From Few Components to an Eulerian Graph by Adding Arcs

Manuel Sorge*, René van Bevern**, Rolf Niedermeier, and Mathias Weller***

Institut für Softwaretechnik und Theoretische Informatik, TU Berlin, Berlin, Germany
{manuel.sorge, rene.vanbevern, rolf.niedermeier, mathias.weller}@tu-berlin.de

Abstract Eulerian Extension (EE) is the problem to make an arc-weighted directed multigraph Eulerian by adding arcs of minimum total cost. EE is NP-hard and has been shown fixed-parameter tractable with respect to the number of arc additions. Complementing this result, on the way to answering an open question, we show that EE is fixed-parameter tractable with respect to the combined parameter "number of connected components in the underlying undirected multigraph" and "sum of $\mathrm{indeg}(v) - \mathrm{outdeg}(v)$ over all vertices v in the input multigraph where this value is positive." Moreover, we show that EE is unlikely to admit a polynomial-size problem kernel for this parameter combination and for the parameter "number of arc additions".

1 Introduction

A directed (multi-)graph G is called *Eulerian* if it contains a tour that traverses every arc in G exactly once. We study the following NP-complete decision problem:

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

Herein, an *Eulerian extension* is a multiset E over $V \times V$ such that $G' = (V, A \cup E)$ is Eulerian. In the weight of an Eulerian extension, multiple arcs are counted multiple times, according to their multiplicity. Recently, there has been renewed interest in Eulerian Extension for at least two reasons. First, there are interesting applications (of special cases) of Eulerian Extension for sequencing problems [10]. Second, it has been pointed out that Eulerian Extension is "parameterized equivalent" to Rural Postman [4], a famous arc routing problem in combinatorial optimization [6, 12]. The main focus of this paper is on assessing the parameterized computational complexity of EE with respect to the

* Supported by the DFG, project AREG, NI 369/9 and project PABI, NI 369/7.
** Supported by the DFG, project AREG, NI 369/9.
*** Supported by the DFG, project DARE, NI 369/11.

parameter c denoting the number of connected components of the underlying undirected multigraph.

We are aware of only two papers explicitly dedicated to studying the parameterized complexity of Eulerian Extension [4, 17]. Dorn et al. [4] studied the "standard parameterization" by the number k of extension arcs and their main result was to show that Eulerian Extension is fixed-parameter tractable with respect to k. Motivated by early work of Orloff [14, 15] and Frederickson [8, 9], we complement the previous considerations and start a deeper study of Eulerian Extension parameterized by the component parameter c. Frederickson [8, 9] showed that Eulerian Extension is polynomial-time solvable for constant c. However, in his algorithm c influences the degree of the polynomial and so this only shows containment in the parameterized complexity class XP; it does not yield fixed-parameter tractability with respect to parameter c. Already in the 1970's Lenstra and Kan [12] and Orloff [15] pointed out the importance of the "complexity parameter" c. Indeed, to date, it is open whether Eulerian Extension can be solved in (fixed-parameter tractable) time $f(c) \cdot n^{O(1)}$ for an n-vertex multigraph, f being an arbitrary computable function exclusively depending on c. In companion work [17], we related Eulerian Extension to a matching variant and derived fixed-parameter tractability with respect to c on some special graph classes.

Our Results. We make some partial progress on resolving the complexity question for Eulerian Extension with respect to the parameter c. More specifically, we show that Eulerian Extension is fixed-parameter tractable with respect to the combined parameter (b, c), where b denotes the sum of all positive values $\text{indeg}(v) - \text{outdeg}(v)$ over all vertices in the multigraph. See Orloff [15] for an early indication towards the relevance of this combined parameter. Notably, both b and c are upper-bounded by k and should typically be significantly smaller than k for most input multigraphs. In addition, we show that there is no polynomial-size problem kernel for the single parameters b, c, or k, or the combined parameter (b, c), unless coNP $\subseteq$ NP/poly. To this end, we introduce the NP-hard Switch Set Cover, a combinatorial problem of potentially independent interest. Due to the lack of space, many details are deferred to a full version of the paper.[1]

Preliminaries. We consider directed *multigraphs* $G = (V, A)$, where $V(G) := V$ is the set of vertices and $A(G) := A$ is the multiset of arcs. We use $n := |V|$ and $m := |A|$. A *trail* W in G is a sequence of arcs in G such that each arc ends in the same vertex as the next arc starts in and such that no arc is used more often than it is present in G. We use $V(W)$ and $A(W)$ to refer to the set of vertices in which arcs of W start or end, and the multiset of arcs of W, respectively. A *path* in the multigraph G is a trail that traverses every vertex of G at most once. A closed trail that traverses its initial and terminal vertex exactly twice and every other vertex of G at most once is called a *cycle*. A directed multigraph G is said to be *weakly connected* if every pair of vertices $u, v \in V(G)$ is *weakly connected*, that is, there is a path with the endpoints u, v in the underlying undirected multigraph of G. For brevity, we also write *connected* instead of weakly connected.

[1] Further details can also be found in the first author's diploma thesis [16].

A *connected component* is a maximal vertex set C such that $G[C]$ is connected. We use $\text{balance}(v) := \text{indeg}(v) - \text{outdeg}(v)$ 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 $\text{balance}(v) > 0$ and $\text{balance}(v) < 0$, respectively. A vertex v is *balanced* if $\text{balance}(v) = 0$.

We use the following characterization of Eulerian multigraphs, due to Euler: A multigraph is Eulerian if and only if all arcs are contained in the same connected component and all vertices are balanced.

Our results are in the context of parameterized complexity, which is a two-dimensional framework for studying computational complexity [5, 7, 13]. 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 depending only on k. For a language $L \subseteq \Sigma^* \times \mathbb{N}$, a *reduction to a problem kernel* is a function that takes as input an instance (x, k) and, in time polynomial in $|x| + k$, outputs an instance (x', k') such that $(x', k') \in L \Leftrightarrow (x, k) \in L$, $|x'| \leq f(k)$, and $k \leq g(k)$. Here, f and g are computable functions solely depending on k; f is called the *size* of the problem kernel. 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 f such that $(x, k) \in L \Leftrightarrow (x', k') \in L'$, where $(x', k') := f(x, k)$, $k' \leq p(k)$, and p is a polynomial depending only on k. If such a reduction exists, we write $L \leq_{\text{m}}^{\text{PPP}} L'$.

2 Limiting Imbalance Helps

The fixed-parameter tractability of Eulerian Extension with respect to the number c of connected components in the input multigraph is a open question that arose implicitly in work of Frederickson [8, 9]. While we cannot answer this question, this section presents a fixed-parameter algorithm for EE that, *additionally* to the parameter c, uses the sum b of all positive balances of vertices as parameter. An early indication that both parameters influence the complexity of EE was given by Orloff [15].

Theorem 1. Eulerian Extension *is solvable in* $\mathrm{O}(4^{c \log(bc^2)} n^2 (b^2 + n \log n) + n^2 m)$ *time. Here, c is the number of connected components in the input multigraph, and b is the sum of all positive balances of vertices in the input multigraph.*

To prove Theorem 1, we consider a restricted version of EE that takes as input, additionally to a regular EE-instance, an "advice" that determines how components in the input multigraph are to be connected. Then, we will see that the number of ways to connect two given components is upper-bounded in terms of b.

More details follow. Let $G = (V, A)$ be a directed multigraph. By $\mathbb{C}_G$ we denote the *component graph*, which is a clique whose vertices one-to-one correspond to the weakly connected components of G. A *hint* for G is an undirected path or cycle t of length at least one in the component graph $\mathbb{C}_G$ together with the information whether t shall form a cycle or a path in an Eulerian extension of G.[2]

[2] The extra information is necessary because a hint to a path may be a cycle in $\mathbb{C}_G$.

We call the corresponding hints *cycle hints* and *path hints*, respectively. We say a set of hints P is an *advice* to the multigraph G if the hints are edge-disjoint.[3] For a trail t in the graph $(V, V \times V)$, we define $\mathbb{C}_G(t)$ as the trail in $\mathbb{C}_G$ that is obtained by making t undirected and, for every connected component C of G, substituting every maximum length subtrail t' of t with $V(t') \subseteq C$ by the vertex in $\mathbb{C}_G$ corresponding to C. We say that a path p in the 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. We say that a cycle c in the graph $(V, V \times V)$ realizes a cycle hint h if $\mathbb{C}_G(c) = h$. We say that an Eulerian extension E *heeds the advice* P if it can be decomposed into a number of paths and cycles that realize all hints in P. An advice P is *connecting* if the hints in P connect every pair of vertices in $\mathbb{C}_G$. Now consider the following restricted version of EE:

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

Section 2.1 first shows how to solve EE with help of an algorithm for EEA:

Lemma 1. EE *can be solved by solving* $\mathrm{O}(c^{4c-2})$ *instances of* EEA*, where each instance is computable in* $O(c^3 + n + m)$ *time.*

Then Section 2.2 shows an algorithm that solves EEA in the following time:

Proposition 1. EEA *is solvable in* $\mathrm{O}(4^{c \log b} n^2 (b^2 + n \log n) + n^2 m)$ *time.*

Then, Theorem 1 follows by combining Lemma 1 and Proposition 1. In order to prove these, we first present two transformations that, applied to the input multigraph, allow us to assume that each vertex has balance between -1 and 1 and to assume that weight functions abide the triangle inequality. For each given transformation, we show that it is *correct*, that is, it transforms yes-instances and only yes-instances to yes-instances.

Transformation 1 (Splitting Vertices). Let the multigraph $G = (V, A)$, the weight function ω, and the maximum weight $\omega_{\max}$ constitute an instance of EE. Compute a new instance as follows: Search for a vertex v with $|\operatorname{balance}(v)| > 1$, and introduce a new vertex u. If $\operatorname{balance}(v) > 0$, choose an arbitrary arc (w, v), delete it, and add the arc (w, u). Proceed analogously if $\operatorname{balance}(v) < 0$. Add the arcs $(u, v), (v, u)$. Finally, define a new weight function ω' for each pair of vertices $x, y \in V$ as follows.

$$\omega'(x, y) = \begin{cases} \infty & \text{if } (x = u \wedge y = v) \vee (x = v \wedge y = u), \\ \omega(v, y) & \text{if } x = u, \\ \omega(x, v) & \text{if } y = u, \\ \omega(x, y) & \text{otherwise} \end{cases}$$

[3] Note the difference between advice in our sense and advice in computational complexity theory. There, a piece of advice applies to every instance of a specific length.

Lemma 2. *Transformation 1 is correct. It can exhaustively be applied in* $\mathrm{O}(n^2m)$ *time. The resulting instance only contains vertices* v *with* $|\operatorname{balance}(v)| \leq 1$.

A further preprocessing routine allows us to assume that weight functions abide the triangle inequality.

Transformation 2 (Shortest-Path Preprocessing). For an input instance of EE consisting of the directed multigraph $G = (V, A)$, the weight function ω and the maximum weight $\omega_{\max}$, derive a new instance by computing a new weight function ω', where for each $u, v \in V$, $\omega'(u, v)$ is the weight of a shortest path from u to v in the graph $(V, V \times V)$.

Lemma 3. *Transformation 2 is correct and can be applied in* $\mathrm{O}(n^3)$ *time.*

Observe that the number of components and the sum of all positive balances of vertices in an instance of EE are invariant under Transformation 1 and Transformation 2. In the following we assume all instances of EE and EEA to be exhaustively preprocessed using Transformation 1 and Transformation 2.

2.1 Generating Advice for Eulerian Extension

This section shows how to generate advice for EE in order to solve EE with the help of an algorithm for EEA, thus proving Lemma 1. To prove Theorem 1, it then remains to prove Proposition 1, that is, to present an algorithm for EEA; this is done in Section 2.2. To solve EE using an algorithm for EEA, we simply try every minimal connecting advice and solve the resulting instances of EEA. Exploiting the structure obtained by Transformations 1 and 2, one can show that we only have to try very restricted forms of advice:

Lemma 4. *Let* G *be a directed multigraph 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* $P = \{h_1, \ldots, h_i\}$ *such that*

(i) E *heeds* P *and*

(ii) the graph defined by the union of all trails $h_1, \ldots, h_i$ *without their initial vertices does not contain a cycle.*

Using this restriction, we can enumerate all forests in $\mathbb{C}_G$ and try all possibilities to extend them to an advice. Thus, we can prove Lemma 1.

Proof (Lemma 1). We give an algorithm that decides EE by solving $\mathrm{O}(4^{c \log(c^2)})$ instances of EEA, each of which can be generated in $\mathrm{O}(c^3 + n + m)$ time. Let the directed multigraph $G = (V, A)$ with c connected components and the weight function $\omega : V \times V \to [0, \omega_{\max}] \cup \{\infty\}$ constitute an instance of EE.

We simply generate all possible pieces of advice and solve the so-obtained EEA-instances. If one of these instances is a yes-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 4.

Concerning the generation of the advices, by Lemma 4 we may assume that the hints without their initial vertices form a forest in $\mathbb{C}_G$. Thus, we may simply enumerate all forests contained in $\mathbb{C}_G$, partition their edges into at most c hints and try all possibilities to reinsert the initial vertices back onto the hints. To enumerate all forests, we first partition the vertices into at most c cells (there are at most c^c such partitions), then enumerate all spanning trees in each cell (in each cell there are at most c^{c-2} spanning trees [2] that can be numerated in $\mathrm{O}(c^{c-2}+c^2)$ time [11]).

Hence, in total, $\mathrm{O}(c^c c^{c-2})$ forests are computed. We partition the edges of each forest into at most c hints (there are at most c^c partitions for each forest), extend every hint by adding an initial vertex (for each of the c hints, there are c possibilities, yielding c^c possibilities in total) and check whether this yields a valid advice—that is, we check whether the hints are paths or cycles and whether the advice is connecting. In total, we generated $\mathrm{O}(c^c c^{c-2} c^c c^c) \subseteq \mathrm{O}(c^{4c}) = \mathrm{O}(4^{c \log(c^2)})$ advices. The validity check for each advice can be carried out in $\mathrm{O}(c^3)$ time. Hence, each instance can be generated in $\mathrm{O}(c^3+n+m)$ time. □

2.2 Solving Eulerian Extension with Advice

Having shown how EE can be solved using EEA, it remains to present an algorithm for EEA to solve EE. To this end, this section proves Proposition 1. This will conclude the proof of Theorem 1. To obtain an algorithm for EEA, we use the fact that EE on connected multigraphs is solvable in $\mathrm{O}(n^3 \log n)$ time [4]. Hence, given an instance of EEA, we can solve it by realizing all hints given in the given minimal connecting advice and then solving EE on the resulting connected multigraph. In this approach, the parameter b helps to bound the number of possible ways we have to try to realize each hint. An algorithm that achieves the running time given in Proposition 1 can simply try each combination of optimal realizations of each hint in the given advice and then solve the resulting instance comprising a connected multigraph via the polynomial-time algorithm of Dorn et al. [4]. We denote a call to this algorithm by solve_connected(G, ω), where G is a connected multigraph and $\omega : V \times V \to [0, \omega_{\max}] \cup \{\infty\}$ is a weight function.

Realizing Hints. First, we show how to realize a given path hint using a path between two given vertices. Then, we can try all possible initial and terminal vertices of such a path in order to optimally realize a path hint. For a directed multigraph $G = (V, A)$ and a weight function $\omega : V \times V \to [0, \omega_{\max}] \cup \{\infty\}$, let p be a path in $\mathbb{C}_G$, 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. Define minpath(G, ω, p, u, v) as a shortest path s from u to v in the complete graph $(V, V \times V)$ such that $\mathbb{C}_G(s) = p$.

In the following, we show that the minpath function indeed yields realizations of hints that can be assumed to be part of an optimal Eulerian extension and how it can be computed in $\mathrm{O}(n^2)$ time.

Observation 1. *Let E be an Eulerian extension for the multigraph G that heeds the advice P, let P contain a path-hint h, and let ω be a weight function $V \times V \to$*

Algorithm SolveEEA: Solving EEA.

Input: A directed multigraph $G = (V, A)$, a weight function $\omega : V \times V \to [0, \omega_{\max}] \cup \{\infty\}$, a cycle-less advice P, and an arc-set E.
Output: A minimum-weight Eulerian extension for G that heeds the advice P.

```
1  if P = ∅ then return E ∪ solve_connected(G, ω);
2  else
3      h ← a hint in P;
4      C_A ← connected component of G corresponding to the initial vertex of h;
5      C_Ω ← connected component of G corresponding to the terminal vertex of h;
6      MinEE ← ∅;
7      found_solution ← false;
8      for (u, v) ∈ I_G^+ × I_G^- such that u ∈ C_A ∧ v ∈ C_Ω or vice versa do
9          p ← minpath(G, ω, h, u, v);
10         ActEE ← SolveEEA(G + p, ω, P \ {h}, E ∪ A(p));
11         if (ω(ActEE) < ω(MinEE)) ∨ (found_solution = false) then
12             found_solution ← true;
13             MinEE ← ActEE;
14     return MinEE;
```

$[0, \omega_{\max}] \cup \{\infty\}$. *Then, there is an Eulerian extension* E' *such that* E' *heeds the advice* P, $\omega(E') \leq \omega(E)$, *and* $A(\text{minpath}(G, \omega, h, u, v)) \subseteq E'$. *Here,* u, v *are vertices contained in the connected components of* G *that correspond to the initial and terminal vertices of* h, *respectively.*

Exploiting the structure obtained by Transformations 1 and 2, we can show:

Lemma 5. $\text{minpath}(G, \omega, p, u, v)$ *is computable in* $\mathrm{O}(n^2)$ *time.*

Having shown how to realize path hints, we can show how to realize all cycle hints in an advice P in $\mathrm{O}(|P|n^3)$ time.

Lemma 6. *For a given directed multigraph* G, *a weight function* $\omega : V \times V \to [0, \omega_{\max}] \cup \{\infty\}$, *and an advice* P, *realizations of all cycle hints in* P *are computable in* $\mathrm{O}(|P|n^3)$ *time.*

Solution Algorithm. We now give an algorithm for Eulerian Extension with Minimal Connecting Advice, thus proving Proposition 1 and, therefore, proving Theorem 1.

Proof (Proposition 1). Given an instance of EEA, we first compute realizations of all cycle hints in the advice P in $\mathrm{O}(|P|n^3)$ time (see Lemma 6), add them to G and remove all cycle hints from P. Hence, in the following, we assume that P only contains path hints. Then, we apply Algorithm SolveEEA that solves instances of EEA whose advice does not contain cycle hints. We first look at the correctness of Algorithm SolveEEA and then analyze the overall running time.

Consider the return value E' of Algorithm SolveEEA when called with an initially empty arc set E and an instance of EEA consisting of the multigraph G, the weight function ω, and minimal connecting advice P without cycle hints. For every hint in P there is realization in E', that is, E' connects all connected components of G. Because of the call to solve_connected(), the set E' also makes every vertex in G balanced. Hence, E' is an Eulerian extension for G that heeds P. Also, E' is of minimum weight among all Eulerian extensions for G that heed the advice P. This is because, first, the solution of solve_connected() is weight-minimal and, second, because, by Observation 1, we may assume that in a minimum-weight Eulerian extension all path hints h are realized by $\text{minpath}(G, \omega, h, u, v)$ for appropriate vertices u, v.

Concerning the running time of the overall procedure, we have to preprocess the input instance using Transformation 1 and Transformation 2 (recall that we assume that all instances are preprocessed accordingly). By Lemmas 2 and 3 this takes $\mathrm{O}(n^3 + n^2 m)$ time. Next, all cycle hints are realized. By Lemma 6, this is possible in $\mathrm{O}(|P| n^3)$ time. Finally, we apply Algorithm SolveEEA: Obviously its recursion depth is at most $|P|$. Because of $b \geq |I_G^+| = |I_G^-|$, every call of Algorithm SolveEEA yields at most b^2 recursive calls. This means that the sum of all calls is $b^{2|P|}$. The running time of one call is dominated by either the computation of b^2 minpath instances which takes $\mathrm{O}(b^2 n^2)$ time (Lemma 5) or the computation of solve_connected() which takes $\mathrm{O}(n^3 \log n)$ time [4]. Thus, Algorithm SolveEEA can be executed in $\mathrm{O}(b^{2|P|}(b^2 n^2 + n^3 \log n)) = \mathrm{O}(2^{2|P| \log(b)} n^2 (b^2 + n \log n))$ time. Since P is a *minimal* connecting advice, we have $|P| \leq c$, and thus the overall procedure runs in $\mathrm{O}(2^{2c \log(b)} n^2 (b^2 + n \log n) + cn^3 + n^2 m) = \mathrm{O}(4^{c \log(b)} n^2 (b^2 + n \log n) + n^2 m)$ time. □

3 Non-Existence of Polynomial-Size Problem Kernels

In this section, we prove the following theorem.

Theorem 2. Eulerian Extension *does not admit a polynomial-size problem kernel with respect to the parameters*

(i) minimum number k of arcs in an Eulerian extension E with $\omega(E) \leq \omega_{\max}$,
(ii) sum b of positive balances,
(iii) number c of connected components in the input multigraph, or
(iv) the combined parameter (b, c),

unless coNP $\subseteq$ NP/poly.

Since parameters b and c are upper-bounded by k, a polynomial-size problem kernel with respect to the parameters b, c, or the combined parameter (b, c) would imply a polynomial-size problem kernel with respect to the parameter k. Thus, we only have to show the theorem for the parameter k. Instead of considering the general problem EE, we show that Theorem 2 holds even for the following, more restricted problem variant. Since this is a special case of EE, the hardness result also holds for EE.

2-Dimensional Eulerian Extension (2DEE)
Input: A directed graph $G = (V, A)$ with $V \subseteq \mathbb{N} \times \mathbb{N}$.
Question: Is there an Eulerian extension E for G with $\forall (u, v) \in E \colon u \preceq v$?

Herein, $(u_1, u_2) \preceq (v_1, v_2)$ means $u_1 \leq v_1$ and $u_2 \leq v_2$ and we call $(u, v) \in V \times V$ an *allowed* arc if $u \preceq v$. The 2DEE problem stems from an application in sequencing [10]. In order to prove that 2DEE does not admit a polynomial-size problem kernel, we use an intermediate problem called Switch Set Cover (SSC). To define SSC, we use the following terminology. Let U be a non-empty set. A *U-position* is a multiset with elements drawn from the *universe* U. A *U-switch* is a multiset whose elements are U-positions. When the set U is clear from the context, we simply speak of positions and switches.

Switch Set Cover (SSC)
Input: A set U and s switches each containing a number of positions.
Question: Is it possible to choose exactly one position in each switch such that each element of U is contained in at least one of the chosen positions?

We show that we can derive a polynomial-size problem kernel for SSC with respect to the combined parameter $(s, |U|)$ from a polynomial-size problem kernel for 2DEE with respect to the parameter k. This is done using a $\leq_{\mathrm{m}}^{\mathrm{PPP}}$-reduction.

Proposition 2. SSC *parameterized by* $(s, |U|)$ *is* $\leq_{\mathrm{m}}^{\mathrm{PPP}}$*-reducible to* 2DEE *parameterized by* k.

We also use the fact that both 2DEE and SSC are NP-complete. For 2DEE this is proven by Höhn et al. [10]. The same can be shown for SSC using a simple reduction from the well-known NP-hard Set Cover problem. Given a polynomial-size problem kernel for 2DEE, and an instance of SSC, we derive a problem kernel for SSC as follows. First, we construct an equivalent instance of 2DEE using the reduction from Proposition 2. Then, we reduce this instance of 2DEE to a polynomial-size problem kernel and transform the kernel-instance back to an equivalent instance of SSC using one of its NP-hardness reductions. This procedure yields a polynomial-size problem kernel for SSC because the first reduction increases the parameter at most polynomially, and the second reduction increases the instance size at most polynomially. However, we also prove that a polynomial-size problem kernel is unlikely to exist for SSC.

Proposition 3. Switch Set Cover *does not admit a polynomial-size problem kernel with respect to the combined parameter* $(s, |U|)$*, unless coNP* $\subseteq$ *NP/poly.*

Consequently, in order to prove Theorem 2, we have to prove Proposition 2 and Proposition 3. To prove Proposition 3, we use a framework introduced by Bodlaender et al. [1]: An *or-composition algorithm* for a parameterized problem $L \subseteq \Sigma^* \times \mathbb{N}$ is an algorithm that

(1) receives a number of instances $(I_1, k), \ldots, (I_m, k)$,
(2) runs in time that is polynomial in $\sum_{i=1}^{m} |I_i| + k$, and
(3) outputs an instance (I^*, k'), such that k' is bounded by a polynomial in k

and $(I^*, k') \in L$ if and only if $(I_j, k) \in L$ for some $1 \leq j \leq m$.

A parameterized problem is called *or-compositional* if there is an or-composition algorithm for it. Bodlaender et al. [1] showed that if an or-compositional parameterized problem admits a polynomial-size problem kernel, then coNP $\subseteq$ NP/poly.

To prove that SSC is or-compositional with respect to the parameter $(s, |U|)$, we employ the following strategy, introduced by Dom et al. [3]: First, we prove that Switch Set Cover is fixed-parameter tractable. More specifically, we show that it is solvable in $O^*(2^{s|U|})$ time.[4] Then, in the composition algorithm, if there are $m \geq 2^{s|U|}$ input instances, then we can directly solve all the instances in $O^*(m2^{s|U|}) \subseteq O^*(m^2)$ time and return a trivial yes- or no-instance depending on whether any of the m instances is a yes-instance. Thus, we may then assume that $m \leq 2^{s|U|}$ and, hence, $\log m \leq s|U|$. We exploit this to create an instance-chooser gadget by introducing $\log m$ new switches and $s \log m$ new elements into the output instance, increasing the parameter at most polynomially since $\log m \leq s|U|$. Every possible way to choose positions in these new switches will correspond to exactly one original instance that then is a yes-instance if the composite instance is a yes-instance. If, however, there is a yes-instance among the original instances, then the composite instance can be solved by simply configuring the chooser for this instance. We obtain the following result:

Observation 2. Switch Set Cover *is or-compositional with respect to the combined parameter* $(s, |U|)$.

In order to prove that EE has no polynomial-size problem kernel, according to our strategy, it remains to show the following:

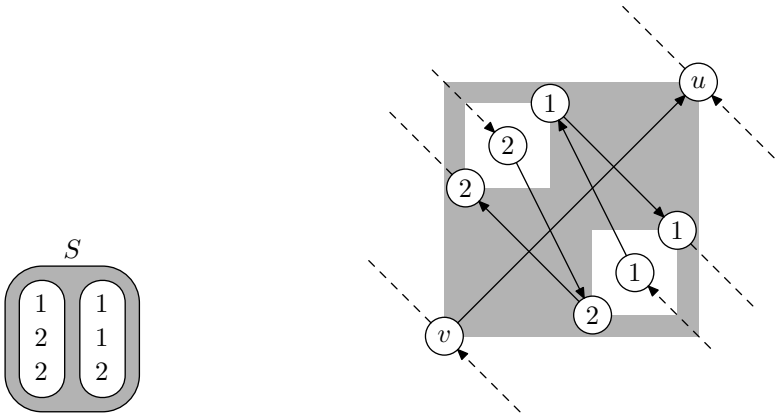
Observation 3. Switch Set Cover *can be solved in* $O^*(2^{s|U|})$ *time.*

Proof. An algorithm to solve SSC may simply try each combination of positions for all the switches: We may assume that in every switch there are at most $2^{|U|}$ positions because positions containing the same elements as other positions may be deleted and multiple copies of one element in one position may also be deleted. Thus, there are at most $(2^{|U|})^s$ combinations of positions. □

Next, we prove Proposition 2 by briefly sketching a $\leq_{\mathrm{m}}^{\mathrm{PPP}}$-reduction from SSC parameterized by the number of elements $|U|$ and the number of switches s to 2DEE parameterized by the number of extension arcs k. Since switches and positions are multisets, we can assume without loss of generality that all positions contain exactly $|U|$ elements. If this is not already the case, we can simply repeat elements or delete repeated elements.

The reduction uses the fact that, in 2DEE, all components of the input have to be connected. Analogously, all elements of an SSC instance have to be covered. We exploit this analogy by modeling elements as connected components. We represent each of the elements in U by a different component and introduce a special component, the "frame", to which the element-components can be connected. We use the geometrical restrictions of 2DEE to only allow connecting elements of exactly one position for each switch. To this end, consider a switch S.

[4] Here, O^* suppresses polynomial factors.



(a) A switch S in an instance of SSC.

(b) A gadget in the constructed instance of 2DEE corresponding to S.

Figure 1: The imbalanced vertices u and v are part of the frame. Since 2DEE only allows inserting arcs pointing to the lower left, only one of the groups of three vertices can be connected to the frame. Hence, an algorithm for 2DEE has to choose which group to connect, thus choosing a position for the switch S. Note that, since u and v are imbalanced, this gadget requires adding at least one arc.

We introduce a gadget for S that allows connecting all elements of exactly one position of S to the frame (see Figure 1). Each switch is represented by one of these gadgets and we use the restrictions of 2DEE to ensure that each of these gadgets is extended independently, thus representing the choice of a position for each of the switches in the original instance. We note that a similar notion of independence is also used in the NP-hardness proof for 2DEE by Höhn et al. [10].

Clearly, the described reduction runs in polynomial time. Furthermore, since each position contains at most $|U|$ elements, each gadget representing a switch allows for at most $|U|+1$ added arcs and it follows that at most $s|U|+s$ arcs have to be added. Hence, the presented construction constitutes a $\leq_{\mathrm{m}}^{\mathrm{PPP}}$-reduction from SSC to 2DEE. Together with Proposition 3 and the NP-hardness of SSC, Theorem 2 follows.

4 Conclusion

The most important remaining open question is to determine whether Eulerian Extension is fixed-parameter tractable solely with respect to the number of weakly connected components. Furthermore, it seems worthwhile to search for more efficient algorithms for the special case 2-Dimensional Eulerian Extension (see Section 3). It would also be interesting to see whether the newly introduced Switch Set Cover turns out be useful in other contexts. Having

(almost) excluded the possibility of polynomial-size many-one problem kernels, it seems tempting to analyze the potential existence of Turing problem kernels. Finally, the algorithms presented and those of previous work [4] appear to be not only of theoretical interest, making it promising to work on implementations and experiments with these algorithms.

References

[1] H. L. Bodlaender, R. G. Downey, M. R. Fellows, and D. Hermelin. On problems without polynomial kernels. *J. Comput. System Sci.*, 75(8):423–434, 2009.

[2] A. Cayley. A theorem on trees. *Quart. J. Math.*, 23:376—378, 1889.

[3] M. Dom, D. Lokshtanov, and S. Saurabh. Incompressibility through colors and IDs. In *Proc. 36th ICALP*, volume 5555 of *LNCS*, pages 378–389. Springer, 2009.

[4] F. Dorn, H. Moser, R. Niedermeier, and M. Weller. Efficient algorithms for Eulerian extension. Manuscript. Preliminary extended abstract appeared in *Proc. 36th WG*, 2010.

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

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

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

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

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

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

[11] S. Kapoor and H. Ramesh. Algorithms for enumerating all spanning trees of undirected and weighted graphs. *SIAM J. Comput.*, 24:247–265, April 1995.

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

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

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

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

[16] M. Sorge. *On Making Directed Graphs Eulerian*. Diplomarbeit, Institut für Informatik, Friedrich-Schiller Universität Jena, 2011. Available electronically. arXiv:1101.4283 [cs.DM].

[17] M. Sorge, R. van Bevern, R. Niedermeier, and M. Weller. A new view on rural postman based on Eulerian extension and matching. In *Proc. 22th IWOCA*, 2011. To appear.