A New View on Rural Postman Based on Eulerian Extension and Matching[☆]

Manuel Sorge[1], René van Bevern[2], Rolf Niedermeier, Mathias Weller[3]

Institut für Softwaretechnik und Theoretische Informatik, TU Berlin

Abstract

We provide a new characterization of the NP-hard arc routing problem Rural Postman in terms of a constrained variant of minimum-weight perfect matching on bipartite graphs. To this end, we employ a parameterized equivalence between Rural Postman and Eulerian Extension, a natural arc addition problem in directed multigraphs. We indicate the NP-hardness of the introduced matching problem. In particular, we use the matching problem to make partial progress towards answering the open question about the parameterized complexity of Rural Postman with respect to the parameter "number of weakly connected components in the graph induced by the required arcs". This is a more than thirty years open and long-neglected question with significant practical relevance.

Keywords: Arc Routing, Arc Addition, Chinese Postman, Parameterized Complexity

1. Introduction

The Rural Postman (RP) problem [13, 28] with its special case, the Chinese Postman problem [26], is a famous arc routing problem in combinatorial optimization. Given a directed, arc-weighted graph G and a subset R of its arcs (called "required arcs"), the task is to find a minimum-cost closed walk in G that visits all arcs of R. The practical applications of RP include snow plowing, garbage collection, and mail delivery [1, 3, 5, 12, 14, 30]. Recently, it has been observed that RP is closely related (more precisely, "parameterized equivalent") to the arc addition problem Eulerian Extension (EE) [10].

[☆]This work is based on the Diploma thesis of one of the authors [33]. A preliminary version of this work has been presented at the 22nd International Workshop on Combinatorial Algorithms (IWOCA '11), Victoria, Canada, June 2011 [35]. We also give full versions of some results which have been presented at the 37th International Workshop on Graph-Theoretic Concepts in Computer Science (WG '11), Teplá Monastery, Czech Republic, June 2011 [34].

Email addresses: manuel.sorge@tu-berlin.de (Manuel Sorge), rene.vanbevern@tu-berlin.de (René van Bevern), rolf.niedermeier@tu-berlin.de (Rolf Niedermeier), mathias.weller@tu-berlin.de (Mathias Weller)

[1]Partially supported by the DFG, project AREG, NI 369/9 and project PABI, NI 369/7.

[2]Supported by the DFG, project AREG, NI 369/9.

[3]Supported by the DFG, project DARE, NI 369/11.

In EE, a directed multigraph G and a function assigning a weight value to each potential arc on the vertices of G is given. The task is to find a minimum-weight set of arcs to add to G such that the resulting multigraph is Eulerian. RP and EE are NP-hard [22, 23]. In fact, their mentioned parameterized equivalence means that many algorithmic and complexity-theoretic results for one of them transfer to the other. In particular, this gives a new view on RP, perhaps leading to novel approaches to attack its computational hardness.

A key issue in both problems is to determine the influence of the number c of connected components on each problem's computational complexity [10, 17, 18, 23, 29]. More precisely, c refers to the number of weakly connected components in the input graph for EE and the number of weakly connected components in the graph induced by the required arcs for RP. If $c = 1$, then RP is efficiently solvable in polynomial-time [10]. Indeed, Frederickson [17, 18] observed that, generally, RP is polynomial-time solvable when c is constant. However, c influences the degree of the polynomial in the running time of Frederickson's algorithm. To date, it is open whether this is unavoidable[4] or whether RP can be solved in $f(c) \cdot n^{O(1)}$ time for some function f. In other words, it remains open whether RP (and EE) is fixed-parameter tractable with respect to the parameter c [10]. See Section 2 and the literature [11, 15, 27] for more on parameterized complexity analysis. We remark that the parameter c is presumably small in a number of applications [10, 17, 18]. This motivates addressing this seemingly hard open question.

Related Work. The RP problem and its various variants have received much attention in the past. Subsequent to RP's introduction [13, 28] it has been shown NP-complete [23]. Heuristics and approximation algorithms have been presented [3, 17, 18, 20, 32] as well as exact exponential-time algorithms based on integer linear programs [7, 8, 19, 25]. See also overview articles by Eiselt et al. [14], by Assad and Golden [1] and the book edited by Dror [12]. There is also a number of papers that evaluate algorithms for RP in practical settings [5, 31]. However, we are not aware of studies in the realm of parameterized complexity except in the context of Eulerian extensions.

Höhn et al. [22] recently introduced a variant of EE in the context of scheduling and proved it to be NP-complete. EE has been shown to be polynomial-time solvable in some special cases [4, 10, 22, 24]. Dorn et al. [10] also proved that EE is fixed-parameter tractable with respect to the parameter "number of arcs in the sought Eulerian extension". Note that this parameter is an upper bound for c, however, it is reasonable to assume that c is much smaller in practice. Also, the parameterized complexity of a number of vertex and edge deletion problems related to Eulerian graphs has been considered recently [6, 9, 16].

Our Results. In this work, we contribute new insights concerning the seemingly hard open question whether RP (and EE) is fixed-parameter tractable with respect to the parameter "number c of components". To this end, our main contribution is a new characterization of RP in terms of a variant of minimum-weight perfect matching on (undirected) bipartite graphs: Conjoining Bipartite Matching (CBM). Here, in

[4] Under reasonable complexity-theoretic assumptions.

addition to searching a matching that matches every vertex and that is of weight at most some given maximum, further constraints are given: The vertices in the input graph are grouped and the additional constraints are of the form "between vertex group A and vertex group B, there must be at least one edge in the matching". A more formal definition is given in Section 4. We show that EE and CBM are parameterized equivalent with respect to the parameters "number of components" for EE and "number of additional constraints" for CBM.

To prove the equivalence of EE and CBM, we use a *parameterized Turing reduction*; thus, we have to separately show that CBM is still NP-hard under classical many-one reductions. As it turns out, this is the case even when the input graph has maximum degree two. We address the open question of whether EE is fixed-parameter tractable with respect to the parameter "number of weakly connected components": We obtain that CBM is fixed-parameter tractable with respect to the parameter "number of additional constraints" when restricted to bipartite graphs where one partition set has maximum vertex degree two. This implies corresponding fixed-parameter tractability results for relevant special cases of RP and EE which would perhaps have been harder to formulate and to detect using the original definitions of these problems. Indeed, we hope that CBM might help to finally answer the puzzling open question concerning the parameterized complexity of RP with respect to the number c of components.

As a side result, we also obtain a fixed-parameter algorithm for EE from one of the reductions we give. It implies that EE is fixed-parameter tractable with respect to the parameters c and "the sum b of positive balances of vertices in the input". Together, these parameters measure the problem's distance from triviality [21].

In this paper, we focus on decision problems. However, our results easily transfer to the corresponding optimization problems. Note that, for the sake of notational convenience and justified by the known parameterized equivalence [10], most of our results and proofs refer to EE instead of RP.

Structure of the Paper. This work is organized as follows. In Section 2, we provide some notation, preliminary observations and useful results. Next, the parameterized equivalence of RP and CBM is proven in two steps. First, in Section 3, variants of EE are introduced and reductions are given that are used as intermediate steps for the reductions that yield the equivalence. This also yields the above-mentioned fixed-parameter algorithm for EE with respect to the parameters b and c. Second, in Section 4, it is shown that CBM can be reduced to one of the variants of EE and another variant of EE can be reduced to CBM. This then concludes the proof of equivalence of CBM, EE, and, thus, RP. Next, in Section 5, we take a closer look at CBM. In particular, we show the fixed-parameter tractability for the mentioned special case. See Figure 1 for an overview of the reductions given in the paper. We conclude in Section 6 with directions for future research.

2. Preliminaries and Preparations

In this section, we first define our notation, then recapitulate preprocessing routines for Eulerian Extension that give useful restrictions on the instances we have to consider.

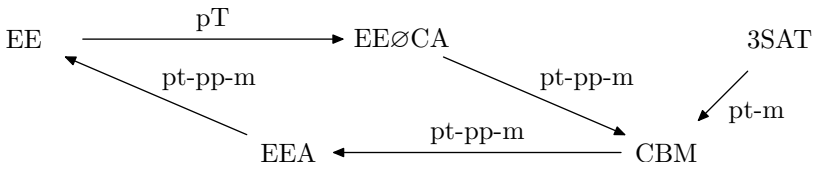


Figure 1: Schematic overview on the reductions given in this work. The label "pT" indicates a parameterized Turing reduction, the label "pt-pp-m" indicates a polynomial-time polynomial-parameter many-one reduction, and the label "pt-m" indicates a classical polynomial-time many-one reduction. EE∅CA and EEA are variants of EE which we use as intermediate problems for proving the equivalence of EE and CBM. The reductions from and to EE are covered in Section 3 (Theorem 3.1 and Theorem 3.2). The reductions between variants of EE and CBM are given in Section 4 (Theorem 4.1 and Theorem 4.2). NP-hardness of CBM is proven in Section 5 via a reduction from 3SAT (Theorem 5.1).

Finally, using the preprocessing routines, we prove a theorem about the structure of Eulerian extensions of minimum weight.

2.1. Notation and Problem Definition

We mainly consider directed multigraphs and we follow the notation of Bang-Jensen and Gutin [2]. For a directed multigraph $G = (V, A)$, we use $V(G)$ and $A(G)$ to denote the set of vertices and the multiset of arcs, respectively. For undirected graphs $H = (V, E)$, we instead use $E(H)$ to refer to the set of edges. Where it is appropriate, we use n to refer to $|V(G)|$ and m to refer to $|A(G)|$ or $|E(G)|$, respectively, for a given graph G. For a given graph $G = (V, A)$ and an arc set B, we sometimes denote the graph $(V, A \cup B)$ by $G + B$. The *underlying undirected multigraph* of a directed multigraph G is the graph obtained by removing the direction of each arc. Two vertices of a directed multigraph G are *weakly connected* if they are connected in the underlying undirected multigraph of G. A maximal subset of pairwise weakly connected vertices of G is called a *weakly connected component*. Since we never consider strongly connected components, we omit the adverb "weakly".

A *walk* w in the multigraph G is a sequence of arcs in G such that each arc ends in the same vertex as the next arc starts in. We sometimes abuse notation and use w to refer to the arc-induced graph instead, that is, to the graph defined by all arcs of w and all vertices it traverses. The first vertex in the sequence is called the *initial* vertex of the walk and the last vertex in the sequence is called the *terminal* vertex of the walk. A trail is called *closed* if its initial vertex is also its terminal vertex and *open* otherwise. A walk w in G such that $A(w)$ is a submultiset of the multiset $A(G)$ is called a *trail* of G. A trail t in G such that every vertex in G has at most two incident arcs in $A(t)$ is called a *cycle* if t is closed, and *path* otherwise. If G is clear from the context, we omit it. Undirected walks, trails, paths, and cycles are defined in the obvious way.

For a directed multigraph $G = (V, A)$ and and a vertex v, $\text{indeg}_G(v)$ denotes $|\{(u, v) \in A\}|$ and $\text{outdeg}_G(v)$ is defined analogously. We use $\text{balance}_G(v) := \text{indeg}_G(v) - \text{outdeg}_G(v)$

to denote the *balance* of a vertex v in G and I_G^+ and I_G^- to denote the set of all vertices v in G with $\text{balance}_G(v) > 0$ and $\text{balance}_G(v) < 0$, respectively. A vertex v is *balanced* in G if $\text{balance}_G(v) = 0$. When the graph is clear from the context, we omit the subscript in indeg, outdeg, and balance.

Our results are in the context of parameterized complexity [11, 15, 27]. A *parameterized problem* $L \subseteq \Sigma^* \times \mathbb{N}$ is called *fixed-parameter tractable* (*FPT*) with respect to a parameter k if $(x, k) \in L$ is decidable in $f(k) \cdot |x|^{O(1)}$ time, where f is a computable function only depending on k.

We consider two types of parameterized reductions between problems: A *polynomial-parameter polynomial-time many-one reduction* ($\leq_{\text{m}}^{\text{PPP}}$-reduction) from a parameterized problem L to a parameterized problem L' is a polynomial-time computable function g such that $(x, k) \in L \Leftrightarrow (x', k') \in L'$, with $(x', k') := g(x, k)$, and $k' \leq p(k)$, where p is a polynomial only depending on k. If such a reduction exists, then we write $L \leq_{\text{m}}^{\text{PPP}} L'$. A *parameterized Turing reduction* ($\leq_{\text{T}}^{\text{FPT}}$-reduction) from a parameterized problem L to a parameterized problem L' is an algorithm that decides $(x, k) \in L$ in $f(k) \cdot |x|^{O(1)}$ time, where queries of the form $(x', g(k)) \in L'$ are assumed to be decidable in O(1) time and f, g are functions solely depending on k. If such a reduction exists, we write $L \leq_{\text{T}}^{\text{FPT}} L'$. If $L \leq_{\text{T}}^{\text{FPT}} L'$ and $L' \leq_{\text{T}}^{\text{FPT}} L$, then we say that L and L' are $\leq_{\text{T}}^{\text{FPT}}$-equivalent. Note that every $\leq_{\text{m}}^{\text{PPP}}$-reduction is a $\leq_{\text{T}}^{\text{FPT}}$-reduction. Also, if $L' \in \text{FPT}$ and $L \leq_{\text{T}}^{\text{FPT}} L'$, then $L \in \text{FPT}$.

In this work, we consider the problem of making a given directed multigraph Eulerian by adding arcs. A directed multigraph G is *Eulerian* if it is connected and each vertex is balanced. An *Eulerian extension* E for $G = (V, A)$ is a multiset over $V \times V$ such that the directed multigraph $G + E = (V, A \cup E)$ is Eulerian.

Eulerian Extension (EE)
Input: A directed multigraph $G = (V, A)$, an integer $\omega_{\max}$, and a weight function $\omega \colon V \times V \to [0, \omega_{\max}] \cup \{\infty\}$.
Question: Is there an Eulerian extension E of G whose weight is at most $\omega_{\max}$?

In the context of EE, we speak of *allowed* arcs $a \in V \times V$ if $\omega(a) \neq \infty$.

2.2. Preprocessing Routines

In this section, we observe that in instances of Eulerian Extension (EE) we may assume that every unbalanced vertex misses at most one incoming or outgoing arc and that the weights fulfill the triangle inequality. The first observation is helpful for simplifying reductions. The second observation is crucial for restricting the structure of Eulerian extensions that we have to consider (see Section 2.3).

A polynomial-time preprocessing routine for EE introduced by Dorn et al. [10] ensures that the balance of every vertex is in $\{-1, 0, 1\}$. Given an imbalanced vertex v, the transformation adds a new, balanced vertex u that is connected to v. It then moves one arc involving v to u such that the absolute imbalance of v decreases by one. Iterating this, we can decrease the absolute imbalance of each vertex to one.

Dorn et al. [10] showed that the corresponding transformation can be computed in $O(n(n + m))$ time. In the following, we assume that all input instances of EE have been transformed in this way and, hence, we assume that the following holds.

Fact 2.1. *In a preprocessed instance of* EE, $\text{balance}(v) \in \{-1, 0, 1\}$ *for each vertex* v.

We use a second preprocessing routine to make further observations about trails in Eulerian extensions. This preprocessing is a variant of the algorithm used by Dorn et al. [10] to remove isolated vertices from the input graph. It simply replaces the weight of a vertex pair by the weight of a "lightest" path in the graph $(V, V \times V)$ with respect to ω. Note that the resulting weight function respects the triangle inequality. This transformation can be computed in $O(n^3)$ time using an all-pairs shortest path algorithm. In the following, we assume all input instances of EE to have gone through this transformation, and hence, we assume that the following holds.

Fact 2.2. *The weight-function ω of a preprocessed instance of* EE *respects the triangle inequality, that is, for any vertices x, y, z, it holds that $\omega(x, z) \leq \omega(x, y) + \omega(y, z)$.*

In the subsequent sections, we use this preprocessing in fixed-parameter algorithms and parameterized reductions. To this end, we note that both transformations are parameter-preserving, that is, they do not change the number of connected components.

The presented transformations lead to useful observations regarding trails in Eulerian extensions, see Section 2.3.

2.3. The Structure of Minimum-Weight Eulerian Extensions

To restrict the structure of solutions we are seeking, we now make some observations on optimal solutions. To conveniently state our results, we first introduce the following notation.

Definition 2.1. The *component graph* $\mathbb{C}_G$ of a directed multigraph G is a clique whose vertices one-to-one correspond to the weakly connected components of G. For a trail t in a multigraph G, $\mathbb{C}_G(t)$ is the trail in $\mathbb{C}_G$ that is obtained in the following way: Take the underlying undirected multigraph of t and, for every connected component C of G, substitute every maximal length subtrail t' of t with $V(t') \subseteq C$ by the vertex in $\mathbb{C}_G$ corresponding to C.

Based on the preprocessing routines from Section 2.2, we obtain the following theorem.

Theorem 2.1. *Let G be a directed multigraph with c connected components. Let G and the weight function $\omega : V \times V \to [0, \omega_{max}] \cup \{\infty\}$ constitute an instance of* Eulerian Extension *such that Fact 2.1 and Fact 2.2 hold. Then, there is a set $S := \{t_1, \ldots, t_k\}$ of pairwise edge-disjoint paths and cycles in the graph $(V, V \times V)$ such that*

- *(i) $\bigcup_{i=1}^{k} A(t_i)$ is an Eulerian extension of minimum weight for G,*
- *(ii) each $t_i \in S$ contains at most $c + 1$ vertices,*
- *(iii) for $t_i, t_j \in S$, both containing at least two arcs, the trails $\mathbb{C}_G(t_i)$ and $\mathbb{C}_G(t_j)$ are edge-disjoint, and*
- *(iv) the graph defined by the union of all trails $\mathbb{C}_G(t_1), \ldots, \mathbb{C}_G(t_k)$ without their initial vertices does not contain a cycle.*

Particularly the last condition helps to improve the running time of deriving a structure that helps finding Eulerian extensions—we use this in Section 3.

We now prove Theorem 2.1 successively, by giving four observations that, in concert, yield the theorem. At first, observe that it is easy to decompose Eulerian extensions into

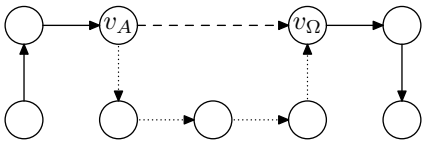


Figure 2: Example of an application of Transformation 2.1. Solid arcs and dotted arcs belong to a trail t, dotted arcs to a subtrail s of t and the dashed arc is substituted for the dotted arcs in t' by the shortcut transformation.

trails: greedily remove a maximal trail t from an Eulerian extension E and repeat.[5] Also observe that if E is an Eulerian extension for G, then $E \setminus A(t)$ is an Eulerian extension for $G + A(t)$ and, thus, it suffices to show the properties in Theorem 2.1 for maximal trails in Eulerian extensions. These properties will mainly be proven by taking such a trail in an Eulerian extension and "shortcutting" it such that the Eulerian extension still connects all components and retains the balance of every vertex. Next, we formally introduce the shortcut transformation.

Transformation 2.1. Let E be an Eulerian extension of G, let t be a trail in the multigraph $(V(G), E)$ and let s be a subtrail of t with initial vertex v_A and terminal vertex v_Ω. Obtain a new trail t' by substituting the arc (v_A, v_Ω) for s in t and derive a new arc set E' by substituting $A(t')$ for $A(t)$ in E. Define $\text{shortcut}(E, t, s) := (E', t')$.

Figure 2 illustrates Transformation 2.1. Next, we observe in which cases we can safely shortcut trails in Eulerian extensions.

Lemma 2.1. *Let* $\text{shortcut}(E, t, s) = (E', t')$ *where the trail* s *has initial vertex* v_A *and terminal vertex* v_Ω*. The following statements hold:*

(i) $\omega(E') \leq \omega(E)$.

(ii) Each vertex in $V(s)$ *is balanced in* $G + E'$.

(iii) If each vertex of s *except* v_A *and* v_Ω *is contained in a connected component of* G *that also contains a vertex of* t'*, then the arc set* E' *is an Eulerian extension for* G.

Proof. Statement (i) is trivial because of Fact 2.2.

By substituting (v_A, v_Ω) for s, both the indegree and outdegree of each vertex on s except v_A and v_Ω decreases by one. Hence, augmenting G with E' results in a graph without unbalanced vertices (statement (ii)).

For statement (iii), it remains to show that the graph $(V(G), A \cup E')$ is connected: If every vertex of s except v_A and v_Ω is contained in a connected component of G that also contains another vertex of t', then augmenting G with E' results in a connected graph, making E' an Eulerian extension for G. □

Let us apply the shortcut transformation for an assumption about how often trails in Eulerian extensions visit a connected component of G.

[5] By such a maximal trail t, we mean a trail such that adding any further arc from E to t would not result in a trail.

Observation 2.1. *For any Eulerian extension E of a multigraph G, there is an Eulerian extension E' of G of at most the same weight such that any trail t in E' does not visit a connected component of G twice, except for the initial and terminal vertex of t.*

Observation 2.1 is easy to prove, since, clearly, Lemma 2.1(iii) holds for a minimum subtrail of t that represents the second visit of a connected component. Observe that Observation 2.1 also implies that every maximal trail t in an Eulerian extension is either a path or cycle, because if t would visit a vertex twice, then it would visit its connected component twice. Hence, we get that for every Eulerian extension there is an Eulerian extension of at most the same weight that can be decomposed into paths and cycles that contain at most $c + 1$ vertices, where c is the number of connected components in G.

We now have decompositions of Eulerian extensions according to Theorem 2.1(i) and (ii). To prove the remaining two statements we have to refine our observations by looking at the component graph of G and multiple trails. The following lemma is a generalization of statement (iii) in Lemma 2.1.

Lemma 2.2. *Let E be an Eulerian extension of G, let t and r be trails in the directed multigraph $(V(G), E)$ such that the trails $\mathbb{C}_G(r)$ and $\mathbb{C}_G(t)$ are not vertex-disjoint. Furthermore, let s be a subtrail of t in the directed multigraph $(V(G), E)$ such that $\mathbb{C}_G(s)$ is a subtrail of $\mathbb{C}_G(r)$. Let s' be a subtrail of t such that s is a subtrail of s' and s traverses exactly one vertex less than s'. Set $(E', t') = \text{shortcut}(E, t, s')$. Then E' is an Eulerian extension for G.*

Proof. Lemma 2.1 shows that the vertices in $G + E'$ are balanced. It remains to show that the resulting graph is connected: Any connected component that is traversed by s is also traversed by r. The trails $\mathbb{C}_G(r)$ and $\mathbb{C}_G(t')$ still share a vertex, because of the way we have chosen s. Thus, $G + E'$ is connected. □

By shortcutting subtrails s that are shared by two trails t_1, t_2 in an Eulerian extension, in the sense that $\mathbb{C}_G(s)$ is a subtrail of both $\mathbb{C}_G(t_1)$ and $\mathbb{C}_G(t_2)$, Observation 2.2 directly follows from Lemma 2.2.

Observation 2.2. *For any Eulerian extension E of G, there is an Eulerian extension E' of G of at most the same weight such that for any two edge-disjoint trails t_1, t_2 in E' it holds that $\mathbb{C}_G(t_1), \mathbb{C}_G(t_2)$ either are vertex-disjoint, share at most one vertex, or share only their initial and terminal vertices.*

This proves statement (iii) in Theorem 2.1. Next, we turn to statement (iv):

Observation 2.3. *For any Eulerian extension E of G, there is an Eulerian extension E' of G of at most the same weight such that for any set of edge-disjoint trails $\{t_1, \ldots, t_k\}$ in E' it holds that the graph defined by the union of all trails $\mathbb{C}_G(t_1)', \ldots, \mathbb{C}_G(t_k)'$ does not contain a cycle as subgraph, where $\mathbb{C}_G(t_i)'$ is $\mathbb{C}_G(t_i)$ without the initial vertex.*

Proof. Assume that the graph C defined by the union of $\mathbb{C}_G(t_1)', \ldots, \mathbb{C}_G(t_k)'$ contains a cycle c. Let $e \in t_i$ be an arbitrary edge on c. There is a subtrail s of t_i such that $\mathbb{C}_G(s)$ traverses e and exactly one further edge—recall that $\mathbb{C}_G(t_i)'$ is $\mathbb{C}_G(t_i)$ without the initial vertex. Let $(E', t_i') = \text{shortcut}(E, t_i, s)$. Since $\mathbb{C}_G(t_i')$ is not vertex-disjoint from c, the Eulerian extension E' still connects the graph G (Lemma 2.2). Iterating the shortcutting

for every cycle in the graph C eventually removes every cycle after a finite number of steps, because obviously the statement of Observation 2.3 holds true if $t_1, \ldots, t_n$ have length one, and because in every step the number of arcs in E decreases by one. □

This concludes the proof of Theorem 2.1.

3. Advice

This section introduces special restricted variants of Eulerian Extension (EE) that serve as intermediate problems for our reductions from EE to Conjoining Bipartite Matching and back. We give a reduction from EE to one of the variants and a reduction from another variant problem to EE. These reductions represent the first step towards proving the equivalence of EE and Conjoining Bipartite Matching and the second and final step is given in Section 4.

Since Eulerian extensions have to balance every vertex, they contain paths starting in vertices with positive balance and ending in vertices with negative balance. These paths together with cycles have to connect all connected components of the input graph. In order to further restrict solutions, we are searching for, we use so-called "advice" as additional information on the structure of optimal Eulerian extensions. Advice consists of hints which specify that there must be a path or cycle in an Eulerian extension that visits connected components in a specified order. Hints, however, do not specify exactly which vertices these paths or cycles visit.

Definition 3.1. A *hint* for a directed multigraph $G = (V, A)$ is an undirected path or cycle t of length at least one in the component graph $\mathbb{C}_G$ together with a flag determining whether t is a cycle or a path.[6] Depending on this flag, the hints are called *cycle hints* and *path hints*, respectively. A set of hints H is an *advice* for the graph G if the hints are edge-disjoint.[7] A path p in the directed graph $(V, V \times V)$ *realizes* a path hint h if $\mathbb{C}_G(p) = h$ and the initial vertex of p has positive balance and the terminal vertex has negative balance in G. A cycle c in the graph $(V, V \times V)$ realizes a cycle hint h if $\mathbb{C}_G(c) = h$. An Eulerian extension E *heeds the advice* H if it can be decomposed into a set of paths and cycles that realize all hints in H.

A topic in this work is how having an advice helps in solving an instance of Eulerian extension. In order to discuss this, we introduce the following version of EE.

Eulerian Extension with Advice (EEA)
Input: A directed multigraph $G = (V, A)$, an integer $\omega_{\max}$, a weight function $\omega\colon V \times V \to [0, \omega_{\max}] \cup \{\infty\}$, and advice H.
Question: Is there an Eulerian extension E of G that is of weight at most $\omega_{\max}$ and heeds the advice H?

For an example of advice, see Figure 3. We will see that the hard part of computing an Eulerian extension that heeds a given advice H is to choose initial and terminal vertices

[6] The flag is necessary because a hint to a path in $\mathbb{C}_G$ may correspond to a cycle in G.
[7] Note that there is a difference between advice in our sense and the notion of advice in computational complexity theory. There, an advice applies to every instance of a specific length.

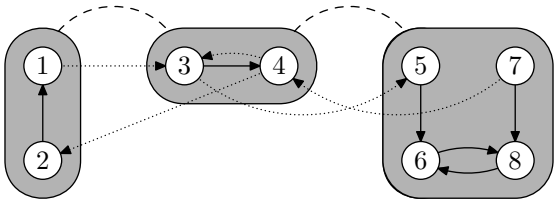


Figure 3: An instance of EEA comprising the vertices 1 through 8 and the solid arcs. Gray objects represent components of the input graph G and the the dashed lines constitute a hint h that forms a piece of advice $P = \{h\}$ for G. The dotted arcs form an Eulerian extension E of G. Both the paths traversing the vertices $1, 3, 5$ and $7, 4, 2$ realize h. Thus, E heeds P.

for path hints in H. In fact, when the endpoints are given, it is possible to compute a realization of a path hint in quadratic time. We use this fact in reductions and formalize it as follows.

Definition 3.2. Let the directed multigraph $G = (V, A)$ and the weight function $\omega : V \times V \to [0, \omega_{\max}] \cup \{\infty\}$ constitute an instance of EE. Let p be a path in $\mathbb{C}_G$ and let u be a vertex in the component of G that corresponds to the initial vertex of p and let v be a vertex in the component that corresponds to the terminal vertex of p. Then, minpath(G, ω, p, u, v) denotes the shortest path s from u to v in the complete graph $(V, V \times V)$ such that $\mathbb{C}_G(s) = p$.

Lemma 3.1. minpath(G, ω, p, u, v) *is computable in* $O(n^2)$ *time.*

Proof. To determine minpath(G, ω, p, u, v), compute a shortest path in the graph $(V, V \times V)$ with a modified weight-function ω': Simply orient the path p such that it leads from the component that contains u to the component that contains v. Then, set the weight to ∞ for all arcs in G that lead from one component to another component such that there is no corresponding arc on p.

The above described algorithm can be carried out in $O(n^2)$ time using Dijkstra's algorithm. By Fact 2.2 we may assume that for the shortest path s computed using ω' it holds that $\mathbb{C}_G(s) = p$ and, thus, the algorithm is correct. □

By a simple modification of the algorithm for minpath, we can also compute an optimal realization for a cycle hint in any given advice in $O(n^3)$ time.

Observation 3.1. *Let* $(G, \omega_{max}, \omega, H)$ *be an instance of* EEA. *In* $O(|H|n^3)$ *time, we can compute an equivalent instance* $(G', \omega_{max}, \omega, H')$ *such that* H' *does not contain a cycle hint. Furthermore, the number of components does not increase.*

Proof. For any cycle hint h, we can choose one connected component C that it traverses and introduce a copy of it into G, extending the weight function accordingly. Then for every vertex v in C we proceed as in Lemma 3.1, computing a shortest path from v to its copy with a modified weight function and keeping the shortest of these paths. Then, merging C and its copy, we get a cycle l such that $\mathbb{C}_G(l) = h$. □

Since we want to derive Eulerian extensions from an advice and every Eulerian extension for a multigraph connects all of the multigraphs connected components, we are mainly interested in "connecting" advice. We say that an advice for a directed multigraph G is *connecting* if all of its hints together connect all vertices in $\mathbb{C}_G$. Furthermore, if there is no connecting advice H' with $H' \subset H$ for a connecting advice H, then H is called *minimal connecting* advice. We consider the following restricted version of EEA that allows only minimal connecting advice (note that, by Observation 3.1, we can assume the given advice to be cycle-free).

Eulerian Extension with Cycle-free Minimal Connecting Advice (EE∅CA)
Input: A directed multigraph $G = (V, A)$, an integer $\omega_{\max}$, a weight function $\omega\colon V \times V \to [0, \omega_{\max}] \cup \{\infty\}$, and minimal connecting cycle-free advice H.
Question: Is there an Eulerian extension E of G that is of weight at most $\omega_{\max}$ and heeds the advice H?

In Section 3.1, we will show how each minimal connecting cycle-free advice can be obtained from a forest in $\mathbb{C}_G$, yielding a parameterized Turing reduction from EE to EE∅CA.

3.1. Deriving Advice from a Minimum-Weight Eulerian Extension

We now combine Theorem 2.1 with the notion of advice and an algorithm to enumerate relevant advices. This yields a Turing reduction from EE to EE∅CA and enables us to use EE∅CA as intermediate problem in a reduction from EE to CBM. In this section, we prove the following.

Theorem 3.1. Eulerian Extension *is* $\leq_{\mathrm{T}}^{\mathrm{FPT}}$*-reducible to* Eulerian Extension with Cycle-free Minimal Connecting Advice *in* $\mathrm{O}(c^{3c+1}n^3)$ *time, where both problems are parameterized by the number* c *of connected components in the input graph.*

To prove Theorem 3.1, first, let us apply Theorem 2.1 to advice in order to restrict the number of advices we have to consider:

Lemma 3.2. *Let G be a directed multigraph with c connected components and let E be a minimum-weight Eulerian extension with respect to a weight function $\omega : V \times V \to [0, \omega_{max}] \cup \{\infty\}$ for G. There is a minimal connecting advice $H = \{h_1, \ldots, h_i\}$ such that*
(i) E heeds H,
(ii) $|H| \leq c$, and
(iii) the graph defined by the union of all trails $h_1, \ldots, h_i$ without their initial vertices does not contain a cycle.

Proof. By Theorem 2.1, there is a decomposition of E into paths and cycles $t_1, \ldots, t_k$ such that the graph defined by the union of all trails $\mathbb{C}_G(t_1), \ldots, \mathbb{C}_G(t_k)$ without their initial vertices does not contain a cycle. We greedily take paths $\mathbb{C}_G(t_j)$ of length at least one into H that connect new vertices in $\mathbb{C}_G$. Statement (i) is trivial. Statement (ii) follows, because there are at most c connected components in G and we only take paths into H that connect new components. Finally, from Theorem 2.1(iv) we get statement (iii). □

Using the above Lemma 3.2 about minimal connecting advice, we can restrict its size, giving a relatively efficient way to enumerate all such advices.

Proof of Theorem 3.1. Let the directed multigraph $G = (V, A)$ and the weight function $\omega : V \times V \to [0, \omega_{\max}] \cup \{\infty\}$ constitute an instance of EE and let c be the number of connected components in G. We give an algorithm that decides EE using an oracle for EE∅CA.

We simply generate all relevant advices, realize each cycle hint, and apply the oracle to the resulting instances. If one of the oracle calls accepts the advice-instance, then, clearly, the original instance is a yes-instance. Also, for every yes-instance of EE, there is an advice derivable from a solution to the instance because of Lemma 3.2. Clearly, the number of components does not increase in instances passed to the oracle.

Concerning the generation of the advices, by Lemma 3.2 we may assume that the hints without their initial vertices form a forest in $\mathbb{C}_G$. Thus, since there are at most c hints in a minimal connecting advice, every minimal connecting advice contains at most $2c$ edges. Thus, we may construct hints in the component graph in a recursive fashion as follows: First, choose one of the c vertices as starting point for a hint, then branch into the cases of extending the hint to one of the remaining at most $c - 1$ vertices. For each of them then recursively branch into the cases of ending the hint or extending it further to one of the remaining vertices. When choosing to end the hint, if the graph is not connected yet, create a next hint, by again choosing a starting vertex and branching analogously to the first hint. End the procedure when the graph is connected, c hints have been generated, or the hints generated so far contain $2c$ edges. Output the set of hints as an advice, if the hints connect all vertices in the component graph.

In this way, the algorithm branches at most $2c + c$ times ($2c$ extensions and c starting vertices) into at most c cases (ending the hint, or extending it to one of at most $c - 1$ vertices). Checking whether the hints connect all vertices can be done in $O(c)$ time. This gives an algorithm to enumerate all advices according to Lemma 3.2 in $O(c^{3c}c)$ time. Additionally, we have to account for posing the oracle question in linear time and for computing realizations for all cycle hints, which can be done $O(cn^3)$ time. Thus, iterating over all relevant advices and applying the oracle takes altogether $O(c^{3c+1}n^3)$ time. □

Using the above reduction, we also obtain a simple fixed-parameter algorithm for EE with respect to the combination of the parameters c and a slightly more complicated parameter:

Corollary 3.1. Eulerian Extension *can be solved in* $O((bc)^{3c}n^3 \log n)$ *time, where c is the number of components in the input graph and $b = \sum_{v \in I_G^+} \text{balance}(v)$ is the sum of all positive balances.*

Proof. To prove this, we use a result of Dorn et al. [10]: EE is solvable in $O(n^3 \log n)$ time if the input multigraph is connected. An algorithm for general EE achieving the above running time first preprocesses instances of EE in $O(n^3)$ time such that Fact 2.1 and Fact 2.2 holds. Then, it uses the Turing reduction from Theorem 3.1 to enumerate instances of EE∅CA. In each of these instances, it enumerates all possible combinations of initial and terminal vertices for realizations of the path hints and computes a

weight-minimal realization for each of the hints using these initial and terminal vertices (Lemma 3.1). Since both the number of vertices of positive balance and the number of vertices of negative balance is bounded by b, there are at most b^{2c} such combinations. Implementing the realizations yields a connected graph, and the algorithm finally solves the resulting instance in $O(n^3 \log n)$ time using the above mentioned result by Dorn et al. [10].

It is not hard to see that this algorithm is correct. To prove the running-time bound, we first need to note that the preprocessing routines we introduced in Section 2.2 preserve b. For the routine used for Fact 2.2, this is easy to see, since modifying the weight function does not alter balances of vertices. The routine used for Fact 2.1 also preserves b, because in each modification step a balanced vertex is introduced and an arc is shifted from one vertex to another. Thus, the sum of all positive balances remains the same. Theorem 3.1 also preserves b since instances of EE are only modified by adding advice and realizing cycle hints. Hence, the running time is $O(n(n+m) + n^3 + c^{3c}c(b^{2c}(cn^2 + n^3 \log n)) \subseteq O((bc)^{3c}n^3 \log n)$ in total. □

3.2. Removing Advice

In order to prove the parameterized equivalence of EE and CBM, we also use an advice problem—in particular, Eulerian Extension with Advice (EEA)—as an intermediate problem in the reduction from CBM to EE. Thus, we have to prove that the advice in a given instance of EEA can be removed, yielding an equivalent instance of EE. That is, we have to prove the following theorem.

Theorem 3.2. Eulerian Extension with Advice *is* $\leq_{\mathrm{m}}^{\mathrm{PPP}}$*-reducible to* Eulerian Extension *with respect to the parameter "number of components in the input graph."*

To eventually prove Theorem 3.2, we show that there is only a polynomial number of optimal ways to realize a hint in an advice. Each of these realizations will be modeled by a pair of imbalanced vertices. These pairs will reside in a new component and this component then can only be connected to the rest of the graph by taking arcs into an Eulerian extension that also connect each component corresponding to inner vertices of the hint.

For convenience, due to Observation 3.1, we assume that all instances of EEA contain cycle-free advice. We first give an intuitive description of the reduction, followed by a detailed construction and then a correctness proof. The construction uses the minpath function introduced in Section 3.

Outline of the Reduction. We look at the hints present in an EEA instance and eliminate them one at a time: For every hint p_i in the advice, first, a connected component is introduced (vertex set W_1^i, arc sets $B_1^{i,\pm}, B_1^{i,=}$ in the construction below) and copied for every inner vertex of the hint (vertex sets W_l^i, arc sets $B_l^{i,\pm}, B_l^{i,=}$ for $2 \leq l \leq k-1$). Each copy is connected to the component corresponding to its vertex in the hint (by the arc set $B_l^{i,\gamma}$). The new component and its copies consist of interconnected imbalanced pairs of vertices. In the construction below, these are the vertices $s_{l,u,v}^{i,\pm}, t_{l,u,v}^{i,\pm}$ contained in the i-th component. Each pair corresponds to a pair of vertices u, v forming the endpoints of a path that realizes the currently considered hint p_i.

A slightly modified weight function ensures that adding an arc $(u, t^{i,+}_{1,u,v})$ or an arc $(s^{i,-}_{1,u,v}, v)$ to an Eulerian extension has the same weight as a minimum-weight realization of the hint that goes from u to v or from v to u, respectively. Notice that the superscript "+"corresponds to paths in one direction and the superscript "−" to paths in the opposite direction. The weight function also ensures that if such an arc is present in an Eulerian extension, then the connected components traversed by the hint are connected to each other.

Construction 3.1. Let the directed multigraph $G_0 = (V_0, A_0)$, the integer $\omega_{\max}$, the weight function $\omega_0 : V_0 \times V_0 \to [0, \omega_{\max}] \cup \{\infty\}$, and the advice P constitute an instance I_{EEA} of EEA. Let $p_1, \ldots, p_d$ be the elements of P and let $C_1, \ldots, C_c$ be the connected components of G.

For every $p_i, 1 \leq i \leq d$, inductively define G_i and ω_i as follows: Let $C_{j_1}, \ldots, C_{j_k}$ be the components of G that correspond to the vertices traversed by p_i, ordered according to an arbitrary path orientation of p_i. For every $1 \leq l \leq k-1$, introduce the vertex set

$$W^{i,+}_l \coloneqq \{t^{i,+}_{l,u,v}, s^{i,+}_{l,u,v} : u \in C_{j_1} \cap I^+_G \wedge v \in C_{j_k} \cap I^-_G\}, \text{ and}$$
$$W^{i,-}_l \coloneqq \{s^{i,-}_{l,u,v}, t^{i,-}_{l,u,v} : u \in C_{j_1} \cap I^-_G \wedge v \in C_{j_k} \cap I^+_G\}.$$

Set $W^i_l \coloneqq W^{i,+}_l \cup W^{i,-}_l$. Make all these vertices imbalanced via the arc set

$$B^{i,\pm}_l \coloneqq \{(t^{i,+}_{l,u,v}, s^{i,+}_{l,u,v}), (t^{i,-}_{l,u,v}, s^{i,-}_{l,u,v})\}.$$

Let $w^1_l, \ldots, w^h_l$ be the vertices in W^i_l. For each $1 \leq l \leq k-1$, interconnect these vertices via a cycle, using the following arc set

$$B^{i,=}_l \coloneqq \{(w^g_l, w^{g+1}_l) : 1 \leq g < h\} \cup \{(w^h_l, w^1_l)\}.$$

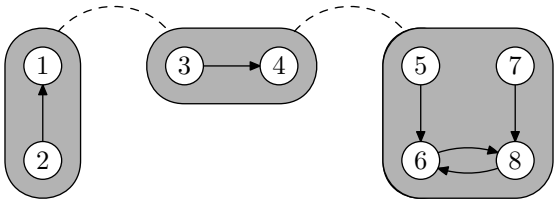
Furthermore, for each $2 \leq l \leq k-1$, choose $c_{j_l} \in C_{j_l}$ and $w_l \in W^i_l$ arbitrarily and add the following arc set connecting W^i_l to C_{j_l}:

$$B^{i,\gamma}_l \coloneqq \{(w_l, c_{j_l}), (c_{j_l}, w_l)\}.$$

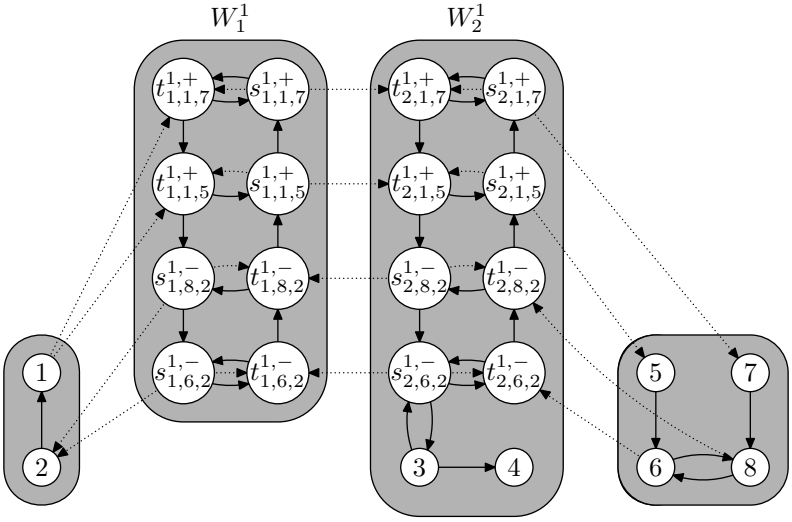
Now set $G_i = (V_i, A_i) \coloneqq (V_{i-1} \cup \bigcup_{l=1}^{k-1} W^i_l, A_{i-1} \cup \bigcup_{l=1}^{k-1}(B^{i,\pm}_l \cup B^{i,=}_l) \cup \bigcup_{l=2}^{k-1} B^{i,\gamma}_l))$ and create a new weight function as follows:

$$\omega_i(u,v) \coloneqq \begin{cases} \omega_{i-1}(u,v), & \text{if } u, v \in V_{i-1} \\ \omega_0(\text{minpath}(G_0, \omega_0, p_i, u, x)), & \text{if } u \in C_{j_1} \cap I^+_G, v = t^{i,+}_{1,u,x} \\ \omega_0(\text{minpath}(G_0, \omega_0, p_i, x, v)), & \text{if } u = s^{i,-}_{1,x,v}, v \in C_{j_1} \cap I^-_G \\ 0, & \text{if } u = s^{i,+}_{k-1,x,v}, v \in C_{j_k} \cap I^-_G \\ 0, & \text{if } u \in C_{j_k} \cap I^+_G, v = t^{i,-}_{k-1,u,x} \\ 0, & \text{if } u = s^{i,\pm}_{l,x,y}, v = t^{i,\pm}_{l,x,y} \\ 0, & \text{if } u = s^{i,\pm}_{l,x,y}, v = t^{i,\pm}_{l+1,x,y} \\ \infty, & \text{otherwise} \end{cases}$$

The graph G_d, the weight function ω_d and the integer $\omega_{\max}$ constitute an instance I_{EE} of EE.



(a) EEA instance



(b) EE instance

Figure 4: Example for Construction 3.1 explained in Example 3.1.

Example 3.1. See Figure 4. At the top, an instance I_{EEA} of EEA is shown. It comprises three connected components and an advice consisting of a single hint p_1 represented by the dashed edges. Below, there is an instance I_{EE} of EE produced by Construction 3.1. The dotted arcs represent the only arcs incident to the new vertices with weight potentially lower than ∞.

In the new instance, the hint p_1 is removed and a new component W_1^1 is introduced. A copy W_2^1 of the vertex set W_1^1 is introduced and connected to the component that corresponds to the inner vertex of p_1. The induced subgraphs of W_1^1, W_2^1 consist of pairs $t^{i,+}_{l,u,v}, s^{i,+}_{l,u,v}$ of vertices that are made imbalanced and that are connected via a directed cycle. Each of the vertices $s^{i,+}_{l,u,v}$—the "sources"—has balance 1 and, because of the way that the weight function is defined, can either be connected to a vertex $t^{i,+}_{l,u,v}$—the "targets"—inside the same component or to another component. Analogously, targets can only accept at most one arc from either inside the same component or from outside.

Consider a solution E to I_{EEA} that also contains the arcs $(1,3),(3,5)$ as realization

of p_1. We may remove these arcs and add the arcs

$$(1, t^{1,+}_{1,1,5}), (s^{1,+}_{1,1,5}, t^{1,+}_{2,1,5}), (s^{1,+}_{2,1,5}, 5)$$

to E, and add arcs from all remaining sources to their corresponding targets that reside in the same component to obtain a solution to I_{EE}. Also, every solution to I_{EE} has to connect the connected component W^1_1 to the rest of the graph. This is only possible by adding an arc from a source to outside its component, for example at $s^{1,-}_{1,6,2}$. Then the vertex $t^{1,-}_{1,6,2}$ has to fetch an arc from $s^{1,-}_{2,6,2}$ in the Eulerian extension in order to become balanced. This means that then also the arc $(6, t^{1,-}_{2,6,2})$ has to be included in an Eulerian extension for I_{EEA} and thus we can include the path from vertex 6 to vertex 2 that realizes p_1 computed by the minpath function.

Polynomial-time Computability and Correctness. We first prove that Construction 3.1 is polynomial-time computable and that the parameter in the reduced instance is polynomial in the original parameter. We then proceed to show the correctness of the construction.

Observation 3.2. *Construction 3.1 can be performed in polynomial-time. There are* $\mathrm{O}(c^2)$ *components in* G_d.

Proof. We first look at the running time of the construction: The size of W^i_l and the arc sets $B^{i,\pm}_l, B^{i,=}_l, B^{i,\gamma}_l$ is at most $\mathrm{O}(n^2)$. It holds that $l \leq c$ and there are at most $\mathrm{O}(c^2)$ hints in an advice (recall that hints in an advice are edge-disjoint). Hence, at most $\mathrm{O}(c^3 n^2)$ vertices and edges are added. This can be done in time linear in the number of added vertices and edges. Thus, the new weight-function can be computed in $\mathrm{O}(c^6 n^4)$ time and this yields a polynomial-time algorithm for Construction 3.1.

Since there are at most $\mathrm{O}(c^2)$ hints in an advice and for every hint there is exactly one new component (the component with vertex set W^i_1) in the reduced instance, the value of the new parameter is in $\mathrm{O}(c^2)$. □

Now we are ready to prove Theorem 3.2.

Proof of Theorem 3.2. By Observation 3.2 it only remains to show that Construction 3.1 is correct. For this, first consider an Eulerian extension E that is a solution to I_{EEA}. For every hint p_i, the set E contains a set of paths that realize p_i. Without loss of generality, we may assume that among those paths is the path $s := \text{minpath}(G_0, \omega_0, p_i, u, v)$. Here, u and v are chosen suitably from components that p_i starts and ends in, respectively. If s is not contained in E, then we can obtain a Eulerian extension of at most the same weight that contains s by simply substituting s for the corresponding realization of p_i. Thus, in order to connect the component W^i_l to the rest of the graph, we may remove s from E and add the arcs

$$(u, t^{i,+}_{1,u,v}), (s^{i,+}_{1,u,v}, t^{i,+}_{2,u,v}), \ldots, (s^{i,+}_{k-2,u,v}, t^{i,+}_{k-1,u,v}), (s^{i,+}_{k-1,u,v}, v).$$

This does not increase the weight of E. To balance all vertices $t^{i,+}_{l,u',v'}, s^{i,+}_{l,u',v'}$ with $1 \leq l \leq k-1, u' \neq u, v' \neq v$, we may add the corresponding arcs $(s^{i,+}_{l,u',v'}, t^{i,-}_{l,u',v'})$ and analogously

for vertices in $W_l^{i,-}$, again without increasing the weight. Thus, doing this for every hint yields an Eulerian extension for I_{EE} of the same weight.

Now consider an Eulerian extension E that is a solution to I_{EE}. The set E has to connect the component W_1^i to the rest of the graph for every hint p_i. Thus, without loss of generality, there is an arc $(u, t_{1,u,v}^{i,+})$ for some vertices u, v in the components that correspond to the endpoints of p_i. For every vertex $t_{l,x,y}^{j,\pm}$, there are only incoming arcs with weight lower than ∞, and since it has balance -1, there is exactly one arc incident to it in E. The same is true for vertices $s_{l,x,y}^{j,\pm}$ since all arcs with weight lower than ∞ start at them and they have balance 1. Hence the arc $(s_{1,u,v}^{i,+}, t_{2,u,v}^{i,+})$ is present in E, by induction also $(s_{l,u,v}^{i,+}, t_{l+1,u,v}^{i,+}) \in E$, $1 \leq l \leq k-2$, and finally also $(s_{k-1,u,v}^{i,+}, v) \in E$. Thus we can remove these arcs from E, add minpath$(G_0, \omega_0, p_i, u, v)$, and repeat this for all hints to obtain an Eulerian extension for G_0 that heeds the advice P and has weight at most $\omega_{\max}$. □

4. Eulerian Extension and Conjoining Bipartite Matching

This section gathers the remaining building blocks for the parameterized equivalence of Eulerian Extension (EE) and Conjoining Bipartite Matching (CBM). We first introduce CBM—a variant of perfect bipartite matching—and then show how CBM relates to the two variants of EE we have introduced in Section 3.

Definition 4.1. Let G be a bipartite graph and let P be a vertex partition with the cells $C_1, \ldots, C_k$. An unordered pair $\{i, j\}$ of integers $1 \leq i < j \leq k$ is a *join* and a set J of such pairs is a *join set* with respect to G and P. We say that a join $\{i, j\} \in J$ is *satisfied* by a matching $M \subseteq E(G)$ if there is at least one edge $e \in M$ with $e \cap C_i \neq \emptyset$ and $e \cap C_j \neq \emptyset$. We say that a matching M of G is *J-conjoining* with respect to a join set J if all joins in J are satisfied by M. If the join set is clear from the context, we simply say that M is conjoining. A matching M in the graph G is called *perfect*, if each vertex in G has an incident edge in M.

Using these definitions, we can conveniently state CBM.

Conjoining Bipartite Matching (CBM)
Input: A bipartite graph $G = (V_1 \uplus V_2, E)$, an integer $\omega_{\max}$, a weight function $\omega\colon E \to [0, \omega_{\max}]$, a partition $P = \{C_1, \ldots, C_k\}$ of the vertices in G, and a join set J.
Question: Is there a matching M in G such that M is perfect, M is conjoining and M has weight at most $\omega_{\max}$?

CBM can be interpreted as a job assignment problem with additional constraints: an assignment of workers to tasks is sought such that each worker is busy and each task is being processed. Furthermore, every worker must be qualified for the assigned task. Both, the workers and the tasks, are grouped and the additional constraints are of the form "At least one worker from group A must be assigned a task in group B". An assignment that satisfies such additional constraints may be favorable in settings where the workers are assigned to projects and the projects demand at least one worker with additional qualifications.

4.1. From Eulerian Extension to Matching

In this section, we give a reduction from EE∅CA to CBM yielding the following theorem.

Theorem 4.1. Eulerian Extension with Cycle-free Minimal Connecting Advice *is* $\leq_{\mathrm{m}}^{\mathrm{PPP}}$*-reducible to* Conjoining Bipartite Matching *with respect to the parameters "number of components" and "join set size."*

Outline of the Reduction. The basic idea of our reduction is to use vertices of positive balance and negative balance in an instance of EE∅CA as the two cells of the graph bipartition in a designated instance of CBM. Edges between vertices in the new instances represent shortest paths between these vertices that consist of allowed extension arcs in the original instance. Every connected component in the original instance is represented by a cell in the partition in the matching instance and hints are basically modeled by joins.

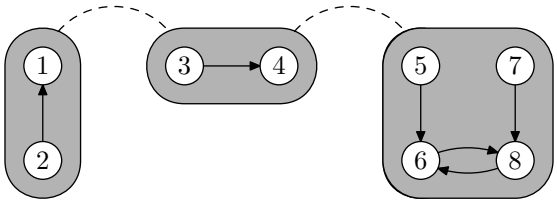
Description of the Reduction. For the description of the reduction, we need the following definition.

Definition 4.2. Let $C_1, \ldots, C_c$ be the connected components of a directed multigraph G, and let H be a cycle-free advice for G. For every $h \in H$, define $\mathrm{connect}(h) := \{i, j\}$, where C_i, C_j are the components corresponding to the initial and terminal vertices of h.

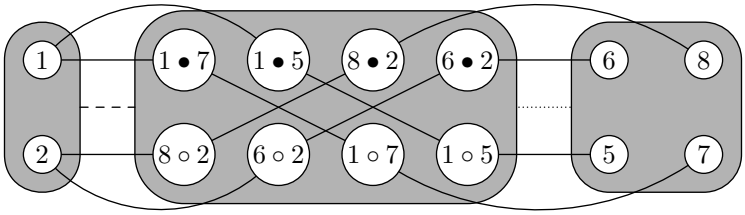
First, consider an EE∅CA-instance $(G, \omega_{\max}, \omega, H)$ such that H is a cycle-free minimal connecting advice that contains only hints of length one. We will deal with longer hints later. We create an instance I_{CBM} of CBM by first defining $B_0 = (I_G^+ \uplus I_G^-, E_0)$ as a bipartite graph. Here, the set E_0 consists of all edges $\{u, v\}$ such that $u \in I_G^+$, $v \in I_G^-$, and $\omega(u, v) < \infty$. This serves the purpose of modeling the structure of "allowed" arcs in the matching instance. Next, we derive a vertex partition $\{V_1', \ldots, V_c'\}$ of B_0 by intersecting the connected components of G with $(I_G^+ \uplus I_G^-)$. The vertex partition obviously models the connected components in the input graph, and the need for connecting them according to the advice H is modeled by an appropriate join-set J_0, defined as $\{\mathrm{connect}(h) : h \in H\}$. Finally, we make sure that matchings also correspond to Eulerian extensions weight-wise, by defining the weight function $\omega'(\{u, v\})$ for every $u \in I_G^+, v \in I_G^-$ as $\omega(u, v)$ with $\omega'_{\max} = \omega_{\max}$.

By Observation 2.1 we may assume that every hint in H of length one is realized by a single arc. Since the advice connects all connected components, by the same observation, we may assume that all other trails in a valid Eulerian extension have length one. Finally, by Fact 2.1, we may assume that every vertex has at most one incident incoming or outgoing arc in the extension and, hence, we get an intuitive correspondence between conjoining matchings and Eulerian extensions.

To model hints of length at least two, we utilize gadgets similar to the one shown in Figure 5. The gadget comprises two vertices ($u \circ v$ and $u \bullet v$) for every pair (u, v) of vertices with one vertex in the component the hint starts and one in the component the hint ends. The vertices $u \circ v$ and $u \bullet v$ are adjacent and each of these two vertices is connected with one vertex of the pair it represents. The edge $\{u \bullet v, u\}$ is weighted with the cost it takes to connect u, v with a path that realizes h. This cost is computed using the minpath function introduced in Section 3. The edges $\{u \bullet v, u \circ v\}$ and $\{u \circ v, v\}$ have



(a) EE∅CA instance



(b) Long-hint gadget in CBM instance

Figure 5: Example of the long-hint gadget. In (a), an EE∅A-instance is shown, consisting of a graph with three connected components and an advice that contains a single path hint h (dashed lines). In (b), a part of an instance of CBM is shown, comprising the cells that correspond to the initial and terminal vertices of h and a gadget to model h. The gadget consists of new vertices put into a new cell which is connected by two joins (dashed and dotted lines) to the cells corresponding to the initial and terminal vertices of h.

weight 0. Intuitively these three edges in the gadget represent one concrete realization of h. If $u \circ v$ and $u \bullet v$ are matched, this means that this specific path does not occur in a designated Eulerian extension. However, by adding the vertices of the gadget as cell to the vertex partition and by extending the join set to the gadget, we enforce that there is at least one outgoing edge that is matched. If a perfect matching matches $u \circ v$ with u, then it also matches $u \bullet v$ with v and vice versa. This introduces an edge to the matching that has weight corresponding to a path that realizes h.

More formally, we use the following construction:

Construction 4.1. Let the directed multigraph $G = (V, A)$, the integer $\omega_{\max}$, the weight function $\omega : V \times V \to [0, \omega_{\max}] \cup \{\infty\}$ and the advice H constitute an instance of EE∅CA. Let $V_1, \ldots, V_c$ be the connected components of G.

Let $H^{=1}$ be the set of hints of length one in H and let $H^{\geq 2}$ be the set of hints in H that have length at least two. Define J_0 by the set $\{\text{connect}(h) : h \in H^{=1}\}$. Let $W_0^1 \coloneqq I_G^+$, $W_0^2 \coloneqq I_G^-$, and let $B_0 = (W_0^1 \uplus W_0^2, E_0)$ be a bipartite graph where

$$E_0 \coloneqq \{\{u, v\} : u \in I_G^+ \wedge v \in I_G^- \wedge \omega(u, v) < \infty\}.$$

Define $V_i' \coloneqq V_i \cap (I_G^+ \cup I_G^-)$, $1 \leq i \leq c$, and $\omega_0'(\{u, v\}) \coloneqq \omega(u, v)$ where $\{u, v\} \in E, u \in I_G^+$.

Next, "long-hint gadgets" are introduced for every hint of length at least two: Let $h_1^{\geq 2}, \ldots, h_j^{\geq 2}$ be the hints in $H^{\geq 2}$. Inductively define B_k, V_{c+k}', ω_k' and J_k, $1 \leq k \leq j$, as

follows: Let connect($h_k^{\geq 2}$) = $\{o, p\}$. Introduce the vertex sets

$$U_1 \coloneqq \{v \circ u : v \in I_G^+ \cap V_o \wedge u \in I_G^- \cap V_p \wedge \omega(\text{minpath}(G, \omega, h_k^{\geq 2}, v, u)) < \infty\} \cup$$
$$\{v \circ u : v \in I_G^- \cap V_o \wedge u \in I_G^+ \cap V_p \wedge \omega(\text{minpath}(G, \omega, h_k^{\geq 2}, u, v)) < \infty\},$$

and $U_2 \coloneqq \{v \bullet u : v \circ u \in U_1\}$. Introduce the edge sets

$$E_k^1 \coloneqq \{\{v \circ u, v\} : v \in I_G^- \wedge v \circ u \in U_1\},$$
$$E_k^2 \coloneqq \{\{v \bullet u, v\} : v \in I_G^+ \wedge v \bullet u \in U_2\}, \text{ and}$$
$$E_k^3 \coloneqq \{\{v \circ u, v \bullet u\} : v \circ u \in U_1 \wedge v \bullet u \in U_2\}.$$

Set $E_k \coloneqq E_k^1 \cup E_k^2 \cup E_k^3$, and set the graph

$$B_k \coloneqq ((W_{k-1}^1 \cup U_1) \uplus (W_{k-1}^2 \cup U_2), E_{k-1} \cup E_k),$$

set $V'_{c+k} \coloneqq U_1 \cup U_2$, set $J_k \coloneqq J_{k-1} \cup \{\{o, c+k\}, \{p, c+k\}\}$ and the weight function as follows:

$$\omega'_k(\{u, v\}) \coloneqq \begin{cases} \omega'_{k-1}(\{u, v\}), & \{u, v\} \in E_{k-1} \\ 0, & \{u, v\} \in E_k^1 \cup E_k^3 \\ \omega(\text{minpath}(G, \omega, h_k^{\geq 2}, v, w)), & \{u, v\} = \{v \bullet w, v\} \in E_k^2 \end{cases}$$

Then the graph B_j, the integer $\omega_{\max}$, the weight function ω'_j, the vertex partition $P \coloneqq \{V_1, ..., V_{c+j}\}$ and the join set C_j constitute an instance of CBM.

For the remainder of this section, let the directed multigraph $G = (V, A)$, the weight function $\omega : V \times V \to [0, \omega_{\max}] \cup \{\infty\}$ and the cycle-free minimal connecting advice H constitute an instance of EE∅CA and let the bipartite graph $B \coloneqq B_j$, the weight function $\omega' \coloneqq \omega'_j$ with the maximum weight $\omega_{\max}$, the vertex partition P and the join set $J \coloneqq J_j$ as in Construction 4.1 constitute an instance of CBM. We first prove both directions of the correctness of the construction and then prove that the running time is polynomial.

Before continuing, we need the following observation.

Observation 4.1. *A maximum trail in an Eulerian extension for a graph G either is closed or starts in I_G^+ and ends in I_G^-.*

Proof. Consider the initial vertex v_A and terminal vertex v_Ω of a trail t in the Eulerian extension E. The vertices v_A and v_Ω are balanced in $G + E$.

Assume that v_Ω is not balanced in G. Every time t traverses v_Ω, it uses one arc in E that enters v_Ω and one that leaves it. This implies that $v_\Omega \neq v_A$ because v_Ω is balanced in $G + E$ and thus there is an odd number of arcs in E incident to v_Ω (recall that t is maximum). Since t ends in v_Ω, this also implies that $v_\Omega \in I_G^-$. Analogously, we get that $v_A \in I_G^+$.

Now assume that v_Ω is balanced in G. Since t cannot be extended, it already uses every arc incident to v_A and v_Ω. However, if v_Ω is not equal to v_A, then there are more arcs entering v_Ω than leaving v_Ω in E. This means that v_Ω is not balanced in $G + E$, a contradiction. □

Next, we show how one can obtain a conjoining matching from a valid Eulerian extension for the original instance of EE∅CA.

Lemma 4.1. *Let E be an Eulerian extension for G that heeds the advice H. Then there is a perfect conjoining matching M for B with $\omega'(M) \leq \omega(E)$.*

Proof. We construct the matching successively by first looking at every long-path gadget in B and then matching the remaining vertices.

Consider the cell $V'_{c+k} \in P$ for $k > 0$. There are two joins $\{c + k, o\}$ and $\{c + k, p\}$ in J. Thus, there is a path hint h from V_o to V_p in H and there is a path s in E that realizes h and starts in a vertex $v \in V$ in the component V_o and ends in a vertex $u \in V$ in V_p. By the definition of minpath, the weight $\omega(s)$ is at least $\omega(\text{minpath}(G, \omega, h, u, v))$. Thus we may match $u \bullet v$ with v, match $u \circ v$ with u (these two edges have weight $\omega(\text{minpath}(G, \omega, h, u, v))$), and match every other pair $w \bullet x, w \circ x$ of vertices in V'_{c+k} with each other (each of these edges have weight 0). Matching like this, we obtain a matching for the long-hint gadget of h that fulfills its two joins and is perfect when restricted to the gadget. The weight of the matching is at most the realization of h in E.

The definition of advice ensures that there is a set of paths in E that is edge-disjoint and realizes all hints in H. Because of this, we may find a matching $M^{\geq 2}$ for B that satisfies the joins of every long-hint gadget and is perfect with respect to the vertex set of each long-hint gadget—as in the previous paragraph, iterated for every gadget. Furthermore, $\omega'(M^{\geq 2})$ is at most the weight according to ω of all paths in E that realize hints of length at least two in H.

Now it is easy to extend $M^{\geq 2}$ to a conjoining matching $M^{\geq 1}$ for B and J just by adding to $M^{\geq 1}$ edges between vertices that realize hints of length one in E. We may assume by Observation 2.1 that each hint of length one is realized by a single arc in E. The weight of such an added edge is exactly the cost of the arc between the corresponding vertices. Because of this, we maintain that $\omega'(M^{\geq 1})$ is at most the weight of all paths in E that realize hints.

Finally, we have to extend $M^{\geq 1}$ to a perfect matching M by matching the remaining non-gadget vertices. We can do this by looking at paths in E that start and end in the vertices in G, corresponding to still unmatched vertices in B. A set of such paths must exist, because each such vertex has at least one incident arc in E and because, by Observation 4.1, maximal open trails in Eulerian extensions start and end in unbalanced vertices. The edges between initial and terminal vertices of these paths in B have at most the weight of such a path (because of Fact 2.2 and because of the definition of ω'). Thus, we can add those edges to $M^{\geq 1}$, obtaining an edge set M. This set is a matching for B that is perfect, conjoining and $\omega'(M) \leq \omega(E)$. □

The following Lemma 4.2 shows that a solution to the matching instance implies a valid Eulerian extension for the original instance. Thus, it concludes the proof of correctness for Construction 4.1.

Lemma 4.2. *Let M be a perfect conjoining matching for B. We can construct an Eulerian extension E for G that heeds the advice H such that $\omega(E) = \omega'(M)$.*

Proof. We simply look at every edge in M that has non-zero weight and add a corresponding path to a designated Eulerian extension E of G: For non-gadget edges in M

(edges that match vertices in $V'_1, \ldots, V'_c$), the corresponding path is the arc between the two vertices in G. For edges that match a vertex v in a cell V'_o, $1 \leq o \leq c$, and a vertex $u \bullet v \in V'_{c+k}$, $1 \leq k \leq j$, where $u \in V'_p, 1 \leq p \leq c$, the corresponding path is minpath(G, ω, h_k, u, v). Here, h_k is the path in H that lead to the introduction of V'_{c+k} (that is, the kth long-hint gadget) in Construction 4.1.

We immediately see that $\omega(E) = \omega'(M)$. Also, it is clear that every hint of length one in H is realized in E because every hint h^1 of length one leads to the pair connect(p^1) in J. Hints $p^{\geq 2}$ of length two are also realized, because every such path leads to a cell V'_{c+k}, where $1 \leq k \leq j$, and also leads to the corresponding joins $\{o, c+k\}$ and $\{p, c+k\}$ in J, where $\{o, p\} = \text{connect}(h^{\geq 2})$. Thus, E heeds the advice H. Since M is a perfect matching, every unbalanced vertex in G is the initial or terminal vertex of exactly one path added to E in the above paragraph. By Fact 2.1 we may assume that this suffices to make every vertex in $G + E$ balanced. Also, $G + E$ is connected, because E heeds the advice H which is a connecting advice. □

To prove Theorem 4.1, it now only remains to show that Construction 4.1 can be carried out in polynomial time.

Lemma 4.3. *Construction 4.1 can be performed in* $O(|H|n^4 + m)$ *time.*

Proof. Computing B_0 takes $O(n^2)$ time. To compute J_0 one needs $O(|H|)$ time by iterating over every path in H. Computing the initial partition $\{V'_1, \ldots, V'_c\}$ takes $O(n + m)$ time and the initial weight function ω'_0 can also be computed within this time. Hence, creating the initial instance is possible in $O(n^2 + m)$ time.

Regarding the addition of the gadget for one path in H, it takes $O(n^4)$ time to compute the sets U_1 and U_2, because n^2 instances of minpath have to be computed, each taking $O(n^2)$ time (see Lemma 3.1). There are only three edges in the gadget for every vertex $v \in U_1$, thus computing the edge sets does not increase the running time bound. For the weight function, we can reuse the values of minpath computed for every pair of vertices $v \in I^+_G, u \in I^-_G$ and thus we can conclude an overall running time bound of $O(|H|n^4 + m)$. □

By Lemma 4.1 and Lemma 4.2, Construction 4.1 is correct, and, by Lemma 4.3, it can be carried out in polynomial time. Hence, Theorem 4.1 follows.

4.2. From Matching to Eulerian Extension with Advice

In the previous section, we have shown that a variant of Eulerian Extension reduces to Conjoining Bipartite Matching (CBM). Now, in the opposite direction, we show that CBM reduces to Eulerian Extension with Advice (EEA). This constitutes the final building block for the equivalence of Eulerian Extension and CBM.

Theorem 4.2. Conjoining Bipartite Matching *is* $\leq^{\text{PPP}}_{\text{m}}$*-reducible to* Eulerian Extension with Advice *with respect to the parameters "join set size" and "connected components in the input graph".*

To prove Theorem 4.2 we first observe that for every instance of CBM there is an equivalent instance such that every cell in the input vertex partition contains equal numbers of vertices from both cells of the graph bipartition. This observation enables us

to model cells as connected components and vertices in the bipartite graph as unbalanced vertices in the designated instance of EEA.

Lemma 4.4. *For every instance of* CBM*, there is an equivalent instance comprising the bipartite graph $G = (V_1 \uplus V_2, E)$, the vertex partition $P = \{C_1, \ldots, C_k\}$ and the join set J, such that*

(i) for every $1 \leq i \leq k$ it holds that $|V_1 \cap C_i| = |V_2 \cap C_i|$, and

(ii) the graph $(P, \{\{C_i, C_j\} : \{i, j\} \in J\})$ is connected.

This equivalent instance contains at most one cell more than the original instance and is polynomial-time computable.

To prove this lemma, we first need the following auxiliary observation.

Observation 4.2. *Let $G = (V_1 \uplus V_2, E)$ be a bipartite graph such that $|V_1| = |V_2|$ and let the set $P = \{C_1, \ldots, C_k\}$ be a partition of the vertices in G. It holds that*

$$\sum_{i:|C_i \cap V_1| > |C_i \cap V_2|} |C_i \cap V_1| - |C_i \cap V_2| = \sum_{i:|C_i \cap V_1| < |C_i \cap V_2|} |C_i \cap V_2| - |C_i \cap V_1|.$$

Proof. Observe that the equation holds if and only if $|V_1| = |V_2|$: Without loss of generality we may assume that there are no cells C_i with $|C_i \cap V_1| = |C_i \cap V_2|$ because these do not contribute summands to the equation. Then we can rewrite the equation such that the left-hand side reads as follows

$$\sum_{i:|C_i \cap V_1| > |C_i \cap V_2|} |C_i \cap V_1| + \sum_{i:|C_i \cap V_1| < |C_i \cap V_2|} |C_i \cap V_1|.$$

This is equal to $|V_1|$. Analogously, the left-hand side in the rewritten formula is equal to $|V_2|$. □

Now, we are able to prove Lemma 4.4.

Proof of Lemma 4.4. We first prove that there is an equivalent instance corresponding to statement (i) and then turn to statement (ii). Let the bipartite graph $G = (V_1 \uplus V_2, E)$, the weight function $\omega : E \to [0, \omega_{\max}] \cup \{\infty\}$, the vertex partition $P = \{C_1, \ldots, C_k\}$ and the join set J constitute an instance I_{CBM} of CBM. First observe that if I_{CBM} is a yes-instance then $|V_1| = |V_2|$, otherwise there could not be a perfect matching. Thus, if $|V_1| \neq |V_2|$ we may simply output a trivial no-instance for which the statement of the lemma holds. Otherwise, for each $1 \leq i \leq k$ let $\alpha_i := |C_i \cap V_1| - |C_i \cap V_2|$. By Observation 4.2, the following procedure can be carried out: Add a new empty cell C_{k+1} to P. At the end of the procedure, C_{k+1} will contain $\sum_{i:\alpha_i > 0} \alpha_i$ vertices in V_1 and the same number of vertices in V_2. Modify the graph G and each cell $C_i \in P$ with $\alpha_i > 0$ as follows: Add new vertices $v_1, \ldots, v_{\alpha_i}$ to V_2 and to the cell C_i, and add an edge from v_j to a vertex in $C_{k+1} \cap V_1$ for every $1 \leq j \leq \alpha_i$ such that every vertex in C_{k+1} gets at most one incident edge. Proceed analogously for cells C_i with $\alpha_i < 0$ by adding vertices to V_1 and adding corresponding edges to C_{k+1}. Finally, expand the weight function ω to the new edges by giving each of them weight 0.

This construction is obviously correct, since each new vertex can only be matched to its corresponding vertex in C_{k+1}.

Concerning statement (ii), assume that the statement does not hold for an instance that contains the vertex partition $P = \{C_1, \ldots, C_k\}$ and a join set J. We greedily choose two cells C_i, C_j that are in different connected components in the "cell-join graph" $(P, \{\{C_i, C_j\} : \{i, j\} \in J\})$, remove them from P, add the cell $C_k := C_i \cup C_j$, and update J accordingly—that is, we replace every join $\{m, l\} \in J$ where $m \in \{i, j\}$ by the join $\{k, l\}$. This is correct because all joins satisfied by any solution M for the new instance are also satisfied by M in the original instance and vice versa. Iterating the merging of cells in different connected components makes the cell-join graph connected and the statement follows. □

Description of the Reduction. To reduce instances of CBM that conform to Lemma 4.4 to instances of EEA we use the simple idea of modeling every cell as connected component, vertices in V_1 as vertices with balance -1, vertices in V_2 as vertices with balance 1, and joins as hints.

Construction 4.2. Let the bipartite graph $B = (V_1 \uplus V_2, E)$, the weight function $\omega\colon E \to [0, \omega_{\max}]$, the vertex partition $P = \{C_1, \ldots, C_k\}$ and the join set J constitute an instance I_{CBM} of CBM that corresponds to Lemma 4.4. Let $v_1^1, v_1^2, \ldots, v_{n/2}^1, v_{n/2}^2$ be a sequence of all vertices chosen alternatingly from V_1 and V_2. Let the graph $G = (V, A) := (V_1 \cup V_2, A_1 \cup A_2)$ where the arc set A_1 assures that each vertex in V_1 has balance -1 and vertices in V_2 have balance 1. The arc set A_2 introduces cycles into the graph such that vertices that stem from the same cell in I_{CBM} are in one connected component of G. For example, we may construct A_1, A_2 as follows: $A_1 := \{(v_i^1, v_i^2) : 1 \leq i \leq n/2\}$. For every $1 \leq j \leq k$, let $C_j = \{v_1, \ldots, v_{j_k}\}$, let

$$A_2^j := \{(v_i, v_{i+1}) : 1 \leq i \leq j_k - 1\} \cup \{(v_{j_k}, v_1\},$$

and define $A_2 := \bigcup_{j=1}^{k} A_2^j$. Then, define a new weight function ω' for every pair of vertices $(u, v) \in V \times V$ by

$$\omega'(u, v) := \begin{cases} \omega(\{u, v\}), & u \in V_2, v \in V_1, \{u, v\} \in E \\ \infty, & \text{otherwise.} \end{cases}$$

Finally, derive an advice H for G by adding a length-one hint h to H for every join $\{o, p\} \in J$ such that h consists of the edge that connects vertices in $\mathbb{C}_G$ that correspond to the connected components C_o, and C_p. The graph G, the weight function ω', the maximum weight $\omega_{\max}$ and the advice H constitute an instance of EEA.

By showing that the above is a $\leq_{\text{m}}^{\text{PPP}}$-reduction we obtain a proof for Theorem 4.2:

Proof of Theorem 4.2. We show that the application of Lemma 4.4 and Construction 4.2 is a $\leq_{\text{m}}^{\text{PPP}}$-reduction from CBM to EEA. It can easily be checked that it can be carried out in polynomial time. Also, by Lemma 4.4 and the definition of A_2 in Construction 4.2, it follows that the instances of EEA generated in this way have a number of connected components that is at most the size of the join set plus one.

Assume that there is a perfect conjoining matching M with weight at most $\omega_{\max}$ for the instance I_{CBM} as in Construction 4.2. Then, we derive an Eulerian extension E

for G that heeds the advice H with the same weight by simply choosing $E := \{(u, v) : u \in I_G^- \wedge \{u, v\} \in M\}$. By the definition of ω', $\omega'(E) = \omega(M)$. Every hint is realized by E because for every join there is an edge in M that satisfies it. Most importantly, E is an Eulerian extension for G: Since M is perfect, every vertex in G has exactly one arc incident in E. Since every vertex in G has balance -1 or 1 (due to the definition of A_1), this suffices to make all vertices balanced. By Lemma 4.4(ii), the advice H is a connecting advice and thus $G + E$ is connected.

Now assume that there is an Eulerian extension E for G that heeds the advice H and has weight at most $\omega_{\max}$. Choosing $M := \{\{u, v\} : (u, v) \in E\}$ yields a perfect conjoining matching of the same weight: It holds the $\omega'(E) = \omega(M)$, because all extension arcs that do not correspond to an edge in B have weight ∞. The matching M is perfect, because every vertex in I_G^- (in I_G^+) has balance -1 (balance 1), has only incoming (outgoing) allowed arcs and thus has exactly one arc incident in E. The matching M is conjoining, because E heeds the advice H. □

4.3. Summary

Over the course of the preceding sections, we gathered the building blocks for proving the following theorem.

Theorem 4.3. *If the problem* Conjoining Bipartite Matching *is W[t]-hard for some t, then* Eulerian Extension *is W[t]-hard. If the problem* Conjoining Bipartite Matching *is fixed-parameter tractable, then* Eulerian Extension *is fixed-parameter tractable. Both statements are with respect to the parameters "join set size" and "number of connected components in the input graph".*[8]

It is easy to see that the $\leq_m^{\text{PPP}}$-equivalence of Eulerian Extension (EE) and Rural Postman given by Dorn et al. [10] also holds for the parameters "number of components" and "number of components in the graph induced by the required arcs." Thus, a statement that is analogous to Theorem 4.3 holds for Rural Postman, when substituting Rural Postman for EE and substituting the parameter accordingly.

Proof of Theorem 4.3. Assume that Conjoining Bipartite Matching (CBM) is W[t]-hard. Then, combining the hardness reduction with the many-one reductions from CBM to Eulerian Extension with Advice (Theorem 4.2) and from there to EE (Theorem 3.2), we obtain a W[t]-hardness reduction for EE.

Now assume that CBM is fixed-parameter tractable. Then, using the $\leq_T^{\text{FPT}}$-reduction from EE to Eulerian Extension with Cycle-free Minimal Connecting Advice (Theorem 3.1), many-one reducing each instance in an oracle question to an instance of CBM (Theorem 4.1), and solving it via the fpt-algorithm yields fixed-parameter tractability for EE. □

In the previous sections, we have set out to step-by-step restrict the solutions for Eulerian Extension that we have to consider. Originally, we hoped for polynomial-time

[8] Note that this is a stronger statement than $\leq_T^{\text{FPT}}$-equivalence of both problems, since it is not clear whether W[t]-hardness under $\leq_T^{\text{FPT}}$-reductions implies W[t]-hardness under parameterized many-one reductions.

algorithms for CBM. However, as we will observe in Section 5, CBM is still NP-hard. Nevertheless, we deem CBM to be more accessible for parameterized complexity analysis. Moreover, we will obtain a tractability result on restricted graphs in the following. This raises hope that CBM might help us to eventually derive a fixed-parameter algorithm for EE.

5. Conjoining Bipartite Matching: Properties and Special Cases

This section investigates the properties of CBM introduced in Section 4. As discussed before, CBM might eventually help us derive a fixed-parameter algorithm for EE with respect to the parameter "number of connected components". Section 5.1 first shows that also CBM is NP-complete. Section 5.2 then establishes tractability of the problem on restricted graph classes and translates this tractability result into the world of EE and RP.

5.1. NP-Hardness

NP-hardness for Conjoining Bipartite Matching (CBM) does not follow from the parameterized equivalence to Eulerian Extension (EE) we gave in Section 4, since the reduction from EE we gave is a *parameterized Turing* reduction. To show that CBM is NP-hard, we polynomial-time many-one reduce from the well-known 3SAT, where a Boolean formula ϕ in 3-conjunctive normal form (3-CNF) is given and it is asked whether there is an assignment to the variables of ϕ that satisfies ϕ. Herein, a formula ϕ in 3-CNF is a conjunction of disjunctions of three literals each, where each literal is either x or $\neg x$ and x is a variable of ϕ. In the following, we represent each clause as three-element-set $\gamma \subseteq X \times \{+, -\}$, where $(x, +) \in \gamma$ means that x is contained in the clause represented by γ and $(x, -) \in \gamma$ means that $\neg x$ is contained in the clause represented by γ.

Construction 5.1. Let ϕ be a Boolean formula in 3-CNF with the variables $X := \{x_1, \ldots, x_n\}$ and the clauses $\gamma_1, \ldots, \gamma_m \subseteq X \times \{+, -\}$. We translate ϕ into an instance of CBM that is a yes-instance if and only if ϕ is satisfiable. To this end, for every variable x_i, introduce a cycle with $4m$ edges consisting of the vertex set $V_i := \{v_i^j : 1 \leq j \leq 4m\}$ and the edge set $E_i := \{e_i^k := \{v_i^k, v_i^{k+1}\} \subseteq V_i\} \cup \{e_i^{4m} := \{v_i^1, v_i^{4m}\}\}$. Let $G := (\bigcup_{i=1}^n V_i, \bigcup_{i=1}^n E_i)$, and let $\omega(e) := 0, e \in E_i$ for any $1 \leq i \leq n$, and define $\omega_{\max} := 1$. To construct an instance of CBM, it remains to find a suitable partition of the vertices of G and a join set.

Inductively define the vertex partition P_m of $V(G)$ and the join set J_m as follows: Let $J_0 = \emptyset$, and let $P_0 := \emptyset$. For every clause γ_j, introduce the cell

$$C_j := \{v_i^{4j-1} : (x_i, +) \in \gamma_j \vee (x_i, -) \in \gamma_j\} \cup \{v_i^{4j-2} : (x_i, +) \in \gamma_j\} \cup \{v_i^{4j} : (x_i, -) \in \gamma_j\}.$$

Define $P_j := P_{j-1} \cup \{C_j\}$ and $J_j := J_{j-1} \cup \{\{0, j\}\}$.

Finally, define $C_0 := V(G) \setminus (\bigcup_{j=1}^m C_j)$. The graph G, the weight function ω, the vertex partition $P_m \cup \{C_0\}$ and the join set J_m constitute an instance of CBM.

Example 5.1. Figure 6 shows an instance of CBM produced from the formula $\phi := (\neg x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2)$ by Construction 5.1. For simplicity, we chose a formula in 2-conjunctive normal form. The instance comprises the graph G that consists of

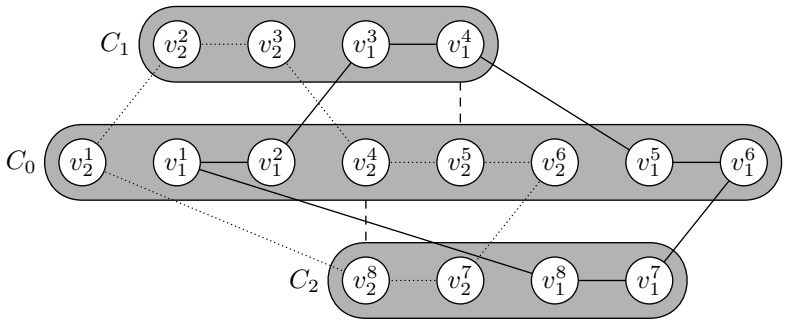


Figure 6: Example of Construction 5.1 explained in Example 5.1.

two directed cycles (solid edges and dotted edges, respectively), three cells C_0, C_1, C_2 forming a partition of $V(G)$ (shaded in gray), and a join set with two joins represented by the dashed lines.

Construction 5.1 introduces the solid-edge cycle for variable x_1 and the dotted-edge cycle for variable x_2. The cycle corresponding to x_i has exactly the two perfect matchings

$$M_i^{\text{true}} := \{\{v_i^k, v_i^{k+1}\} : k \text{ odd}\} \text{ and}$$
$$M_i^{\text{false}} := \{\{v_i^k, v_i^{k+1}\} : k \text{ even}\} \cup \{\{v_i^1, v_i^8\}\}.$$

The cell C_1 models the clause $\neg x_1 \vee x_2$ and the vertices are chosen such that only edges of M_1^{false} and edges of M_2^{true} connect the cells C_0 and C_1. Analogously, only edges of M_1^{false} and edges of M_2^{false} connect the cells C_0 and C_2.

There is a correspondence between the clauses a variable x_i satisfies using a particular truth assignment and the joins that are satisfied by matching the cycle that corresponds to x_i using one of the two available matchings. For example, the variable x_1 satisfies both clauses in ϕ when assigned false and no clause when assigned true. Accordingly, the matching M_1^{false} satisfies both the joins $\{0, 1\}$, and $\{0, 2\}$ and the matching M_1^{true} satisfies no join. This holds true analogously for x_2 and thus finding a perfect conjoining matching in G is equivalent to satisfying ϕ.

Using Construction 5.1, we can prove the following theorem.

Theorem 5.1. CBM *is NP-complete, even in the unweighted case and when the input graph $G = (V \uplus W, E)$ has maximum degree two, and for every cell C_i in the given vertex partition of G it holds that $|C_i \cap V| = |C_i \cap W|$.*

Proof. CBM is contained in NP because a perfect conjoining matching of weight at most $\omega_{\max}$ is a certificate for a yes-instance.

We prove that Construction 5.1 is a polynomial-time many-one reduction from 3SAT to CBM. Notice that in instances created by Construction 5.1 any matching has weight lower than $\omega_{\max}$ and, thus, the correctness of the reduction implies that CBM is hard even without the additional weight constraint. Also, since the cells in the instances

of CBM are disjoint unions of edges, every cell in the partition P_m contains the same number of vertices from each cell of the graph bipartition.

It is easy to check that Construction 5.1 is polynomial-time computable. For the correctness, we first need the following definition: For every variable $x_i \in X$, let

$$M_i^{\text{true}} := \{e_i^k \in E_i : k \text{ odd}\} \text{ and}$$
$$M_i^{\text{false}} := E_i \setminus M_i^{\text{true}} = \{e_i^k \in E_i : k \text{ even}\}.$$

Observe that all perfect matchings in G are of the form $\bigcup_{i=1}^{n} M_i^{\nu(x_i)}$, where ν is an assignment of truth values to variables in X. We show that the matching $\bigcup_{i=1}^{n} M_i^{\nu(x_i)}$ is a conjoining matching for G with respect to the join set J_m if and only if ν satisfies ϕ. For this, it suffices to show that for every variable $x_i \in X$ it holds that

$$\{j : (x_i, +) \in \gamma_j\} = \{j : M_i^{\text{true}} \text{ satisfies the join } \{0, j\}\}, \text{ and} \tag{1}$$
$$\{j : (x_i, -) \in \gamma_j\} = \{j : M_i^{\text{false}} \text{ satisfies the join } \{0, j\}\}. \tag{2}$$

We only show that (1) holds; (2) can be proven analogously. Assume that $(x_i, +) \in \gamma_j$. By Construction 5.1 $v_i^{4j-2} \in C_j, v_i^{4j-3} \in C_0$ and thus, since

$$\{v_i^{4j-2}, v_i^{4j-3}\} = e^{4j-3} \in M_i^{\text{true}},$$

the matching M_i^{true} satisfies the join $\{0, j\}$. Now assume that $(x_i, +) \notin \gamma_j$, that is, either (1) $(x_i, \pm) \notin \gamma_j$ or (2) $(x_i, -) \in \gamma_j$. If $(x_i, \pm) \notin \gamma_j$, then V_i and C_j are disjoint and, thus, no matching in $G[V_i]$ can satisfy the join $\{0, j\}$. If $(x_i, -) \in \gamma_j$, then the only edges in E_i that can satisfy the join $\{0, j\}$ are e_i^{4j-2} and e_i^{4j}. Both edges are not in M_i^{true} and, thus, this matching cannot satisfy the join $\{0, j\}$. □

5.2. Tractability on Restricted Graph Classes

This section presents data reduction rules and employs them to give an algorithm for CBM on a restricted graph class, leading to the following theorem:

Theorem 5.2. Conjoining Bipartite Matching *can be solved in* $O(2^{j(j+1)}n + n^3)$ *time, where* j *is the size of the join set, provided that in the bipartite input graph* $G = (V_1 \uplus V_2, E)$ *each vertex in* V_1 *has degree at most two.*

We note that there is a simple procedure that transforms in polynomial time any instance of CBM into an instance where the bipartite graph has maximum degree three.[9] This motivates to consider graphs of bounded degree. Despite this small difference, to date we were not able to extend the tractability result to instances which allow vertices of degree three in both cells of the graph bipartition. This is an intriguing open question.

In this section, let $(G, \omega_{\max}, \omega, P = \{C_1, \ldots, C_c\}, J)$ be an instance of CBM, where for $G = (V_1 \uplus V_2, E)$ it holds that each vertex in V_1 has degree at most two. The following

[9] The basic idea is to recursively replace a high-degree vertex v by a three-vertex path p and shifting the incident edges of v to the initial and terminal vertices of p. The middle vertex then models that v is matched to either of the groups of its incident edges.

lemma plays a central role in the proof of Theorem 5.2. It implies that, in a yes-instance, every component of G consists of an even-length cycle with a collection of pairwise vertex-disjoint paths incident to it.

Lemma 5.1. *If G has a perfect matching, then every connected component of G contains at most one cycle as subgraph.*

For the proof, recall that a bipartite graph $G = (V_1 \uplus V_2, E)$ has a perfect matching if and only if $|V_1| = |V_2|$ and for all subsets U of V_1 it holds that $|N(U)| \geq |U|$ (Hall's condition). For a proof, see Bang-Jensen and Gutin [2], for example.

Proof of Lemma 5.1. We show that if G contains a connected component that contains two cycles c_1, c_2 as subgraphs, then G does not have a perfect matching. First assume that c_1, c_2 are vertex-disjoint. Then, there is a path p from a vertex $v \in V(c_1)$ to a vertex $w \in V(c_2)$ such that p is vertex-disjoint from c_1 and c_2 except for v, w. It is clear that $v \in V_2$ and $w \in V_2$ because they both have degree three. Consider the vertices $V_1^{\text{cp}} := (V(c_1) \cup V(p) \cup V(c_2)) \cap V_1$ and the set $V_2^{\text{cp}} := (V(c_1) \cup V(p) \cup V(c_2)) \cap V_2$. The set of vertices V_2^{cp} is the set of neighbors of vertices in V_1^{cp}, because they have degree two and thus have neighbors only within p, c_1, and c_2. It is $|V_1^{\text{cp}}| = (|E(c_1)| + |E(p)| + |E(c_2)|)/2$ since neither of these paths and cycles overlap in a vertex in V_1. However, it is $|V_2^{\text{cp}}| = |V_1^{\text{cp}}| - 1$ because c_1 and p overlap in v and c_2 and p overlap in w. This is a violation of Hall's condition and thus G does not have a perfect matching.

The case where c_1 and c_2 share vertices can be proven analogously. (Observe that then there is a subpath of c_2 that is vertex-disjoint from c_1 and contains an even number of edges.) □

We now present four polynomial-time executable data reduction rules for CBM. It is easy to verify that the first three rules are correct and can be applied exhaustively in $O(n^3)$ time, thus, we omit the corresponding proofs.

Reduction Rule 5.1 removes paths incident to the cycles of a graph G in a yes-instance. As a side-result, Reduction Rule 5.1 solves CBM in linear time on forests.

Reduction Rule 5.1. *If there is an edge $\{v, w\} \in E(G)$ such that $\deg(v) = 1$, then remove both v and w from G, and remove all joins $\{i, j\}$ from J, with $v \in C_i, w \in C_j$. Decrease ω_{max} by $\omega(\{v, w\})$.*

If exhaustively applying Reduction Rule 5.1 to G does not transform G such that each connected component is a cycle, which is checkable in linear time, then, by Lemma 5.1, G does not have a perfect matching and we can return "NO". Hence, in the following, assume that each connected component of G is a cycle. Reduction Rule 5.2 now deletes connected components that cannot satisfy joins.

Reduction Rule 5.2. *If there is a connected component D of G such that it contains no edge that could satisfy any join in J, then compute a minimum-weight perfect matching M in $G[D]$, remove D from G and decrease ω_{max} by $\omega(M)$.*

After exhaustively applying Reduction Rule 5.2, we may assume that each connected component of G contains an edge that could satisfy a join. We next present a data reduction rule that removes joins that are always satisfied. To this end, we need the following definition.

Definition 5.1. For each connected component D (that is, each cycle) in G, denote by $M_1(D)$ a minimum-weight perfect matching of D with respect to ω and denote by $M_2(D) \coloneqq E(D) \setminus M_1(D)$ the other perfect matching of D.[10] Furthermore, denote

$$\sigma_1(D) \coloneqq \{j \in J : \exists e \in M_1(D) : e \text{ satisfies } j\},$$
$$\sigma_2(D) \coloneqq \{j \in J : \exists e \in M_2(D) : e \text{ satisfies } j\},$$

and the *signature* $\sigma(D)$ of D as $\{\sigma_1(D), \sigma_2(D)\}$.

Reduction Rule 5.3. *Let D be a connected component of G. If there is a join $j \in \sigma_1(D) \cap \sigma_2(D)$, then remove j from J.*

A final data reduction rule removes connected components that satisfy the same joins.

Reduction Rule 5.4. *Let $S = \{D_1, \ldots, D_j\}$ be a maximal set of connected components of G such that $\sigma(D_1) = \ldots = \sigma(D_j)$ and $j \geq 2$. Let $M_1^* = \bigcup_{k=1}^{j} M_1(D_k)$, let $D_l \in S$ such that $\omega(M_2(D_l)) - \omega(M_1(D_l))$ is minimum, and let $M_1^{\sim} = M_1^* \setminus M_1(D_l)$.*

(i) If the matching M_1^* is conjoining for the join set $\sigma_1(D_1) \cup \sigma_2(D_1)$, then remove each component in S from G, remove each join in $\sigma_1(D_1) \cup \sigma_2(D_1)$ from the join set J, and reduce ω_{max} by $\omega(M_1^*)$.

(ii) If the matching M_1^* is not conjoining for the join set $\sigma_1(D_1) \cup \sigma_2(D_1)$, then remove each component in $S \setminus \{D_l\}$ from G, remove any join in $\sigma_1(D_1)$ from the join set J, and reduce ω_{max} by $\omega(M_1^{\sim})$.

In either case, update the partition P accordingly.

Lemma 5.2. *Reduction Rule 5.4 is correct.*

Proof. Let $G = (V_1 \uplus V_2, E)$ be a graph with maximum degree two, let $\omega : E \to [0, \omega_{\max}] \cup \{\infty\}$ be a weight function, let $P = \{C_1, \ldots, C_c\}$ be a vertex partition of G and let J be a join set with respect to G and P. The objects G, ω, $\omega_{\max}$, P, and J constitute an instance I of CBM. Furthermore, let the graph G', the weight function ω, the maximum weight $\omega'_{\max}$, the vertex partition P', and the join set J' with respect to G' and P' constitute the instance I' that is obtained from I by applying Reduction Rule 5.4 with the set $S = \{D_1, \ldots, D_j\}$ as defined there.

Let M be a perfect J-conjoining matching for G with $\omega(M) \leq \omega_{\max}$ and assume that the matching $M_1^* = \bigcup_{k=1}^{j} M_1(D_k)$ is conjoining for the join set $\sigma_1(D_1) \cup \sigma_2(D_1)$. Then either $M_1^* \subseteq M$, or we can obtain another perfect J-conjoining matching with weight at most $\omega(M)$ that satisfies this property. Without loss of generality assume that $M_1^* \subseteq M$. Then $M \setminus M_1^*$ is a perfect J'-conjoining matching for G' of weight $\omega(M) - \omega(M_1^*) \leq \omega'_{\max}$.

Now assume that M_1^* is not conjoining for the join set $\sigma_1(D_1) \cup \sigma_2(D_1)$. Then either

(1) $M_1^* \subseteq M$ or

(2) there is an integer n such that $M_2(D_n) \subseteq M$.

We first show that, in case (2), we may assume without loss of generality that n is unique and that $n = l$ as in Reduction Rule 5.4. Otherwise we can find another perfect J-conjoining matching with weight at most $\omega(M)$ that satisfies this property: Since M_1^* is

[10] Note that in bipartite graphs every cycle is of even length.

not conjoining for the join set $\sigma_1(D_1) \cup \sigma_2(D_1)$, it holds that

$$\sigma_1(D_1) = \ldots = \sigma_1(D_j), \qquad \text{and} \qquad \sigma_2(D_1) = \ldots = \sigma_2(D_j),$$

because all signatures of the components in S are equal by prerequisite of Reduction Rule 5.4. If n is not unique, there are n, m such that $M_2(D_n), M_2(D_m) \subseteq M$. However, by definition $\omega(M_1(A)) \leq \omega(M_2(A))$ and if we substitute $M_1(D_m)$ for $M_2(D_m)$ in M, the resulting matching has at most the same weight and is still J-conjoining because $\sigma_2(D_n) = \sigma_2(D_m)$. Hence we can assume that n is unique. We can also assume that $n = l$ because by definition of l

$$\omega(M_2(D_l)) - \omega(M_1(D_l)) \leq \omega(M_2(D_n)) - \omega(M_1(D_n))$$

and thus we can substitute $M_1(D_n)$ for $M_2(D_n)$ and $M_2(D_l)$ for $M_1(D_l)$ in the matching M to obtain a perfect J-conjoining matching of at most the same weight. Consider the matching $M_1^\sim = \bigcup_{1 \leq k \leq j, k \neq l} M_1(D_k)$. Both in case (1) and in case (2), when assuming that $n = l$ is unique, $M \setminus M_1^\sim$ is a perfect J'-conjoining matching for G' of weight $\omega(M) - \omega(M_1^\sim) \leq \omega'_{\max}$.

We now have that if I is a yes instance then I' is a yes instance. For the other direction, assume that M' is a perfect J'-conjoining matching for G' of weight $\omega(M') \leq \omega'_{\max}$. Assume that each component in S of G has been removed in G' by Reduction Rule 5.4. Then the matching $M' \cup M_1^*$ for G is perfect, J-conjoining and of weight $\omega(M) + \omega(M_1^*) \leq \omega_{\max}$. Now assume only the component D_l of the components in S is still present in G'. Then, the matching $M \cup M_1^\sim$ is a perfect J-conjoining matching for G of weight $\omega(M) + \omega(M_1^\sim) \leq \omega_{\max}$. □

Lemma 5.3. *Reduction Rule 5.4 can be applied exhaustively in* $O(n^3)$ *time.*

Proof. To apply Reduction Rule 5.4 once, we can first search for a set of components S as defined there by first finding all connected components in linear time. Then we find out the signature of each connected component. For this, we first compute a minimum-weight perfect matching for every connected component in overall $O(m)$ time by simply iterating over the edges in each component, alternatingly summing up the edge weights and choosing the lower one of the two values. We annotate every edge with whether it is contained in the minimum-weight matching or not and which join it satisfies, if any, in $O(m^2)$ time. We then iterate over every edge and add the information saved in the annotation to the signature of the connected component it is contained in.

Having computed the signatures, we create a map in $O(n \log(n))$ time that maps every signature present to the list of connected components that have this signature. We then simply iterate over every list present in the map to obtain a maximal list of components that have the same signature or decide that there is no such list with at least two elements. This is possible in $O(n)$ time.

The removal of the connected components and joins, the update of $\omega_{\max}$ and the partition P is then possible in linear time, because the matchings for each component have already been computed and thus the overall running time is $O(m^2 + n \log n)$. Observe that in graphs with only vertices of degree two $m \in O(n)$ and thus we can derive a running time bound in $O(n^2)$.

In any application, either no set S is found and thus the procedure terminates, or at least 4 vertices are deleted—this is the minimum size of a connected component. Hence the procedure can be applied at most n times and exhaustively applying Reduction Rule 5.4 takes $O(n^3)$ time. □

Now, mainly using Reduction Rule 5.4, we are able to prove Theorem 5.2.

Proof of Theorem 5.2. An algorithm to solve CBM may first exhaustively apply Reduction Rule 5.1 through Reduction Rule 5.4. Then, since Reduction Rule 5.4 is not applicable anymore, it follows that for every signature there is at most one connected component in the reduced instance. If j is the size of the join-set, then there are at most 2^{j+1} signatures, and thus we may employ the following search tree algorithm to achieve the claimed running time of $O(2^{j(j+1))}n + n^3)$: In $O(n)$ time, choose an arbitrary join $k \in J$ that is not satisfied yet, and branch into all possibilities of choosing one of the at most 2^{j+1} connected components of the graph that can satisfy k. Match the vertices in the chosen connected component such that it satisfies k and recurse until all joins are satisfied. □

Analyzing the pre-images that lead to tractable instances of CBM under the reductions we gave in Section 3 and Section 4, Theorem 5.2 can be translated to a tractability result for EE. A similar tractability result can also be shown for Rural Postman.

Corollary 5.1. *Let the graph G and the weight function ω constitute an instance I_{EE} of* Eulerian Extension. *Let c be the number of connected components in G.*

(i) If every path or cycle in the set of allowed arcs with respect to ω has length at most one,
(ii) if G contains only vertices with balance between -1 and 1,
(iii) if every vertex in I_G^+ (every vertex in I_G^-) has only outgoing allowed arcs (incoming allowed arcs), and
(iv) if in every connected component C of G, either all vertices in $I_G^+ \cap C$ or in $I_G^- \cap C$ have at most two incident allowed arcs,

then it is decidable in $O(2^{c(c+3\log c+1)}(n^4 + m))$ time whether I_{EE} is a yes-instance.

6. Conclusion

The most important remaining open question is to determine whether Rural Postman is fixed-parameter tractable with respect to the number of weakly connected components of the graph induced by the required arcs. This question also extends to the presumably harder undirected case of Rural Postman. The newly introduced Conjoining Bipartite Matching (CBM) problem might also be useful in answering this question. Additionally, it may enable us to spot new, computationally feasible special cases of Rural Postman and Eulerian Extension. The development of polynomial-time approximation algorithms for CBM or the investigation of other (structural) parameterizations for CBM seem worthwhile challenges as well. Finally, we remark that previous work [10, 22] also left open a number of interesting open problems referring to variants of Eulerian Extension. Due to the practical relevance of the considered problems, our work is also meant to further stimulate more research on these challenging combinatorial problems.

Acknowledgement. We thank Greg N. Frederickson for scanning relevant pages of his PhD-thesis and making them available to us.

References

[1] A. A. Assad and B. L. Golden. Arc routing methods and applications. In *Network Routing*, volume 8 of *Handbooks in Operations Research and Management Science*, pages 375–483. Elsevier B. V., 1995.

[2] J. Bang-Jensen and G. Gutin. *Digraphs: Theory, Algorithms and Applications*. Springer, second edition, 2008.

[3] E. Benavent, A. Corberán, E. Piñana, I. Plana, and J. M. Sanchis. New heuristic algorithms for the windy rural postman problem. *Comput. Oper. Res.*, 32(12): 3111–3128, 2005.

[4] F. T. Boesch, C. Suffel, and R. Tindell. The spanning subgraphs of Eulerian graphs. *J. Graph Theory*, 1:79–84, 1977.

[5] E. A. Cabral, M. Gendreau, G. Ghiani, and G. Laporte. Solving the hierarchical chinese postman problem as a rural postman problem. *European J. Oper. Res.*, 155(1):44–50, 2004.

[6] L. Cai and B. Yang. Parameterized complexity of even/odd subgraph problems. *J. Discrete Algorithms*, 9:231–240, 2011.

[7] N. Christofides, V. Campos, A. Corberán, and E. Mota. An algorithm for the rural postman problem on a directed graph. *Mathematical Programming Study*, 26: 155–166, 1986.

[8] A. Corberán and J. M. Sanchis. A polyhedral approach to the rural postman problem. Technical report, Departamento de Estadística e Investigación Operativa, Universidad de Valencia, Spain, 1991.

[9] M. Cygan, D. Marx, M. Pilipczuk, M. Pilipczuk, and I. Schlotter. Parameterized complexity of Eulerian deletion problems. In *Proc. 37th WG*, volume 6986 of *LNCS*, pages 131–142. Springer, 2011.

[10] F. Dorn, H. Moser, R. Niedermeier, and M. Weller. Efficient algorithms for Eulerian extension. In *Proc. 36th WG*, volume 6410 of *LNCS*, pages 100–111. Springer, 2010. Full version submitted to *SIAM Journal on Discrete Mathematics*.

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

[12] M. Dror. *Arc Routing: Theory, Solutions, and Applications*. Kluwer Academic Publishers, 2000.

[13] J. Edmonds and E. L. Johnson. Matching, euler tours and the chinese postman. *Math. Program.*, 5:88–124, 1973.

[14] H. A. Eiselt, M. Gendreau, and G. Laporte. Arc routing problems, part II: The rural postman problem. *Oper. Res.*, 43(3):399–414, 1995.

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

[16] F. V. Fomin and P. A. Golovach. Parameterized Complexity of Connected Even/Odd Subgraph Problems. In *Proc. 29th STACS*, volume 14, pages 432–440. LZI Dagstuhl, Germany, 2012.

[17] G. N. Frederickson. *Approximation Algorithms for NP-hard Routing Problems*. PhD thesis, Faculty of the Graduate School of the University of Maryland, 1977.

[18] G. N. Frederickson. Approximation algorithms for some postman problems. *J. ACM*, 26(3):538–554, 1979.

[19] G. Ghiani and G. Laporte. A branch-and-cut algorithm for the undirected rural postman problem. *Math. Program.*, 87:467–481, 2000.

[20] G. Groves and J. Van Vuuren. Efficient heuristics for the rural postman problem. *ORiON*, 21(1):33–51, 2005.

[21] J. Guo, F. Hüffner, and R. Niedermeier. A structural view on parameterizing problems: Distance from triviality. In *Proc. 1st IWPEC*, volume 3162 of *LNCS*, pages 162–173. Springer, 2004.

[22] W. Höhn, T. Jacobs, and N. Megow. On Eulerian extensions and their application to no-wait flowshop scheduling. *J. Sched.*, 2011. Available electronically.

[23] J. K. Lenstra and A. H. G. R. Kan. On general routing problems. *Networks*, 6(3): 273–280, 1976.

[24] L. Lesniak and O. R. Oellermann. An Eulerian exposition. *J. Graph Theory*, 10 (3):277–297, 1986.

[25] A. N. Letchford. *Polyhedral Results for some Constrained Arc-Routing Problems*. PhD thesis, Lancaster University, Lancaster, United Kingdom, 1996.

[26] K. Mei-Ko. Graphic programming using odd or even points. *Chinese Math.*, 1: 273–277, 1962.

[27] R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, 2006.

[28] C. S. Orloff. A fundamental problem in vehicle routing. *Networks*, 4(1):35–64, 1974.

[29] C. S. Orloff. On general routing problems: Comments. *Networks*, 6(3):281–284, 1976.

[30] N. Perrier, A. Langevin, and J. F. Campbell. A survey of models and algorithms for winter road maintenance. Part IV: Vehicle routing and fleet sizing for plowing and snow disposal. *Comput. Oper. Res.*, 34(1):258–294, 2007.

[31] N. Perrier, A. Langevin, and C.-A. Amaya. Vehicle routing for urban snow plowing operations. *Transport. Sci.*, 42(1):44–56, 2008.

[32] M. Pérez-Delgado. A solution to the rural postman problem based on artificial ant colonies. In *Current Topics in Artificial Intelligence*, volume 4788 of *LNCS*, pages 220–228. Springer, 2007.

[33] M. Sorge. On making directed graphs Eulerian. Diplomarbeit, Institut für Informatik, Friedrich-Schiller-Universität Jena, 2011. Available electronically. arXiv:1101.4283 [cs.DM].

[34] M. Sorge, R. van Bevern, R. Niedermeier, and M. Weller. From few components to an Eulerian graph by adding arcs. In *Proc. 37th WG*, volume 6986 of *LNCS*, pages 307–319. Springer, 2011.

[35] M. Sorge, R. van Bevern, R. Niedermeier, and M. Weller. A new view on rural postman based on Eulerian extension and matching. In *Proc. 22nd IWOCA*, volume 7056 of *LNCS*, pages 310–322. Springer, 2011.