will have exactly one edge from $E_{\{i,j\}}$ for each choice of i, j.

The structure of our PBS instance will force us to choose a vertex v_i from each class V_i, corresponding to the vertices of a k-clique. In addition, for each chosen vertex v_i and each $j \neq i$, we choose an edge $e_{i \to j}$ between v_i and V_j. A set of $O(k^2)$ Checksum gadgets will ensure that for each $i \neq j$, the chosen edges $e_{i \to j}$ and $e_{j \to i}$ must be the same. This ensures that v_i and v_j are adjacent for each $i \neq j$, and that therefore the vertices $v_1, \ldots, v_k$ form a clique.

We build our PBS instance (D, w, X, Y) with double arcs X and forbidden pairs Y out of Duplication, Choice, and Checksum gadgets as follows. Let Start be a Duplication gadget with input arc a^* and k output arcs. Label each output arc with a distinct integer i from $[k]$. For each $i \in [k]$, let ChooseVertex(i) be a Choice gadget with $|V_i|$ output arcs. Label each output arc with a distinct vertex v from V_i. Join the input arc of ChooseVertex(i) to the output arc of Start with label i.

For each $i \in [k], v \in V_i$, let AssignVertex(i, v) be a Duplication gadget with $k - 1$ output arcs. Label the output arcs with the integers from $[k] \setminus \{i\}$. Join the input arc of AssignVertex(i, v) to the output arc of ChooseVertex(i) with label v.

For each $i \in [k], v \in V_i, j \in [k] \setminus \{i\}$ let $\textsc{ChooseEdge}(i, v, \to j)$ be a Choice gadget with $|N(v) \cap V_j|$ output arcs. Label each output arc with a distinct edge e_r between v and V_j. Join the input arc of $\textsc{ChooseEdge}(i, v, \to j)$ to the output arc of $\textsc{AssignVertex}(i, v)$ with label j.

For each $i \in [k], v \in V_i, j \in [k] \setminus \{i\}$ and edge e_r between v_i and V_j, let $\textsc{AssignEdge}(i, v, \to j, e_r)$ be a Duplication gadget with r output arcs. Label this whole set of output arcs as $\textsc{Output}(i, v, \to j, e_r)$. Join the input arc of $\textsc{AssignEdge}(i, v, \to j, e_r)$ to the output arc of $\textsc{ChooseEdge}(i, v, \to j)$ with label e_r.

Finally, for each $i, j \in [k]$ with $i < j$, let $\textsc{CheckEdge}(i, j)$ be a Checksum gadget with $\sum\{r : e_r \in E_{\{i,j\}}\}$ left input arcs and $\sum\{r : e_r \in E_{\{i,j\}}\}$ right input arcs. Partition the left and right input arcs of $\textsc{CheckEdge}(i, j)$ as follows. For each $e_r \in E_{\{i,j\}}$, let $\textsc{Input}(i, v, \to j, e_r)$ be a set of r left input arcs, where v is the endpoint of e_r in V_i. Similarly, let $\textsc{Input}(j, u, \to i, e_r)$ be a set of r right input arcs, where u is the endpoint of e_r in V_j. Now, join each set of arcs of the form $\textsc{Input}(i, v, \to j, e_r)$ to the set of arcs of the form $\textsc{Output}(i, v, \to j, e_r)$ from the gadget $\textsc{AssignEdge}(i, v, \to j, e_r)$.

Finally, we assign weights and define X and Y. Let a^* have weight -1 and let all other arcs have weight 0. Let X be the set of double arcs created by the join operations above, and let $Y = \emptyset$. This concludes the construction of (D, w, X, Y). Observe that every output arc is joined to an input arc, and every input arc except a^* is joined to an output arc.

Correctness. We now show that D has a properly balanced subgraph of negative weight if and only if G has a clique with k vertices. Observe that by repeated use of Lemma 1, a subgraph D' of D is a properly balanced subgraph if and only if

- D' restricted to any gadget $\textsc{Start}$, $\textsc{ChooseVertex}(i)$, $\textsc{AssignVertex}(i, v)$, $\textsc{ChooseEdge}(i, v, \to j)$, $\textsc{AssignEdge}(i, v, \to j, e_r)$ or $\textsc{CheckEdge}(i, j)$ is a properly balanced subgraph for that gadget; and
- for each output arc a that is joined to an input arc a', a is in D' if and only if a' is in D'.

First suppose that D has a properly balanced subgraph D' of negative weight. Then D' must contain a^*, the input arc of $\textsc{Start}$ with weight -1. By Proposition 1, D' must contain all of the output arcs of $\textsc{Start}$. Thus for each $i \in [k]$, D' contains the input arc of $\textsc{ChooseVertex}(i)$. By Proposition 1, D' contains exactly one output arc of $\textsc{ChooseVertex}(i)$; let $v_i \in V_i$ be the unique vertex in G such that D' restricted to $\textsc{ChooseVertex}(i)$ contains the output arc labeled v_i.

It now follows that for each $i \in [k], v \in V_i$, D' contains the input arc of $\textsc{AssignVertex}(i, v)$ if and only if $v = v_i$. Then by Proposition 1, if $v = v_i$, then D' contains the all the output arcs of $\textsc{AssignVertex}(i, v)$, and otherwise D' contains none of the output arcs of $\textsc{AssignVertex}(i, v)$.

It follows that for each $i \in [k], v \in V_i, j \in [k] \setminus \{i\}$, D' contains the input arc of $\textsc{ChooseEdge}(i, v, \to j)$ if and only if $v = v_i$. If $v \neq v_i$ then, by Proposition 1, D' contains none of the output arcs of $\textsc{ChooseEdge}(i, v, \to j)$. If $v = v_i$, then again by Proposition 1 D' contains exactly one output arc of $\textsc{ChooseEdge}(i, v, \to j)$. So for each $i \in [k], j \in [k] \setminus \{i\}$, let $r(i \to j)$ be the index such that D' contains the output arc of $\textsc{ChooseEdge}(i, v_i, \to j)$ labeled with $e_{r(i \to j)}$. (Later we will show that $r(i \to j) = r(j \to i)$, implying that v_i and v_j are adjacent.)

It now follows that for each $i \in [k], v \in V_i, j \in [k] \setminus \{i\}$ and edge e_r between v_i and V_j, D' contains the input arc of $\textsc{AssignEdge}(i, v, \to j, e_r)$ if and only $v = v_i$ and $r = r(i \to j)$. Furthermore by Proposition 1, D' contains the set of output arcs $\textsc{Output}(i, v, \to j, e_r)$ if $v = v_i$ and $r = r(i \to j)$, and otherwise D' contains none of the arcs from $\textsc{Output}(i, v, \to j, e_r)$.

We now have that for each $i, j \in [k]$ with $i < j$, the left input arcs of $\textsc{CheckEdge}(i, j)$ in D' are exactly those in $\textsc{Input}(i, v_i, \to j, e_{r(i\to j)})$, and the right input arcs of $\textsc{CheckEdge}(i, j)$ in D' are exactly those in $\textsc{Input}(j, v_j, \to i, e_{r(j\to i)})$. By Proposition 1, we have that

$$|\textsc{Input}(i, v_i, \to j, e_{r(i\to j)})| = |\textsc{Input}(j, v_j, \to i, e_{r(j\to i)})|$$

and so $r(i \to j) = r(j \to i)$. It follows that $e_{r(i\to j)}$ and $e_{r(j\to i)}$ are the same edge, and that therefore this is an edge in G between v_i and v_j.

Thus we have that $v_1, \ldots, v_k$ form a clique in G, as required.

Now for the converse, suppose G has a clique on k vertices. By definition of G, this clique must have exactly one vertex from each class $V_i, i \in [k]$. For each $i \in [k]$, let v_i be the vertex of V_i that is in the clique. For each $i \neq j$, let $r(i, j)$ be the index such that $e_{r(i,j)}$ is the edge between v_i and v_j.

We will now describe a graph D' by describing its restriction to each gadget. The construction will be such that an output arc is in D' if and only if the input arc it is joined to is also in D'. Refer to a gadget as *passive* if no arcs in it are contained in D', and *active* otherwise; further, for a Choice gadget, say that it *selects* arc i if the i'th output arc is contained in D'. Note that by Proposition 1 these options all correspond to restrictions of balanced subgraphs to D'.

The graph D' is constructed as follows. The $\textsc{Start}$ gadget is active; for every $i \in [k]$, the $\textsc{ChooseVertex}(i)$ gadget is active and selects the output arc labeled v_i, and the $\textsc{AssignVertex}(i, v)$ gadget is active for $v = v_i$; and for every $i \in [k]$ and every $j \in [k]\backslash\{i\}$, the gadget $\textsc{ChooseEdge}(i, v, \to j)$ with $v = v_i$ is active, selecting the output arc labeled $e_{r(i,j)}$, and the $\textsc{AssignEdge}(i, v, \to j, e)$ gadget is active for $v = v_i$ and $e = e_{r(i,j)}$. All other $\textsc{AssignVertex}$, $\textsc{ChooseEdge}$ and $\textsc{AssignEdge}$ gadgets are passive. Note that D' contains an arc set $\textsc{Output}(i, v, \to j, e_r)$ if and only if $v = v_i$ and $r = r(i, j)$.

Finally, for each $i, j \in [k]$ with $i < j$, let D' restricted to $\textsc{CheckEdge}(i, j)$ be a properly balanced subgraph containing the left input arcs from $\textsc{Input}(i, v_i, \to j, e_{r(i,j)})$, the right input arcs from $\textsc{Input}(j, v_j, \to i, e_{r(j,i)})$, and no other input arcs. As $r(i, j) = r(j, i)$, such a subgraph exists by Proposition 1.

This concludes the construction of D'. As D' restricted to each gadget is a properly balanced subgraph, and an output arc is in D' if and only if the input arc it is joined to is in D', we have that D' is a properly balanced subgraph of D. As D' contains the arc a^* of weight -1 and all other arcs have weight 0, D' is a properly balanced subgraph with negative weight, as required.

Structure of the constructed graph and wrap-up of the proof. We have proved that D represents the instance of k-$\textsc{Multicolored Clique}$. Now it remains to show that D satisfies all properties given in the statement of this theorem, that $\mathrm{pw}(D)$ is bounded by a function of k, and that it can be constructed in fixed-parameter time.

We now address the properties of D in turn. It is clear that there exists a single arc a^* of weight -1, that a^* is not part of a double arc, that all other arcs have weight 0, and that there are no forbidden pairs. It is also clear from the construction that D is semi-simple. To see that pathwidth is bounded, let D^* be the graph derived from D by deleting the vertices w and z from every $\textsc{CheckEdge}$ gadget, along with incident arcs. (That is, D^* is the graph we had before adding $\textsc{CheckEdge}$ gadgets in the construction of D.) We constructed D^* by joining arcs in $\textsc{Start}$ to the input arcs of the $\textsc{ChooseVertex}(i)$ gadgets, then joining arcs of the resulting graph to the input arcs of the $\textsc{AssignVertex}(i, v)$ gadgets, then joining arcs of the resulting graph to the input arcs of the $\textsc{ChooseEdge}(i, v, \to j)$ gadgets, then joining arcs of the resulting graph to the input arcs of the $\textsc{AssignEdge}(i, v, \to j, e_r)$ gadgets. Observe that a Duplication gadget can be turned into a path by the removal of one vertex, and Choice and Checksum gadgets can be turned into disjoint unions of paths by the removal of two vertices. Therefore

Duplication gadgets have pathwidth 2, and Choice and Checksum gadgets have pathwidth 3. Hence by repeated use of Lemma 2, D^* has pathwidth at most $((((2+3+1)+2+1)+3+1)+2+1) = 16$.

There are $\binom{k}{2} = \frac{k^2-k}{2}$ CheckEdge gadgets, and therefore we can remove $k^2 - k$ vertices from D to get D^*. It follows that D has pathwidth at most k^2-k+16 (as we can add the k^2-k extra vertices to every bag in a path decomposition of D^*).

Finally, it is clear that the reduction can be performed in polynomial time, as we construct a polynomial number of gadgets and each gadget can be constructed in polynomial time. Thus we have provided a parameterized reduction from any instance of k-Multicolor Clique to an instance of PBS with the required properties and with pathwidth $O(k^2)$. □

4 Reducing PBS to MCPP

We now show how to reduce an instance of PBS, of the structure given in Theorem 2, to MCPP. This will prove that MCPP parameterized by pathwidth is W[1]-hard. Let $(D = (V, A), w, X = \{(a_i, a'_i) : i \in [t]\}, Y = \emptyset)$ be an instance of PBS with double arcs X and no forbidden pairs, and where $w(a^*) = -1$ for a single arc a^* and $w(a) = 0$ for every other arc. We may assume that a^* is not in a double arc. We will produce an instance G of MCPP and an integer W, such that G has a solution of weight W, and G has a solution of weight less than W if and only if our instance of PBS has a solution with negative weight. All edges and arcs in our MCPP instance will have weight 1.

We derive G by replacing every double arc and individual arc of D by an appropriate gadget. The gadgets will be such that within each gadget, there are only two possibilities for MCPP solutions of weight at most W: a solution can be balanced within the gadget (corresponding to not using the original arc/double arc in a solution to D), or a solution can be imbalanced at some vertices by the same amount that the original arc or double arc is (which corresponds to using the original arc or double arc in a solution to D). Thus, every solution for G of reasonable weight corresponds to a properly balanced subgraph of D, and vice versa.

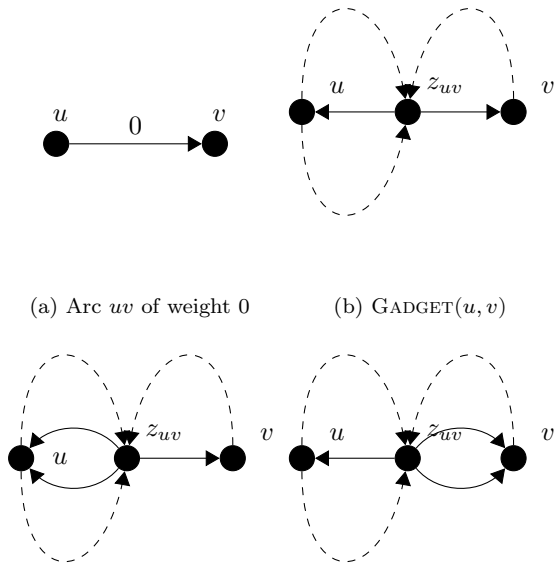
For each gadget, except for the gadget corresponding to the negative weight arc, the weights of the two solutions will be the same. In the case of the negative weight arc, the solution that corresponds to using the arc will be cheaper by 1. Thus, there are two possible weights for a solution to G, and the cheaper weight is only possible if D has a properly balanced subgraph of negative weight.

In what follows, we will construct arcs and edges of two weights, *standard* and *heavy*. Standard arcs and edges have weight 1; heavy arcs and edges have weight M, where M is a large enough (but polynomially bounded) integer such that we may assume that no solution traverses a heavy arc or edge more than once. This will be useful to impose structure on the possible solutions when constructing gadgets. A heavy arc (edge) is equivalent to a directed (undirected) path of length M, and so we also show W[1]-hardness for the unweighted case.

Given a directed multigraph H (corresponding to part of a solution to an MCPP instance) and a vertex v, the *imbalance* of v is $d^+_H(v) - d^-_H(v)$. The gadgets are constructed as follows.

For each double arc between a pair of vertices u, v, we produce a gadget $\textsc{Gadget}(u, v)$ that contains u, v and new vertices appearing in $\textsc{Gadget}(u, v)$. Similarly, we produce a gadget $\textsc{Gadget}(u, v)$ for any arc uv that is not part of a double arc, with the structure of the gadget depending on the weight of the arc.

For an arc uv of weight 0 that is not part of a double arc: Construct $\textsc{Gadget}(u, v)$ by creating a new vertex z_{uv}, with standard arcs $z_{uv}u$ and $z_{uv}v$, two heavy arcs uz_{uv}, and a heavy arc vz_{uv}. (See Figure 3.)



(a) Arc uv of weight 0 (b) $\textsc{Gadget}(u, v)$

(c) Passive solution (d) Active solution

Figure 3: The gadget $\textsc{Gadget}(u, v)$ when uv is an arc of weight 0 that is not part of a double arc. Dashed lines represent heavy arcs.

For any solution in which each heavy arc is traversed exactly once, z_{uv} has in-degree exactly 3. The remaining arcs are the two arcs with tail z_{uv}, which must be traversed exactly three times in total as described below:

- **Passive Solution:** Traverse $z_{uv}u$ twice and $z_{uv}v$ once. In this solution, every vertex is balanced within $\textsc{Gadget}(u, v)$ and the cost is $3M + 3$.
- **Active Solution:** Traverse $z_{uv}u$ once and $z_{uv}v$ twice. In this solution, every vertex is balanced except for u, which has imbalance 1, and v, which has imbalance -1. The cost of this solution is also $3M + 3$.

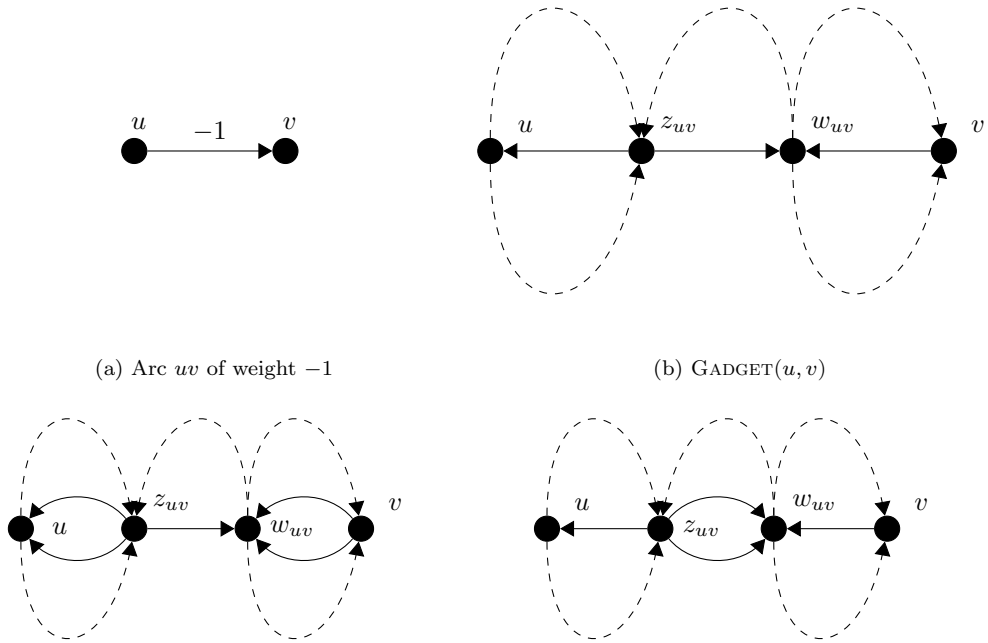
Observe that the difference between the weight of the passive and active solutions is 0, and the imbalance at u and v for the active solution is the same as in an arc from u to v.

The total weight of $\textsc{Gadget}(u, v)$ is $3M + 2$.

For an arc uv of weight -1 that is not part of a double arc: Construct $\textsc{Gadget}(u, v)$ by adding two new vertices w_{uv} and z_{uv}, with standard arcs $z_{uv}u, z_{uv}w_{uv}$ and vw_{uv}, two heavy arcs uz_{uv}, one heavy arc $w_{uv}z_{uv}$, and two heavy arcs $w_{uv}v$. (See Figure 4.)

For any solution in which each heavy arc is traversed exactly once, z_{uv} will have exactly 3 in-arcs, and w_{uv} will have exactly 3 out-arcs. It remains to decide how many times to use the arcs out of z_{uv} and into w_{uv}. There are two possible solutions:

- **Passive Solution:** Traverse $z_{uv}u$ twice, $z_{uv}w_{uv}$ once and vw_{uv} twice. In this solution, every vertex is balanced within $\textsc{Gadget}(u, v)$ and the cost is $5M + 5$.



(a) Arc uv of weight -1

(b) $\textsc{Gadget}(u, v)$

(c) Passive solution

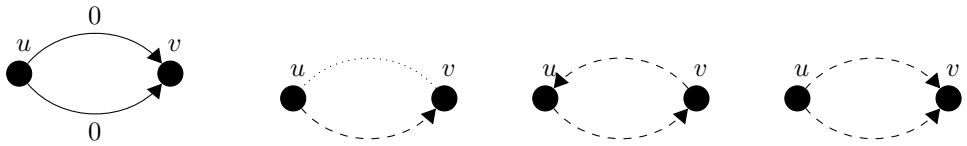
(d) Active solution

Figure 4: The gadget $\textsc{Gadget}(u, v)$ when uv is an arc of weight -1 that is not part of a double arc. Dashed lines represent heavy arcs.

- **Active Solution:** Traverse z_{uv} once, $z_{uv}w_{uv}$ twice and vw_{uv} once. In this solution, every vertex is balanced except for u, which has imbalance 1, and v, which has imbalance -1. The cost of this solution is $5M + 4$.

Observe that the active solution costs 1 less than the passive solution, and the imbalance at u and v for the active solution is again the same as in an arc from u to v.

The weight of this gadget is $5M + 3$.



(a) A double arc from u to v (b) $\textsc{Gadget}(u, v)$ (c) Passive solution (d) Active solution

Figure 5: The gadget $\textsc{Gadget}(u, v)$ when there is a double arc from u to v. Dashed lines represent heavy arcs; the dotted line represents a heavy edge.

For a double arc from u to v: $\textsc{Gadget}(u, v)$ consists of a heavy arc uv and a heavy edge $\{u, v\}$. (See Figure 5.) Assuming a solution traverses each heavy arc/edge exactly once,

it remains to decide in which direction to traverse the undirected edge. Thus there are two possible solutions:

- **Passive Solution:** Traverse the edge from v to u. In this solution, every vertex is balanced within $\textsc{Gadget}(u, v)$ and the cost is $2M$.
- **Active Solution:** Traverse the edge from u to v. In this solution, every vertex is balanced except for u, which has imbalance 2, and v, which has imbalance -2. The cost of this solution is also $2M$.

Observe that the difference between the weight of the passive and active solutions is 0, and the imbalance at u and v for the active solution is the same as in a double arc from u to v.

The weight of this gadget is $2M$.

Removing heavy arcs and edges. Finally, we replace every arc (edge, respectively) we just created of weight M by a directed (undirected) path of length M, where all internal vertices have in-degrees and out-degrees one (degree two, respectively). Note that in any minimal solution, each arc or edge in such a path will be traversed the same number of times (if one edge in the path is traversed more times than its neighbor, then it must be traversed at least twice more, including at least once in each direction, and so the solution is not minimal). Thus in the analysis, we may treat such a path as effectively being a single edge or arc of weight M.

In the proof of the main result of this section, Theorem 3, we will use the following technical lemma, whose proof is identical to that of Lemma 2.

Lemma 3. *Let H be a mixed multigraph, and G the mixed multigraph derived by replacing each arc or double arc from u to v with a gadget G_{uv}. Then $pw(G) \leq pw(H) + \max_{uv} pw(G_{uv})$.*

We can now prove the following:

Theorem 3. MCPP *is W[1]-hard parameterized by pathwidth even if all weights are 1.*

Proof. Let $(D = (V, A), w, X = \{(a_i, a'_i) : i \in [t]\}, Y = \emptyset)$ be a semi-simple instance of PBS (see Definition 4) with double arcs X and no forbidden pairs, and where $w(a^*) = -1$ for a single arc a^* and $w(a) = 0$ for every other arc. Let G be the graph in which every arc / double arc in D is replaced with its corresponding gadget.

Let m_1 be the number of arcs of weight 0 not in a double arc in D. By Theorem 2, we may assume there is only one arc of weight -1 not in a double arc in D. Let m_2 be the number of double arcs in D.

If we use the passive solution for every gadget, then every vertex is balanced, every arc and edge is covered, and the total cost is $m_1(3M + 3) + 5M + 5 + m_2(2M) = (3m_1 + 2m_2 + 5)M + 3m_1 + 5$. Therefore this is an upper bound on the weight of a optimal solution.

The total weight of the graph is $m_1(3M+2)+5M+3+m_2(2M) = (3m_1+2m_2+5)M+2m_1+3$. Therefore, any minimal solution that does not traverse each heavy arc/edge exactly once will have weight at least $(3m_1 + 2m_2 + 6)M + 2m_1 + 3$. This is $M - m_1 - 2$ greater than the solution in which we use the passive solution for every gadget. So by setting M to be $m_1 + 3$, we may assume that the optimal solution traverses each heavy arc/edge exactly once. Therefore we may assume that we use either the active or passive solution for each gadget.

Let $W = (3m_1 + 2m_2 + 5)M + 3m_1 + 5$, the cost of using the passive solution for each gadget. We now show that G has a solution of weight less than W if and only if D has a solution of negative weight.

Suppose first that G has a solution of weight less than W. As discussed above, we may assume that every gadget is either given the active or passive solution. Let u^*, v^* be the vertices

such that u^*v^* is the only arc in D of weight -1. Then $\textsc{Gadget}(u^*, v^*)$ is the only gadget for which one solution weighs less than the other. Therefore we may assume $\textsc{Gadget}(u^*, v^*)$ is given the active solution.

Let D' be the subgraph of D whose arc set is the set of all arcs whose corresponding gadget in G has the active solution (with both arcs in a double arc included if their gadget has the active solution, and neither if it has the passive solution). Then D' contains u^*v^* and so D has negative weight. By construction, D' respects double arcs, and, as there are no forbidden pairs, D' trivially respects forbidden pairs. It remains to show that D' is balanced.

In our MCPP solution, the imbalance of a vertex v is equal to the sum of its imbalance in the active gadgets containing it. The imbalance of v in an active gadget is $+1$ if the gadget corresponds to a single arc with tail v, -1 if the gadget corresponds to a single arc with head v, $+2$ if the gadget corresponds to a double arc starting at v, and -2 if the gadget corresponds to a double arc ending at v. It follows that the imbalance of v in our MCPP solution is equal to its imbalance in D', Then as our MCPP solution is balanced, D' is also balanced, as required.

Suppose on the other hand that D has a solution D' of negative weight. Construct a solution G^* to MCPP on G by assigning each gadget the active assignment if the corresponding arc/double-arc appears in D', and the passive assignment otherwise. As D has negative weight, it must use u^*v^* and so $\textsc{Gadget}(u^*, v^*)$ gets the active solution. It follows that the cost of G^* is $W - 1$. It is clear that every arc and edge is traversed at least once. As before, the imbalance of each vertex in G^* is equal to its imbalance in D', implying that G^* is balanced. Furthermore, G^* is strongly connected, as G is. An Euler trail in this multigraph corresponds to a closed walk G traversing every arc and edge at least once, as required.

We now show that G has pathwidth bounded by a function of $\mathrm{pw}(D)$, the pathwidth of D.

Observe that for each gadget $\textsc{Gadget}(u, v)$ in our construction of G, $\textsc{Gadget}(u, v)$ can be turned into a disjoint union of paths by the removal of at most 4 vertices, and therefore $\textsc{Gadget}(u, v)$ has pathwidth at most 5. Furthermore, G can be derived from D by replacing each arc or double arc with a corresponding gadgets. It follows from Lemma 3 that G has pathwidth at most $\mathrm{pw}(D) + 5$.

Let m be the number of arcs in D. Then G is derived from D by introducing at most m gadgets, and each gadget has at most $5M \leq 5m + 15$ arcs. Therefore G can be constructed in $O(m^2)$ time.

We now have that, given an instance (D, w, X, Y) of PBS of the type specified in Theorem 2, we can in polynomial time create an instance G of MCPP with pathwidth bounded by $\max(\mathrm{pw}(D), 4)$. We therefore have a parameterized reduction from this restriction of PBS, parameterized by pathwidth, to MCPP parameterized by pathwidth. □

5 PBS Parameterized by Treedepth

In this section we show that a certain restriction of PBS is fixed-parameter tractable with respect to treedepth. In Section 7 we discuss how to generalize the results to the arbitrary integer-weighted case.

Aside from some standard dynamic programming techniques, our main technical tool is Lemma 5, which implies that there exists a solution with size bounded by a function of treedepth. The following simple observation will be useful in the proof of Lemma 5.

Lemma 4. *Let $\mathcal{I}$ be a set of indices and $\{H_i : i \in \mathcal{I}\}$ a family of pairwise arc-disjoint subgraphs of G, such that each H_i respects double arcs. Then $H = \bigcup_{i \in \mathcal{I}} H_i$ is a properly balanced subgraph of G if and only if H is balanced and H respects forbidden pairs.*

We are now ready to prove that any properly balanced subgraph decomposes into properly balanced subgraphs of size bounded by a function of treedepth. This will allow us to assume, when constructing the algorithm (Lemma 6), that a solution has bounded size.

Lemma 5. *Let G be a directed multigraph (with double arcs and forbidden pairs) of treedepth k, and let H be a properly balanced subgraph of G. Then H is a union of pairwise arc-disjoint graphs H_i, each of which is a properly balanced subgraph of G, with $|A(H_i)| \leq f(k)$ where $f(k) = 2^{2^k}$.*

Proof. We prove the claim by induction on the treedepth k. For the base case, observe that if $k = 1$ then G has no arcs, and the claim is trivially true. So now assume that $k \geq 2$, and that the claim holds for all graphs of treedepth less than k. We also assume that the underlying undirected graph of H is 2-connected, as otherwise a block decomposition of H is a decomposition into properly balanced subgraphs, and we may apply our result to each block of H. Similarly, if the underlying undirected graph of G is not 2-connected but H is, then H lies inside one block of G, and we may restrict our attention to this block. Hence assume that the underlying undirected graph of G is 2-connected as well.

Let G be embedded in a tree T of depth k, and let x be the root of T. Observe that x has only one child in T, as otherwise x is a cut-vertex in G. Let y be this child, and let G' be the multigraph derived from G by identifying x and y and removing loops. Similarly, let H' be the subgraph of G' derived from H by identifying x and y and removing loops. Observe that H' is balanced as H is balanced and so the number of arcs into $\{x, y\}$ equals the number of arcs out of it. Let B be the set of arcs in H between x and y, and observe that there is a one-to-one correspondence between the arcs of H' and the arcs of H not in B. By identifying x and y in T, we get that G' has treedepth at most $k - 1$.

By the induction hypothesis, H' can be partitioned into a family $\{H'_i : i \in \mathcal{I}'\}$ of pairwise arc-disjoint properly balanced subgraphs of G', each having at most $f(k-1)$ arcs. For each $i \in \mathcal{I}'$, let F_i be the subgraph of G corresponding to H'_i. Observe that B can also be partitioned into a family $\{F_i : i \in \mathcal{I}''\}$ of subgraphs with at most 2 arcs, that respect double arcs (we add any double arc from B as a subgraph F_i, and add every other arc as a single-arc subgraph).

Letting $\mathcal{J} = \mathcal{I}' \cup \mathcal{I}''$, we have that $\{F_i : i \in \mathcal{J}\}$ is a partition of H, each F_i has at most $f(k-1)$ arcs, and each F_i respects double arcs and is balanced everywhere except possibly at x and y. We now combine sets of these subgraphs into subgraphs that are balanced everywhere.

For each $i \in \mathcal{J}$, let t_i be the imbalance of F_i at x, i.e. $t_i = d^+_{F_i}(x) - d^-_{F_i}(x)$. Observe that $|t_i| \leq \frac{f(k-1)}{2}$ for each i and, as H is balanced, $\sum_{i \in \mathcal{J}} t_i = 0$. Suppose that there exists a subset $\mathcal{J}' \subseteq \mathcal{J}$ such that $|\mathcal{J}'| \leq f(k-1)-1$ and $\sum_{i \in \mathcal{J}} t_i = 0$. Then define H_1 to be $\bigcup_{i \in \mathcal{J}'} F_i$. By construction, H_1 is balanced at every vertex (as it is balanced for every vertex other than y, and a directed multigraph cannot be imbalanced at a single vertex), and H_1 respects double arcs. As H_1 is a subgraph of H, and H respects forbidden pairs (since H is properly balanced), H_1 also respects forbidden pairs. Therefore H_1 is a properly balanced subgraph, with number of arcs at most $(f(k-1)-1)f(k-1)$. Observe that $f(k) = 2^{2^k}$ is a solution to the recursion $(f(k-1)-1)f(k-1) < f(k)$ with $f(1) = 4$. Thus, H_1 has at most 2^{2^k} arcs, as required. By applying a similar argument to $H \setminus H_1$, we get a properly balanced subgraph H_2 with at most $f(k)$ arcs. Repeating this process, we get a partition of H into properly balanced subgraphs each with at most $f(k)$ arcs.

We now show that $\mathcal{J}'$ exists. Initially choose $j \in \mathcal{J}$ such that $|t_j|$ is minimized, and set $\mathcal{J}_1$ to be $\{j\}$. Then iteratively construct sets $\mathcal{J}_r$ by adding i such that $t_i < 0$ to $\mathcal{J}_{r-1}$ if $\sum_{p \in \mathcal{J}_{r-1}} t_p > 0$, and adding i such that $t_i > 0$ otherwise. Now note that either $t_i = \pm f(k-1)/2$ for every $i \in \mathcal{J}$, in which case we have a subset $\mathcal{J}'$ with $|\mathcal{J}'| = 2 \leq f(1) - 1 \leq f(k-1) - 1$, or $|\sum_{i \in \mathcal{J}_r} t_i| < \frac{f(k-1)}{2}$ for each r, and therefore there are at most $f(k-1) - 1$ possible values

that $\sum_{i \in \mathcal{J}_r} t_i$ can take. Then there exist r, r' such that $r' < r$, $r - r' \leq f(k-1) - 1$, and $\sum_{i \in \mathcal{J}_r \setminus \mathcal{J}_{r'}} t_i = 0$. So let $\mathcal{J}' = \mathcal{J}_r \setminus \mathcal{J}_{r'}$.

□

Using Lemma 5, we may now assume that if G has a properly balanced subgraph with negative weight, then it has a properly balanced subgraph of negative weight with at most $f(k)$ arcs (as any negative weight properly balanced subgraph can be partitioned into properly balanced subgraphs of at most $f(k)$ arcs, at least one of which must have negative weight).

5.1 Fixed-Parameter Tractability of PBS

In the Appendix, we prove the following lemma:

Lemma 6. *Given an instance $(D = (V, A), w, X, Y)$ of* PBS *with integer weights, a path decomposition of the underlying undirected graph of D with width k, and an integer l, we can find a properly balanced subgraph of D of weight at most W in time $O^*(2^{k^2+3k}(2l+1)^{k+1})$, where W is the minimum weight of a properly balanced subgraph of D with at most l arcs.*

A graph with treedepth at most k has pathwidth at most $k-1$ [3]. As PBS asks us to find a properly balanced subgraph of negative weight, and as we may assume such a graph exists with at most $f(k)$ arcs, Lemma 6 implies the following:

Theorem 4. PBS *with integer weights is FPT with respect to treedepth.*

The proof in the Appendix uses standard dynamic programming techniques.

For the purposes of solving MCPP with unit weights, it will be enough to show that PBS is FPT with respect to treedepth when all arcs in double arcs have weight 0, all arcs in forbidden pairs have weight -1, and all other arcs have weight 1 or -1. For this special case, we can give a proof using Courcelle's theorem:

Proof. Let D be an instance of PBS and let k be the treedepth of D. By Lemma 5, we may assume that a solution has at most $f(k)$ arcs. We will prove the theorem by guessing all possible solutions D' (up to isomorphism), and then checking whether D' is isomorphic to a subgraph of D using Courcelle's theorem.

First, we assign each arc a of D the label Double if a is in a double arc, the label Forbidden if a is in a forbidden pair, and otherwise the label Negative if a has weight -1 and Positive if a has weight 1. Next, enumerate all balanced directed multigraphs D' with at most $f(k)$ arcs, together with all possible labelings $\lambda : (A(D')) \to \{\text{Negative}, \text{Positive}, \text{Double}, \text{Forbidden}\}$.

We may assume that no vertex has degree 1 in D', as otherwise D' cannot be balanced. Thus we may assume every vertex in D' has degree at least 2, and therefore we may assume D' has at most $f(k)$ vertices. For each $i \in [f(k)]$, either D' contains fewer than i arcs, or the i'th arc is from one of $f(k)$ vertices to one of $f(k)$ vertices, and has one of four possible labels. Therefore there are at most $1 + 4f(k)^2$ choices for the identity (or non-existence) of each arc. Thus, we can enumerate all possible graphs D' in time $O^*((1 + 4f(k)^2)^{f(k)})$.

For each constructed directed multigraph D', we first check whether D' would be a properly balanced subgraph of D, if it existed as a subgraph of D. This can be done by checking, for each pair of vertices, that D' has at most one arc labeled Forbidden, and either 0 or 2 arcs (in the same direction) labeled Double. If D' is not a properly balanced subgraph, we disregard it. Otherwise, we then check whether D' has negative weight. (This can be done by checking that the number of Negative and Forbidden arcs is greater than the number of Positive arcs.)

If D' has negative weight, it remains to check whether D' is isomorphic to a subgraph of D (respecting the labels). As D has treewidth at most k and D' has at most $f(k)$ vertices, this can be done in fixed-parameter time by Courcelle's theorem (see Theorem 11.37 in [14]). □

6 Positive Result: Reducing MCPP to PBS

In this section, we consider MCPP parameterized by treedepth. To simplify the exposition, we restrict ourselves to the case where all weights equal 1; this restriction will be lifted in the next section. In contrast to pathwidth, we will show that MCPP parameterized by treedepth is FPT. Hereinafter, $b_H(v)$ denotes the imbalance of v, i.e. $b_H(v) = d_H^+(v) - d_H^-(v)$. In the problem COMP-MCPP, we are given an instance of MCPP together with a solution H, and asked to find a solution H' of weight less than $w(H)$, if one exists. To solve an instance of MCPP, it would be enough to find some (not necessarily optimal) solution of weight M, then repeatedly apply COMP-MCPP to find better solutions, until we find a solution H for which no solution H' of weight less than $w(H)$ exists, i.e. an optimal solution. As COMP-MCPP returns an improved solution if one is available, we would have to apply COMP-MCPP at most M times. For the case that all weights are 1, this is sufficient to give an FPT algorithm. For the weighted case, we need some further careful analysis, which we defer to the next section.

As with the hardness proof, we will use PBS as an intermediate problem, giving a reduction from COMP-MCPP to PBS. In order to bound the size of the gadgets we produce for this reduction, we need a bound on the number of times an arc or edge can be used in a solution to MCPP. We also need a bound on the size of an initial solution to MCPP, in order to bound the number of times we need to improve the solution. In the case when all weights are 1, Lemma 7 (below) is sufficient to provide a bound on the number of times each edge or arc is used in a solution, and hence on the size of a solution. For the weighted case, we also need Lemma 8.

In the rest of the paper, m will denote the number of arcs and edges in the instance (G, w) under consideration.

Lemma 7. *Given an instance (G, w) of* MCPP*, we can, in polynomial time, find a closed walk of G that traverses each edge and arc at least once, if such a walk exists, and this walk traverses each arc at most $m + 1$ times, where m is the number of arcs and edges in G.*

Proof. For each arc or edge a in G, we will construct a closed walk WALK_a that contains a. If a is an edge, we may let WALK_a be a walk that traverses a once in each direction. Then this is a closed walk that traverses a twice and traverses no other arcs or edges. If a is an arc, let u be the tail and v the head of a. In polynomial time, find a simple walk from v to u. (If no such walk exists, then (G, w) has no solution). Adding $a = uv$ to this walk, we have a closed walk that traverses a once and each other edge or arc at most once. Taking the union of walks WALK_a for every edge and arc a, we get a walk that traverses each arc or edge a at most $m + 1$ times (at most twice by WALK_a, and at most once for all other walks). □

Lemma 8. *Let $(G = (V, E \cup A), w)$ be an instance of* MCPP*. For every solution H to (G, w), there is a solution H' which is a subset of H where each arc and edge has multiplicity at most $m + 1$, where m is the number of arcs and edges in G. Furthermore, such a solution H' can be found in polynomial time.*

Proof. Let $e \in A \cup E$ be an edge or arc of G that is repeated more than $m + 1$ times in H. Consider the decomposition of H into directed cycles (this can be found greedily). Let a cycle in this decomposition be *critical* if it contains the only occurrence(s) of some arc or edge e' in $A \cup E$, and observe that if C is a non-critical cycle in H, then $H - C$ is also a solution. Assume thus that every cycle in the decomposition that contains e is critical. Then this is

either a single cycle, in which case e has multiplicity at most $2 \leq |A \cup E| + 1$ in H, or by a pigeonhole argument there are at most $m - 1$ such cycles. The lemma follows. □

We now reduce COMP-MCPP to PBS, in the following sense: For any input graph G and initial solution H, we produce a directed multigraph D such that D has a properly balanced subgraph of negative weight if and only if G has a solution of weight less than $w(H)$. The multigraph D has double arcs and forbidden pairs; a double arc represents re-orienting an edge from one direction to the other, and a forbidden pair represents two arcs in opposite directions, such that we may remove one but not both.

For any adjacent vertices u, v in G, let G_{uv} be the subgraph of G induced by $\{u, v\}$, and similarly let H_{uv} be the subgraph of H induced by $\{u, v\}$. We say H_{uv} traverses uv from u to v (from v to u) exactly t times if H_{uv} contains exactly t copies of the arc uv (vu). Note that by Lemma 8, we may assume H_{uv} traverses uv from u to v at most $m + 1$ times for each $u, v \in V(G)$.

For each edge and arc uv in G, we will produce a gadget D_{uv}, based on G_{uv} and H_{uv}. The gadget D_{uv} is a directed multigraph, possibly containing double arcs or forbidden pairs, and by combining all the gadgets, we will get an instance D of PBS.

We now construct D_{uv} according to the following cases (roughly speaking, including a positive weight arc in the solution to the PBS instance represents adding an arc in that direction to H, and including a negative weight arc represents removing an arc in the opposite direction from H):

If G_{uv} is an arc from u to v and H_{uv} traverses uv exactly $t \leq m+1$ times: Then D_{uv} has $t - 1$ arcs from v to u of weight $-w(uv)$, and $m + 1 - t$ arcs from u to v of weight $w(uv)$.

If G_{uv} is an edge between u and v, and H_{uv} traverses uv from u to v exactly $t \leq m+1$ times, and from v to u exactly 0 times: Then D_{uv} has a double arc (a, a'), where a and a' are both arcs from v to u of weight 0. In addition, D_{uv} has $t - 1$ arcs from v to u of weight $-w(uv)$, $m + 1 - t$ arcs from u to v of weight $w(uv)$, and m arcs from v to u of weight $w(uv)$.

If G_{uv} is an edge between u and v and H_{uv} traverses uv from u to v exactly $t > 0$ times, and from v to u exactly $s > 0$ times: Then we may assume $s = t = 1$, as otherwise we may remove a pair of arcs (uv, vu) from H and get a better solution to MCPP. Then D_{uv} has m arcs from u to v of weight $w(uv)$, m arcs from v to u of weight $w(uv)$, and a forbidden pair (a, a'), where a is an arc from u to v, a' is an arc from v to u, and both a and a' have weight $-w(uv)$.

The next lemma shows that the gadgets D_{uv} behave as we require. Intuitively, B represents the imbalance at one vertex in a subgraph of D_{uv}, and the change in balance given by the corresponding change in solution on G_{uv}. As these numbers are the same, a balanced subgraph in D corresponds to a change in solution on G that remains balanced. W represents the weight of a subgraph in D_{uv}, and the corresponding change in weight of a solution on G_{uv}. As these are the same, a negative weight solution in D corresponds to an improved solution on G.

Lemma 9. *Let uv be an edge or arc in G, and let B and W be arbitrary integers such that $w(H_{uv}) + W \leq (m + 1)w(uv)$. Then the following are equivalent.*

1. *There exists a graph H'_{uv} with vertex set $\{u, v\}$ that covers G_{uv} such that $w(H'_{uv}) = w(H_{uv}) + W$ and the imbalance $b_{H'_{uv}}(u) = b_{H_{uv}}(u) + B$;*

2. *D_{uv} has a subgraph D'_{uv} which respects double arcs and forbidden pairs such that $w(D'_{uv}) = W$ and $b_{D'_{uv}}(u) = B$.*

Proof. We consider each of the cases separately.

If G_{uv} is an arc from u to v and H_{uv} traverses uv exactly $t \leq m+1$ times: Then recall D_{uv} has $t-1$ arcs from v to u of weight $-w(uv)$, and $m+1-t$ arcs from u to v of weight $w(uv)$. Observe that $w(H_{uv}) = tw(uv)$.

(1) $\to$ (2): Observe that (1) can only hold if $W = Bw(uv)$. As H'_{uv} covers G_{uv}, we have $1 \leq b_{H_{uv}}(u) + B \leq m+1$, and therefore $1-t \leq B \leq m+1-t$. If $B = 0$, let D'_{uv} have no arcs. If $B > 0$, let D'_{uv} have B positive weight arcs from u to v. If $B < 0$, let D'_{uv} have $|B|$ negative weight arcs from v to u. Observe that in each case D'_{uv} is a subgraph of D_{uv} and satisfies condition 2 (D'_{uv} trivially satisfies double arcs and forbidden pairs, as there are none of these with arcs in D_{uv}).

(2) $\to$ (1): Observe that by construction of D_{uv}, $W = Bw(uv)$, and also observe that the minimum possible weight of D'_{uv} is $-(t-1)w(uv)$ and so $1-t \leq B$. If $B = 0$, then let $H'_{uv} = H_{uv}$ and observe that H'_{uv} satisfies condition 1. If $B > 0$, let H'_{uv} be H_{uv} with the addition of B extra arcs from u to v. Then $w(H'_{uv}) = w(H_{uv}) + W$, and H'_{uv} satisfies condition 1. If $B < 0$, let H'_{uv} be H_{uv} with $|B|$ arcs from u to v removed. As $|B| < t$, H'_{uv} still has at least one arc from u to v and so covers G_{uv}, and condition 1 is satisfied.

If G_{uv} is an edge between u and v, and H_{uv} traverses uv exactly $t \leq m+1$ times from u to v, and from v to u exactly 0 times: Then recall that D_{uv} has a double arc (a, a'), where a and a' are both arcs from v to u of weight 0. In addition, D_{uv} has $t-1$ arcs from v to u of weight $-w(uv)$, $m+1-t$ arcs from u to v of weight $w(uv)$, and m arcs from v to u of weight $w(uv)$.

(1) $\to$ (2): Let t' be the number of arcs from u to v in H'_{uv}, and s' the number of arcs from v to u in H'_{uv}. Observe that $W = (s'+t'-t)w(uv)$ and $B = t'-t-s'$.

If $s' = 0$, then $t' > 0$ and we have $W = (t'-t)w(uv) = Bw(uv)$. If $B = 0$, then let D'_{uv} have no arcs. If $B > 0$, then note that $B \leq m+1-t$ and let D'_{uv} have B positive weight arcs from u to v. If $B < 0$, then note that $|B| \leq t-1$ (as otherwise H'_{uv} does not cover uv) and let D'_{uv} have $|B|$ negative weight arcs from v to u. Observe that in each case D'_{uv} satisfies condition 2.

If $t' = 0$, then $s' > 0$ and we have $W = (s'-t)w(uv)$ and $B = -t-s'$. Then let D'_{uv} have all $t-1$ negative weight arcs from v to u, both arcs in the double arc from v to u, and $s'-1$ positive weight arcs from v to u. Observe that D'_{uv} has weight $(s'-1-(t-1))w(uv) = W$ and $b_{D'_{uv}}(u) = -t-s' = B$, and D'_{uv} respects double arcs, and so D'_{uv} satisfies condition 2.

If $t' > 0$ and $s' > 0$, then if $t'+s' > 2$ we may remove a pair of arcs (uv, vu) from H' and get a better solution to MCPP. Therefore we may assume $t' = s' = 1$ and so $W = (2-t)w(uv)$ and $B = -t$. If $t \geq 2$ let D'_{uv} contain $t-2$ negative weight arcs from v to u, and both arcs of the double arc from v to u. Otherwise $t = 1$. In this case, let D'_{uv} contain both arcs of the double arc from v to u, and one positive weight arc from u to v. In either case, D'_{uv} has weight $-(t-2)w(uv) = W$ and $b_{D'_{uv}}(u) = -t = B$, and D'_{uv} respects double arcs, and so D'_{uv} satisfies condition 2.

(2) $\to$ (1): Let s' be the number of positive weight arcs from u to v in D'_{uv}, t' the number of negative weight arcs from v to u in D'_{uv}, and r' the number of positive weight arcs from v to u in D'_{uv}. Then $W = (s'-t'+r')w(uv)$.

Suppose first that D'_{uv} does not contain the double arc. Then $B = s'-t'-r'$. If $s'-t' = 0$, then let H'_{uv} be H_{uv} with r' arcs from v to u added. If $s'-t' > 0$, then let H'_{uv} be H_{uv} with $s'-t'$ arcs from u to v added and r' arcs from v to u added. If $s'-t' < 0$, then let H'_{uv} be H_{uv} with $t'-s'$ arcs from u to v removed and r' arcs from v to u added (note that as $t' < t$, removing $t'-s'$ arcs from u to v still leaves uv covered). Observe that in each case, H'_{uv} satisfies condition 1.

Now suppose that D'_{uv} contains the double arc. Then $B = s'-t'-r'-2$. If $s'-t' = 0$, then let H'_{uv} be H_{uv} with one arc from u to v removed and $r'+1$ arcs from v to u added. If $s'-t' > 0$, then let H'_{uv} be H_{uv} with $s'-t'-1$ arcs from u to v added and $r'+1$ arcs

from v to u added. If $s' - t' < 0$, then let H'_{uv} be H_{uv} with $t' - s' + 1$ arcs from u to v removed and $r' + 1$ arcs from v to u added. Observe that in each case, H'_{uv} satisfies condition 1.

If G_{uv} is an edge between u and v and H_{uv} traverses uv exactly $t > 0$ times from u to v, and exactly $s > 0$ times from v to u: Then recall that we may assume $s = t = 1$, and D_{uv} has a forbidden pair (a, a'), where a is an arc from u to v, a' is an arc from v to u, and both a and a' have weight $-w(uv)$. In addition, D_{uv} has m arcs from u to v of weight $w(uv)$, and m arcs from v to u of weight $w(uv)$.

(1) $\to$ (2): If $H'_{uv} = H_{uv}$, then let D'_{uv} have no arcs. Otherwise, we may assume all arcs in H'_{uv} are in the same direction (as otherwise we may remove a pair of arcs $\{uv, vu\}$ to get a better solution). So assume that all arcs in H'_{uv} are from u to v (the other case is symmetric).

Let $t' > 0$ be the number of arcs from u to v in H'_{uv}. Then $W = (t' - 2)w(uv)$ and $B = t'$. Here, the number $t' - 2$ comes from the fact W that represents the difference in solution weight, and two copies of the edge were already accounted for in H_{uv}. Then let D'_{uv} contain the arc uv from the forbidden pair, together with $t' - 1$ positive weight arcs from u to v. Observe that D'_{uv} satisfies condition 2.

(2) $\to$ (1): Let t' be the number of positive weight arcs from u to v in D'_{uv}, and let s' be the number of positive weight arcs from v to u in D'_{uv}.

Suppose first that D'_{uv} doesn't contain either arc from the forbidden pair. Then $W = (t' + s')w(uv)$ and $B = t' - s'$. Then let H'_{uv} be H_{uv} with t' arcs added from u to v and s' arcs added from v to u. Observe that H'_{uv} satisfies condition 1.

So now suppose that D' contains one arc from the forbidden pair; assume D' contains the arc uv (the other case is symmetric). Then $W = (t' + s' - 1)w(uv)$ and $B = t' - s' + 1$. Then let H'_{uv} be H_{uv} with t' arcs added from u to v, s' arcs added from v to u and one arc from v to u removed (note that even if $t' = s' = 0$, removing the arc from v to u still leaves an arc from u to v covering uv). Observe that H'_{uv} satisfies condition 1. □

Note that in a graph H'' with two vertices u and v, $b_{H''}(u) = -b_{H''}(v)$. Thus, in addition to implying that $b_{D'_{uv}}(u) = b_{H'_{uv}}(u) - b_{H_{uv}}(u)$, Lemma 9 also implies that $b_{D'_{uv}}(v) = b_{H'_{uv}}(v) - b_{H_{uv}}(v)$.

Using Lemma 9, we show that the combination of gadgets give us an instance of PBS which represents an instance of COMP-MCPP.

Lemma 10. *Let D be the directed multigraph derived from G and H by taking the vertex set $V(G)$ and adding the gadget D_{uv} for every arc and edge uv in G. Then there exists a solution H' to* COMP-MCPP *on G with weight less than H if and only if D has a properly balanced subgraph of negative weight.*

Proof. Suppose first that H' is a solution with weight less than $w(H)$, and let H'_{uv} be the subgraph of H' induced by $\{u, v\}$, for every edge or arc uv in G. For each edge and arc uv, let D'_{uv} be the subgraph of D_{uv} corresponding to H'_{uv} in Lemma 9 (which exists as $w(H'_{uv}) \geq 1$ for each uv, which in turn implies $w(H'_{uv}) < (m + 1)w(uv)$ for each uv). Let D' be the union of all such D'_{uv}. As each D'_{uv} respects double arcs and forbidden pairs, so does D'. By Lemma 9, the weight of D' is $\sum_{uv} w(D'_{uv}) = \sum_{uv}(w(H'_{uv}) - w(H_{uv})) = w(H') - w(H) < 0$ (where the sums are taken over all edges and arcs uv). Finally, for each vertex u, the imbalance of D' at u is $\sum_{a \in A(u)} b_{D'_a}(u) = \sum_{a \in A(u)}(b_{H'_a}(u) - b_{H_a}(u)) = \sum_{a \in A(u)} b_{H'_a}(u) - \sum_{a \in A(u)} b_{H_a}(u) = 0 - 0 = 0$ (where $A(u)$ is the set of all edges or arcs containing u). Thus, D' is balanced. It follows that D' is a properly balanced subgraph of D of negative weight, as required.

Conversely, suppose that D' is a properly balanced subgraph of D of negative weight, and let D'_{uv} be the subgraph of D' induced by $\{u, v\}$, for every edge or arc uv in G. For each edge and arc uv, let H'_{uv} be the graph corresponding to D'_{uv} in Lemma 9. (To see that this exists, observe

by inspection of D_{uv} that either $w(D'_{uv})$ has weight at most $(m+1)w(uv)-w(H_{uv})$, or we could replace D' with a properly balanced subgraph of smaller weight by removing two positive weight arcs, or by replacing a positive weight arc with a negative weight arc in D'_{uv}. Thus by taking D' to be the minimum weight properly balanced subgraph of D, we may assume that $w(D'_{uv})+w(H_{uv}) \leq (m+1)w(uv)$.) Let H' be the union of all such H'_{uv}. As each H'_{uv} covers uv, H' covers all edges and arcs of G. By Lemma 9, the weight of H' is $\sum_{uv} w(H'_{uv}) = \sum_{uv}(w(H_{uv})+w(D'_{uv})) = w(H)+w(D') < w(H)$ (where the sums are taken over all edges and arcs uv). Finally, for each vertex u, the imbalance of H' at u is $\sum_{a\in A(u)} b_{H'_a(u)} = \sum_{a\in A(u)}(b_{H_a}(u)+b_{D'_a}(u)) = \sum_{a\in A(u)} b_{H_a}(u) + \sum_{a\in A(u)} b_{D'_a}(u) = 0+0 = 0$ (where $A(u)$ is the set of all edges or arcs containing u). Thus, H' is balanced. It follows that H' is a solution with weight less than $w(H)$, as required. □

We note that as each gadget D_{uv} only has arcs between u and v, the graph D in Lemma 10 has the same treedepth as G. Therefore Lemma 10 implies that we have a parameterized reduction from COMP-MCPP parameterized by treedepth to PBS parameterized by treedepth. Then by Theorem 4, together with the initial solution given by Lemma 7, we have the following:

Theorem 5. MCPP *with all weights equal to 1 is FPT with respect to treedepth.*

7 Handling Arbitrary Integer Weights

We now discuss how to solve MCPP for arbitrary integer weights in FPT time. The strategy of the algorithm will be the same as in Sections 5 and 6, however, we need some adjustments, as the number of calls to COMP-MCPP is no longer trivially polynomially bounded in the input length.

We begin by showing the necessary tractability of weighted PBS.

Lemma 11. *Let $I = (D, w, X, Y)$ be an instance of* PBS*, such that D has m arcs and treedepth at most k. Assume that D has a properly balanced subgraph D^* of weight $W < 0$. In FPT time, we can find a properly balanced subgraph D' of D of weight at most W/m.*

Proof. By Lemma 5, D^* decomposes into a union of pairwise arc-disjoint properly balanced subgraphs D_i of D, where each subgraph D_i contains at most $f(k)$ arcs, for some $f(k)$. Clearly, the number of such subgraphs is at most the number of arcs in D^*, which is at most m. We therefore find that at least one such subgraph D_i has weight at most $W/m < 0$. By Lemma 6, we can then find a properly balanced subgraph with weight at most that of D_i in FPT time. □

Theorem 6. MCPP *with arbitrary integer weights is FPT with respect to treedepth, with a running time of $O(f(k) \cdot poly(n+m+\ell))$ for some $f(k)$, where $\ell = \max_{x\in A\cup E}\lceil \log_2 w(x)\rceil$.*

Proof. We will show first that the reduction from COMP-MCPP to PBS still applies to the weighted versions of the problems, then that the number of calls to COMP-MCPP is bounded. Let $\ell = \max_{x\in A\cup E}\lceil \log_2 w(x)\rceil$.

For the first part, we will reuse the gadget reduction given in Section 6. By Lemma 7 we can produce an initial solution, and by Lemma 8 we only need to consider improvement steps that produce an MCPP solution with multiplicity bounded by $m+1$. We use the gadgets D_{uv} for every pair of edge or arc uv in G, as in Section 6. This results in a weighted PBS instance where the weight of a solution is identical to the change in the weight of the MCPP solution produced by applying the improvement.

For the second part, let $W = w(H^*)-w(H)$, where H^* is an optimal solution to the MCPP instance. By Lemma 11, in FPT time we can find a new solution H_1, whose improvement

over H is of value at least W/m. Observe that $w(H_1) - w(H^*) \leq (1 - 1/m)(w(H) - w(H^*))$. Let us repeat the process of computing an improvement m times, and let H_i be the improvement derived after i iterations. Then by an inductive proof, $w(H_{i+1}) - w(H^*) \leq (1 - 1/m)(w(H_i) - w(H^*)) \leq (1 - 1/m)^{i+1}(w(H) - w(H^*))$. Thus after m iterations we produce an MCPP solution H' where $w(H') - w(H^*) \leq (1 - 1/m)^m(w(H) - w(H^*)) \leq (1 - 1/e)(w(H) - w(H^*))$ (note that this still takes FPT time). Hence, in FPT time the gap between the current solution and the optimal solution is reduced by a constant factor less than 1/2. Since the gap is an integer bounded by $2^\ell(m^2 + m)$, repeating the process $O(\ell + \log m)$ times leads to a solution H' where $w(H') - w(H) < 1$, hence H' will be an optimum. □

8 Discussion

We proved that MCPP parameterized by pathwidth is W[1]-hard, even if all edges and arcs of the input graph G have weight 1. This solves the second open question of van Bevern *et al.* [2] on parameterizations of MCPP; the first being the parameterization by the number of arcs in G, which was settled in [16].

We also showed that MCPP is FPT with respect to treedepth. Our algorithm uses a local search-style approach, where we start from a suboptimal MCPP solution H, and then repeatedly in FPT time find an alternative solution H' of lower cost than H (or conclude that H is an optimal solution). The crucial component that makes the approach work is a structural result that implies that it suffices to look for improved solutions H' such that the total difference (in a certain sense) between H and H' is bounded as a function of the treedepth. This allows us to construct an FPT algorithm using standard dynamic programming techniques.

This is the first problem we are aware of that has been shown to be W[1]-hard with respect to treewidth but FPT with respect to treedepth. Note that the pathwidth of a graph lies between its treewidth and treedepth. Open problems include pinning down the tractability border for the problem more precisely, or to find parameterizations that allow for more practical FPT algorithms. Some candidate parameters are *distance to linear forest*, which is weaker than pathwidth, and the *feedback vertex set* number.

Another parameterization of MCPP in [2] is as follows. Call a vertex v of G *even* if the total number of arcs and edges incident to v is even. Motivated by the fact that if each vertex of G is even, then MCPP is polynomial-time solvable [10], van Bevern *et al.* [2] ask whether MCPP parameterized by the number of non-even vertices is FPT. Here, even membership in XP is open.

To the best of own knowledge, kernelization for MCPP is an unexplored area of research. For MCPP, van Bevern *et al.* [2] do not mention any results on polynomial kernels or lower bounds; they provide such results for directed and undirected versions of the Rural Postman Problem, which generalize DCPP and UCPP, respectively.

Acknowledgment. We are very grateful to the referees for several very helpful suggestions. Research of GG was partially supported by Royal Society Wolfson Research Merit Award.

References

[1] E. J. Beltrami and L. D. Bodin. Networks and vehicle routing for municipal waste collection. *Networks*, 4(1):65–94, 1974.

[2] R. van Bevern, R. Niedermeier, M. Sorge, and M. Weller. *Complexity of Arc Routing Problems.* In *Arc Routing: Problems, Methods and Applications* (A. Corberán and G. Laporte, eds.), SIAM, 2014.

[3] H. L. Bodlaender, J. R. Gilbert, H. Hafsteinsson, and T. Kloks. Approximating treewidth, pathwidth, frontsize, and shortest elimination tree. *Journal of Algorithms*, 18(2):238–255, 1995.

[4] Bodlaender, H.L. A linear-time algorithm for finding tree-decompositions of small treewidth. SIAM J. Computing 25(6), 1305–1317 (1996)

[5] P. Brucker. The Chinese postman problem for mixed graphs. *Proc. WG 1980*, LNCS 100:354–366, 1981.

[6] N. Christofides. The optimum traversal of a graph. *Omega*, 1(6):719 – 732, 1973.

[7] M. Cygan, F.V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. Parameterized Algorithms, Springer 2015.

[8] M. Dom, D. Lokshtanov, S. Saurabh, and Y. Villanger. Capacitated domination and covering: A parameterized perspective. In *Proc. IWPEC 2008*, 78–91, 2008.

[9] R. G. Downey and M. R. Fellows. *Fundamentals of Parameterized Complexity.* Springer, 2013.

[10] J. Edmonds and E. L. Johnson. Matching, Euler tours and the Chinese postman. *Mathematical Programming*, 5(1):88–124, 1973.

[11] H. A. Eiselt, M. Gendreau, and G. Laporte. Arc routing problems. I. The Chinese postman problem. *Operations Research*, 43:231–242, 1995.

[12] M. Fellows, F. Fomin, D. Lokshtanov, F. Rosamond, S. Saurabh, S. Szeider, and C. Thomassen. On the complexity of some colorful problems parameterized by treewidth. *Inf. Comput.* (IANDC) 209(2):143–153, 2011.

[13] C. G. Fernandes, O. Lee, and Y. Wakabayashi. Minimum cycle cover and Chinese postman problems on mixed graphs with bounded tree-width. *Discrete Appl. Math.*, 157(2):272–279, 2009.

[14] J. Flum and M. Grohe. *Parameterized Complexity Theory.* Springer, 2006.

[15] P. A. Golovach, J. Kratochvil, and O. Suchy. Parameterized complexity of generalized domination problems. *Discrete Appl. Math.*, 160(6):780–792, 2012.

[16] G. Gutin, M. Jones, and B. Sheng. Parameterized complexity of the k-arc Chinese postman problem. In *ESA 2014*, Lect. Notes Comput. Sci. 8737:530–541, 2014.

[17] G. Gutin, M. Jones, and M. Wahlström. Structural Parameterizations of the Mixed Chinese Postman Problem. In *ESA 2015*, Lect. Notes Comput. Sci. 9294: 668–679, 2015.

[18] T. Kloks, Treewidth: Computations and Approximations. LNCS, vol 842, Springer, Heidelber (1994)

[19] C. H. Papadimitriou. On the complexity of edge traversing. *J. ACM*, 23:544–554, 1976.

[20] Y. Peng. Approximation algorithms for some postman problems over mixed graphs. *Chinese J. Oper. Res.*, 8:76–80, 1989.

A PBS Parameterized by Pathwidth and Solution Size

In this section, we will prove the following lemma:

Lemma 6. *Given an instance $(D = (V, A), w, X, Y)$ of* PBS *with integer weights, a path decomposition of the underlying undirected graph of D with width k, and an integer l, we can find a properly balanced subgraph of D of weight at most W in time $O^*(2^{k^2+3k}(2l+1)^{k+1})$, where W is the minimum weight of a properly balanced subgraph of D with at most l arcs.*

Proof. In what follows, let $[-l, l]$ denote the set of all integers z such that $-l \leq z \leq l$.

Recall that an instance of PBS is *semi-simple* if replacing every double arc and every forbidden pair by an edge yields a simple mixed graph (i.e., for every pair of vertices u, v, there is either one double arc, one forbidden pair, a single arc, or neither). We refer to an arc a in D that is not in a double arc or forbidden pair as a *standard arc*.

If $(D = (V, A), w, X, Y)$ is not semi-simple, we may construct an equivalent semi-simple instance as follows. For each standard arc $a = uv$ in D, introduce a new vertex x_a, and replace a with a pair of arcs ux_a, x_av, where $w(ux_a) = w(x_av) = w(uv)$. For a double arc $\{a, a'\}$ from u to v, introduce a new vertex $x_{\{a,a'\}}$, and replace $\{a, a'\}$ with two double arcs $\{a_1, a'_1\}, \{a_2, a'_2\}$, where a_1, a'_1 are arcs from u to $x_{\{a,a'\}}$, a_2, a'_2 are arcs from $x_{\{a,a'\}}$ to v, and $w(a_1) = w(a_2) = w(a)$, $w(a'_1) = w(a'_2) = w(a')$. Finally for a forbidden pair $\{b, b'\}$, where b is an arc uv and b' is an arc vu, introduce a new vertex $x_{\{b,b'\}}$, and replace $\{b, b'\}$ with two forbidden pairs $\{b_1, b'_1, \}, \{b_2, b'_2\}$, where b_1 is the arc $ux_{\{b,b'\}}$, b'_1 is the arc $x_{\{b,b'\}}u$, b_2 is the arc $x_{\{b,b'\}}v$, b'_2 is the arc $vx_{\{b,b'\}}$, and $w(b_1) = w(b_2) = w(b)$, $w(b'_1) = w(b'_2) = w(b')$.

Let D' be the resulting graph, and observe that the pathwidth of D' is at most the pathwidth of D plus 1. Indeed, consider a minimum width path decomposition of D. For each new vertex x that was added with neighbors u, v, choose a bag in the path decomposition D that contains both u and v, choosing a bag of smallest size if there is a choice of bags. Now add a bag immediately after this bag that is identical, except that it also contains x. The resulting decomposition is a path decomposition of D'. By construction and the fact that no new vertices are adjacent to each other, the width of this decomposition is at most $\mathrm{pw}(D) + 1$. It is also easy to verify that D has a properly balanced subgraph of weight W with l arcs if and only if D' has a properly balanced subgraph of weight $2W$ with $2l$ arcs. Thus, we may now assume that our instance $(D = (V, A), w, X, Y)$ is semi-simple.

Following [18], we may assume that we have a "nice" path decomposition with at most $4n$ nodes. In particular, we may assume we are given a series $(V_1, \ldots, V_r)$ of subsets of V, such that

- $r \leq 4n$;
- $|V_i| \leq k + 1$ for all $i \in [r]$;
- $\bigcup_{i=1}^{r} V_i = V$;
- For every arc $uv \in A$, there exists $i \in [r]$ such that $\{u, v\} \subseteq V_i$;
- For every $v \in V$, there exist $i, j \in [r]$, $i \leq j$ such that $v \in V_{i'}$ if and only if $i \leq i' \leq j$;
- For every $i \in [r-1]$, $|V_i \setminus V_{i+1}| + |V_{i+1} \setminus V_i| = 1$.

For $i > 1$, we call V_i an *Introduce bag* if $|V_i \setminus V_{i-1}| = 1$, and we call V_i a *Forget bag* if $|V_{i-1} \setminus V_i| = 1$. In other words, an Introduce bag adds a new vertex to its predecessor, and a Forget bag removes a vertex from its predecessor. Observe that by the above condition, V_i is either an Introduce bag or a Forget bag, for each $i > 1$. Let $\mathcal{V}_i = \bigcup_{j=1}^{i} V_j$. Thus in particular $\mathcal{V}_r = V$.

We now prove the claim using standard dynamic programming techniques. Informally, we will define a value $\phi(i, H, b)$, where i is the index of some bag V_i in the path decomposition, H is a subgraph of $D[V_i]$, and b is an *imbalance function* mapping vertices in V_i to integers. The value of ϕ will, roughly speaking, tell us the minimum weight of a subgraph D' of $\mathcal{V}_i$ satisfying a certain structural property, that agrees with H on $D[V_i]$, and is properly balanced except for the fact that the imbalance of each vertex $v \in V_i$ is equal to $b(v)$. The structural property is that the imbalance of every vertex is in $[-l, l]$, not just for D' but for the restriction of D' to $\mathcal{V}_j$ for every $j \leq i$. We note that every subgraph with at most l arcs satisfies this property. So by letting $i = r$ and b be the function that maps every vertex to 0, $\phi(i, H, b)$ gives us the minimum weight of a properly balanced subgraph of weight at most W, for some choice of H. Given the structual property, we may assume that the imbalance function b maps each vertex to an integer in $[-l, l]$. Thus, the number of values $\phi(i, H, b)$ that need to be computed will be restricted to a function of k and l. We will compute values $\phi(i, H, b)$ using values of the form $\phi(i-1, H', b')$, for $i > 1$. As each value can be computed in FPT time, we will have that we can compute all relevant values of $\phi(i, H, b)$ in FPT time, and thus find the weight of a properly balanced subgraph with weight at most W in FPT time.

For the actual algorithm, we let $\phi(i, H, b)$ be defined for any subgraph H of D and any function b from V to $\mathbb{Z}$. However, we will define $\phi(i, H, b)$ to be ∞ unless H is a subgraph of $D[V_i]$ and $b(v) = 0$ for any $v \notin V_i$. This allows us to simplify the description of parts of our algorithm.

We now give the precise definition of $\phi(i, H, b)$ (giving an explanation for the definition immediately afterwards). Given an integer $i \in [r]$, a subgraph H of D, and an *imbalance function* $b : V \to \mathbb{Z}$, we say that the tuple (i, H, b) is *well-formed* if the following conditions hold.

1. H is a subgraph of $D[V_i]$ (that is, H contains no arcs except those within $D[V_i]$);
2. H respects double arcs;
3. H respects forbidden pairs;
4. $b(u) = 0$ for all $u \in V \setminus V_i$; and
5. $b(u) \in [-l, l]$ for all $u \in V_i$.

Furthermore, for a well-formed tuple (i, H, b), we say that a subgraph D' of D *conforms with* (i, H, b) if the following conditions hold.

1′. D' is a subgraph of $D[\mathcal{V}_i]$;

2′. $D'[V_i] = H$;

3′. for each $u \in V$, the imbalance $d^+_{D'}(u) - d^-_{D'}(u)$ of u in D' is exactly $b(u)$;

4′. D' respects double arcs;

5′. D' respects forbidden pairs;

6′. for each $u \in V$ and each $j \leq i$, the imbalance of u in $D'[\mathcal{V}_j]$ is contained in $[-l, l]$;

We now define $\phi(i, H, b)$ to be the minimum weight of a subgraph D' of D that conforms with (i, H, b), if (i, H, b) is well-formed and if such a subgraph exists; otherwise we let $\phi(i, H, b) = \infty$.

Conditions 1–5 are conditions on H and b that can be checked in polynomial time, and without which there can be no D' that conforms with (i, H, b). Condition 5 in particular also

limits the number of values of b for which we need to perform a calculation. Conditions $1'$–$3'$ ensure that D' corresponds properly with the choice of (i, H, b). Conditions $4'$ and $5'$ enforce part of the requirements for a properly balanced subgraph (the other requirement, that of being a balanced graph, is enforced using Conditions 4, $3'$ and a suitable choice of b). Condition $6'$ is a condition added to reduce the search space of our algorithm. (We show in the next paragraph that we may assume there is a subgraph satisfying this condition.)

Let $b^* : V \to \mathbb{Z}$ be the function mapping every vertex to 0. We note for any properly balanced subgraph D' of D with at most l arcs, Conditions 1–5 hold for $(r, D'[V_r], b^*)$ and D' satisfies Conditions $1'$–$6'$. Thus $\phi(r, D'[V_r], b^*) \leq w(D')$ for any such D', and in particular $\phi(r, H, b^*) \leq W$ (the minimum weight of a properly balanced subgraph with at most l arcs) for some subgraph H of $D[V_r]$. We next show how to compute $\phi(r, H, b^*)$ for each H. At the end of the proof we show how to find a properly balanced subgraph D' of D with weight $\phi(r, H, b^*)$, from which the claim follows.

For each $i \in [r]$ in turn, each subgraph H of $D[V_i]$ and each function $b : V \to [-l, l]$ where $b(u) = 0$ for all $u \in V \setminus V_i$, we compute $\phi(i, H, b)$ in the following way:

If $i = 1$: If $(1, H, b)$ is not well-formed, or if $(1, H, b)$ is well-formed but H does not conform with $(1, H, b)$, set $\phi(1, H, b) = \infty$. Otherwise, set $\phi(1, H, b) = w(H)$.

Proof of correctness. By definition $\phi(i, H, b) = \infty$ if $(1, H, b)$ is not well-formed. Otherwise, we claim that D' conforms with $(1, H, b)$ only if $D' = H$. Indeed, as D' satisfies Conditions $1'$ and $2'$ with respect to $(1, H, b)$, we have that $D' = D'[\mathcal{V}_1] = D'[V_1] = H$. Thus, to compute $\phi(1, H, b)$ it is enough to check that H satisfies Conditions $1'$–$6'$ with respect to $(1, H, b)$. □

We now consider the cases when $i > 1$. We may now assume that we have correctly computed $\phi(i-1, H', b')$ for any H', b' such that H' is a subgraph of $D[V_{i-1}]$ and $b' : V \to [-l, l]$ has $b'(u) = 0$ for all $u \in V \setminus V_{i-1}$. For any H', b' not satisfying these conditions, we know by Conditions 1–5 that $\phi(i-1, H', b') = \infty$ and so we assume this in the algorithm.

If V_i is an Introduce bag:

Recall that $|V_i \setminus V_{i-1}| = 1$ (and therefore $|V_{i-1} \setminus V_i| = 0$), i.e., there exists a single vertex $v \in V_i$ such that $v \notin V_{i-1}$ and $V_i = V_{i-1} \cup \{v\}$.

Assume that (i, H, b) is well-formed; otherwise set $\phi(i, H, b) = \infty$. Define H' to be the graph $H[V_{i-1}]$, and define $b' : V \to \mathbb{Z}$ by setting $b'(u) = b(u) - (d^+_H(u) - d^-_H(u)) + d^+_{H'}(u) - d^-_{H'}(u)$ for each $u \in V$ (taking $d^+_{H''}(u) = d^-_{H''}(u)$ to be 0 whenever u is not in the subgraph H''). Set $\phi(i, H, b) = \phi(i-1, H', b') + w(H) - w(H')$.

Proof of correctness. By definition $\phi(i, H, b) = \infty$ if (i, H, b) is not well-formed. So assume that this is not the case.

We first show that $\phi(i, H, b) \leq \phi(i-1, H', b') + w(H) - w(H')$. Assume without loss of generality that $\phi(i-1, H', b') \neq \infty$, i.e. $(i-1, H', b')$ is well-formed and there exists a subgraph D'' of D that conforms with $(i-1, H', b')$. Let D'' be such a graph of minimum weight, i.e., $w(D'') = \phi(i-1, H', b')$. Now let D' be the graph derived from D'' by adding all arcs that are in H but not in H', i.e., all arcs of H that are incident to v. Observe that $w(D') = w(D'') + w(H) - w(H') = \phi(i-1, H', b') + w(H) - w(H')$. Thus it remains to show that $\phi(i, H, b) \leq w(D')$, which we will do by showing that D' conforms with (i, H, b).

As D'' is a subgraph of $D[\mathcal{V}_{i-1}]$ and the only arcs added to D'' were those within V_i, we have that D' is a subgraph of $D[\mathcal{V}_i]$, satisfying Condition $1'$. By construction of D' and the fact that $D''[V_{i-1}] = H'$, we have that $D'[V_i] = H$, satisfying Condition $2'$. For each $u \in V$, we have that the imbalance $d^+_{D'}(u) - d^-_{D'}(u) = (d^+_{D''}(u) + d^+_H(u) - d^+_{H'}(u)) - (d^-_{D''}(u) + d^-_H(u) + d^-_{H'}(u)) = b'(u) + (d^+_H(u) - d^-_H(u)) - (d^+_{H'}(u) - d^-_{H'}(u)) = b(u)$, satisfying Condition $3'$. As D'' respects double arcs and forbidden pairs, and H respects double arcs and forbidden pairs, we have

that D' respects double arcs and forbidden pairs. (If D' contains exactly one arc from a double arc, then D'' or H also contains exactly one arc from the double arc. Similarly if D' contains both arcs of a forbidden pair, then so does D'' or H). Thus, D' satisfies Conditions $4'$ and $5'$. Finally, as $D'[\mathcal{V}_j] = D''[\mathcal{V}_j]$ for $j < i$, we have that the imbalance of each vertex in $D'[\mathcal{V}_j]$ is in $[-l, l]$ for each $j < i$. Furthermore as $D'[\mathcal{V}_i] = D'$, and by Condition $3'$, the imbalance of any vertex u in $D'[\mathcal{V}_i]$ is $b(u) \in [-l, l]$. Thus D' satisfies Condition $6'$.

Thus, D' conforms with (i, H, b), and so $\phi(i, H, b) \leq w(D') = \phi(i-1, H', b') + w(H) - w(H')$.

We now show that $\phi(i, H, b) \geq \phi(i-1, H', b') + w(H) - w(H')$. Suppose without loss of generality that $\phi(i, H, b) \neq \infty$, i.e., (i, H, b) is well-formed and there exists a subgraph D' of D that conforms with (i, H, b). Let D' be such a graph of minimum weight, i.e. $w(D') = \phi(i, H, b)$. Now let $D'' = D'[\mathcal{V}_{i-1}]$, i.e. D'' is D' with vertex v deleted. Observe that $w(D'') = w(D') - w(H) + w(H') = \phi(i, H, b) - w(H) + w(H')$. Thus it remains to show that $\phi(i-1, H', b') \leq w(D'')$, which we will do by showing that $(i-1, H', b')$ is well-formed and that D'' conforms with $(i-1, H', b')$.

By construction, H' is a subgraph of $D[V_{i-1}]$ and so Condition 1 holds. As H' is an induced subgraph of H which is an induced subgraph of D', and D' respects double arcs and forbidden pairs, it follows that H' also respects double arcs and forbidden pairs, satisfying Conditions 2 and 3. For any $u \in V \setminus V_{i-1}$, if $u \neq v$ then $b'(u) = b(u) = 0$. As the only arcs of D' incident to v are in H, and none of them are in H', the imbalance of v in D' is exactly $d^+_H(v) - d^-_H(v)$, and this is also the value of $b(v)$. Thus $b'(v) = b(v) - (d^+_H(v) - d^-_H(v)) + d^+_{H'}(v) - d^-_{H'}(v) = (d^+_H(v) - d^-_H(v)) - (d^+_H(v) - d^-_H(v)) + 0 = 0$. Thus we have that Condition 4 holds. For $u \in V_{i-1}$, as $b(u)$ is the imbalance of u in D' and by construction of D'' and b', we have that $b'(u) = d^+_{D'}(u) - d^-_{D'}(u) - (d^+_H(u) - d^-_H(u)) + d^+_{H'}(u) - d^-_{H'}(u)$ which is the imbalance of u in D''. As $D'' = D'[\mathcal{V}_{i-1}]$, and as the imbalance of any vertex in $D'[\mathcal{V}_j]$ is in $[-l, l]$ for each $j \leq i$, we have that $b'(u) \in [-l, l]$, and so Condition 5 holds. We thus have that $(i-1, H', b')$ is well-formed.

Next we show that D'' conforms with $(i-1, H', b')$. By construction, D'' is a subgraph of $D[\mathcal{V}_{i-1}]$, satisfying Condition $1'$. By construction of D'' and H' and the fact that $D'[V_i] = H$, we have $D''[V_{i-1}] = H'$, satisfying Condition $2'$. For any $u \in V$, the arcs incident to u in D'' are exactly the same as in D', except for those arcs which are in H and not H'. Therefore the imbalance is $d^+_{D''}(u) - d^-_{D''}(u) = (d^+_{D'}(u) - d^+_H(u) + d^+_{H'}(u)) - (d^-_{D'}(u) - d^-_H(u) + d^-_{H'}(u)) = b(u) - (d^+_H(u) - d^-_H(u)) + (d^+_{H'}(u) - d^-_{H'}(u)) = b'(u)$, satisfying Condition $3'$. As D'' is an induced subgraph of D' and D' respects double arcs and forbidden pairs, D'' also respects double arcs and forbidden pairs, satisfying Conditions $4'$ and $5'$. Finally, for each $u \in V$ and each $j \leq i-1$, $D''[\mathcal{V}_j] = D'[\mathcal{V}_j]$, and so the imbalance of u in $D''[\mathcal{V}_j]$ is contained in $[-l, l]$, satisfying Condition $6'$.

Thus, D'' conforms with $(i-1, H', b')$, and so $\phi(i-1, H', b') \leq w(D'')$, and therefore $\phi(i, H, b) = w(D') = w(D'') + w(h) - w(H') \geq \phi(i-1, H', b') + w(h) - w(H')$. We therefore have that $\phi(i, H, b) = \phi(i-1, H', b') + w(h) - w(H')$ and so the algorithm is correct. □

If V_i is a Forget bag:

Recall that $|V_{i-1} \setminus V_i| = 1$ (and therefore $|V_i \setminus V_{i-1}| = 0$), i.e., there exists a single vertex $v \in V_{i-1}$ such that $v \notin V_i$ and $V_{i-1} = V_i \cup \{v\}$.

If (i, H, b) is not well-formed, set $\phi(i, H, b) = \infty$. Otherwise, set $\phi(i, H, b)$ to be the minimum value of $\phi(i-1, H', b)$ over all subgraphs H' of $D[V_{i-1}]$ such that $H'[V_i] = H$ and $(i-1, H', b)$ is well-formed. If no such H' exists, then set $\phi(i, H, b) = \infty$.

Proof of correctness. By definition $\phi(i, H, b) = \infty$ if (i, H, b) is not well-formed. So assume that this is not the case.

We first show that $\phi(i, H, b) \leq \phi(i-1, H', b)$ for any choice of H' such that $H'[V_i] = H$ and $(i-1, H', b)$ is well-formed. Assume without loss of generality that $\phi(i-1, H', b) \neq \infty$ i.e. there exists a graph D' that conforms with $(i-1, H', b)$. Let D' be such a graph of minimum weight, i.e. $w(D') = \phi(i-1, H', b)$. We now show that D' also conforms with (i, H, b), which implies that $\phi(i, H, b) \leq w(D') = \phi(i-1, H', b)$.

As $V_i \subset V_{i-1}$, we have that $\mathcal{V}_i = \mathcal{V}_{i-1}$. Therefore as D' is a subgraph of $D'[\mathcal{V}_{i-1}]$ it is also a subgraph of $D'[\mathcal{V}_i]$, satisfying Condition 1′. As $D'[V_{i-1}] = H'$, $H'[V_i] = H$ and $V_i \subset V_{i-1}$, we have that $D'[V_i] = H'[V_i] = H$, satisfying Condition 2′. As Conditions 3′, 4′, and 5′ only concern D' and b, and D' satisfies these Conditions with respect to $(i-1, H', b)$, D' trivially satisfies these conditions with respect to (i, H, b). Finally as the imbalance of each vertex u in $D'[\mathcal{V}_j]$ is in $[-l, l]$ for each $j \leq i-1$, and $\mathcal{V}_i = \mathcal{V}_{i-1}$, we have that D' satisfied Condition 6′ with respect to (i, H, b).

Thus we have that $\phi(i, H, b) \leq w(D') = \phi(i-1, H', b)$ for each choice of H' such that $H'[V_i] = H$ and $(i-1, H', b)$ is well-formed, and in particular for the choice of H' that minimizes $\phi(i-1, H', b)$.

We now show that if $\phi(i, H, b) \neq \infty$, then $\phi(i, H, b) \geq \phi(i-1, H', b)$ for some choice of H' such that $H'[V_i] = H$ and $(i-1, H', b)$ is well-formed. Suppose that $\phi(i, H, b) \neq \infty$, and let D' be a subgraph of D that conforms with (i, H, b) such that $w(D') = \phi(i, H, b)$. Let $H' = D'[V_{i-1}]$, and note that $H'[V_i] = H$. We now show that $(i-1, H', b)$ is well-formed and that D' conforms with $(i-1, H', b)$, which implies that $\phi(i, H, b) = w(D') \geq \phi(i-1, H', b)$.

By construction H' is a subgraph of $D[V_{i-1}]$, satisfying condition 1. As H' is an induced subgraph of D' and D' respects double arcs and forbidden pairs, H' also respects double arcs and forbidden pairs, satisfying Conditions 2 and 3. As $V \setminus V_{i-1} \subset V \setminus V_i$ and $b(u) = 0$ for all $u \in V \setminus V_i$, Condition 4 holds for $(i-1, H', b)$. As $b(u) \in [-l, l]$ for all $u \in V_i$ and $b(v) = 0$ (since Condition 4 holds for (i, H, b)), we have that $b(u) \in [-l, l]$ for all $u \in V_{i-1}$, satisfying Condition 5. We thus have that $(i-1, H', b)$ is well-formed.

It remains to show that D' conforms with $(i-1, H', b)$. As D' is a subgraph of $D[\mathcal{V}_i]$ and $\mathcal{V}_{i-1} = \mathcal{V}_i$, D' satisfies Condition 1′. By construction of H', $D'[V_{i-1}] = H'$ satisfying Condition 2′. As Conditions 3′, 4′, and 5′ only concern D' and b, and D' satisfies these conditions with respect to (i, H, b), D' trivially satisfies these conditions with respect to $(i-1, H', b)$. Finally, 6′ for $(i-1, H', b)$ is a special case of the same condition for (i, H, b), and so D' satisfies this condition as well.

Thus, D' conforms with $(i-1, H', b)$, and so $\phi(i, H, b) = w(D') \geq \phi(i-1, H', b)$. We therefore have that $\phi(i, H, b) \leq \phi(i-1, H', b)$ for every H' such that $H'[V_i] = H$ and (i, H', b) is well-formed, and that for some such H', $\phi(i, H, b) = w(D') \geq \phi(i-1, H', b)$. This implies that $\phi(i, H, b) = \phi(i-1, H', b)$ for the choice of H' that minimizes $\phi(i-1, H', b)$, and so the algorithm is correct. □

This concludes the description of the algorithm to compute $\phi(i, H, b)$ and the proof of its correctness. We now analyze the running time.

When $i = 1$ or V_i is an Introduce bag, the time taken to compute $\phi(i, H, b)$ is clearly polynomial. When V_i is a Forget bag, we need to check the value of $\phi(i-1, H', b)$ for every subgraph H' of D such that $H'[V_i] = H$ and $(i-1, H', b)$ is well-formed. As such an H' must be a subgraph of $D[V_{i-1}]$, the only arcs that can be in H' but not H are those within V_{i-1} incident to v. As D is semi-simple, there are at most $2|V_{i-1} \setminus \{v\}| \leq 2k$ such arcs. Thus there are at most 2^{2k} possible graphs H' to consider. Thus $\phi(i, H, b)$ can be computed in time $O^*(2^{2k})$.

The number of values $\phi(i, H, b)$ to compute is bounded by the the number of tuples (i, H, b) where $i \in [r]$, H is a subgraph of $D[V_i]$, and $b : V \to [-l, l]$ is a function that maps u to 0 for any $u \in V \setminus V_i$. As D is semi-simple and $|V_i| \leq k+1$, the number of arcs in $D[V_i]$ is at most $k(k+1)$ and so the number of possible H is at most $2^{k(k+1)}$ for each i. The number

of possible b is $(2l+1)^{|V_i|} \leq (2l+1)^{k+1}$ for each i. Thus the number of values $\phi(i, H, b)$ to compute is at most $r2^{k(k+1)}(2l+1)^{k+1} \leq 4n2^{k(k+1)}(2l+1)^{k+1} = O^*(2^{k^2+k}(2l+1)^{k+1})$. As each value $\phi(i, H, b)$ can be computed in time $O^*(2^{2k})$, the time taken to compute all values is $O^*(2^{k^2+k}(2l+1)^{k+1}2^{2k}) = O^*(2^{k^2+3k}(2l+1)^{k+1})$.

Recall that $b^* : V \to \mathbb{Z}$ is the function mapping every vertex to 0. We now show how to make the algorithm constructive, i.e. how to find a subgraph D' with weight $\phi(r, H, b^*)$ that conforms with (r, H, b^*), for any subgraph H of $D[V_r]$ such that $\phi(r, H, b^*) \neq \infty$.

Define $H_r = H$ and $b_r = b^*$. Then given graph H_i and function b_i for $i \in [r]$ with $i > 1$, we recursively define H_{i-1} and b_{i-1} as follows. If V_i is an Introduce bag, then let H_{i-1} be the graph H' and let b_i be the function b' as constructed by the algorithm for $\phi(i, H_i, b_i)$. If V_i a Forget bag, then let H_{i-1} be the subgraph H' of $D[V_{i-1}]$ that has $H'[V_i] = H_i$ and minimized $\phi(i-1, H', b_i)$, as described in the algorithm for $\phi(i, H_i, b_i)$, and let $b_{i-1} = b_i$. Now for each $i \in [r]$, let D_i be the unique subgraph of $D[\mathcal{V}_i]$ such that $D_i[V_j] = H_j$ for every $j \leq i$. Let $D' = D_r$. Now observe that $\phi(i, H_i, b_i) = w(D_i)$ for each $i \in [r]$, and so in particular $\phi(r, H, b^*) = w(D')$. Observe also that D' conforms with (i, H_i, b^*). Thus D' is a properly balanced subgraph with weight $\phi(i, H, b^*) \leq W$, as required. $\square$