Directing Road Networks by Listing Strong Orientations

Alessio Conte, Roberto Grossi, Andrea Marino, Romeo Rizzi, Luca Versari

Directing Road Networks by Listing Strong Orientations*

Alessio Conte[1], Roberto Grossi[1], Andrea Marino[1], Romeo Rizzi[2], Luca Versari[3]

[1] conte,grossi,marino@di.unipi.it, Università di Pisa, Italy
[2] romeo.rizzi@di.univr.it, Università di Verona, Italy
[3] luca.versari@sns.it, Scuola Normale Superiore, Italy

Abstract. A connected road network with N nodes and L edges has $K \leq L$ edges identified as one-way roads. In a feasible direction, these one-way roads are assigned a direction each, so that every node can reach any other [Robbins '39]. Using $O(L)$ preprocessing time and space usage, it is shown that all feasible directions can be found in $O(K)$ amortized time each. To do so, we give a new algorithm that lists all the strong orientations of an undirected connected graph with m edges in $O(m)$ amortized time each, using $O(m)$ space. The cost can be deamortized to obtain $O(m)$ delay with $O(m^2)$ preprocessing time and space.

1 Introduction

Consider a road network as a connected network with N nodes that correspond to road intersections, and L edges that correspond to road traits. Of the latter, $K \leq L$ are tagged as one-way roads whose direction must be decided, whereas the rest are two-way roads taken in both directions. The network has a feasible direction if there is an assignment of direction to each one-way road, so that from every node it is possible to reach all the other ones in the network. The problem of finding a feasible direction in a road network has been studied since Robbins' theorem, which gives the necessary and sufficient conditions [21]. In particular, the problem is named *one-way street problem* in [22].

Problem definition. This paper addresses the problem of discovering *all* the feasible directions in the one-way street problem, which might find application in situations where no clear apriori optimality criterion is available for directing the network, and multiple criteria must be tailored for the special situation at hand (e.g. some populous areas of big cities, which contain many narrow one-way roads). We reduce the problem of finding feasible directions in the road network to the problem of finding *strong orientations* of an undirected graph G with $n \leq 2K$ nodes and $m \leq 2K$ edges, where each strong orientation (SO afterwards) of G produces a distinct directed graph that is strongly connected, that is, every node can reach any other node.

* Work partially supported by the Italian Ministry of Education, University, and Research (MIUR) under PRIN 2012C4E3KT national research project AMANDA — Algorithmics for MAssive and Networked DAta.

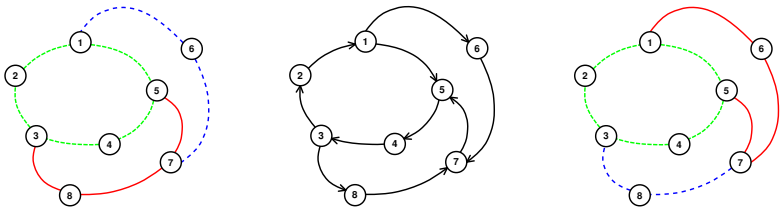

Fig. 1. Two ear decompositions (left and right) and a SO obtained from both (center)

Related work. Several papers by Roberts and Xu deal with these feasible directions [23–26] in the one-way street problem. The results reduce the latter to the problem of finding a SO of a *mixed multigraph*, which is a multigraph where both directed and undirected edges coexist. Robbins' theorem has been extended by Boesch and Tindel [3] accordingly, and Chung et al. [7] describe a linear time algorithm for finding a strong orientation in a mixed multigraph. In our reduction to listing SOs, however, we have the additional requirement of preserving all feasible directions in the reduction (see Section 2).

Some variations of the one-way street problem have been considered with the purpose of minimizing the average [11] or the maximum [12, 13, 15, 19] distance among all pairs of nodes, both of which are NP-hard problems [8, 20] (see [18] for a survey). Moreover, the minimum diameter among all the strong orientations of a given graph has been shown to be related with its domination number [13]. Other variations consider, for instance, the distance stretch for each pair of nodes [16], other connectivity constraints [1], cost-based constraints [5], degree-based constraints [2], and forced orientations [6].

The previous works mentioned above do not extend efficiently to our problem. By Robbins' theorem [21] the graphs that admit SOs are exactly the 2-edge connected graphs: in these graphs, for every pair of nodes there are two edge-disjoint paths connecting them; hence, if G is not 2-edge connected, the corresponding road network has no feasible direction. Its proof contains the following remarkable hint to find all the SOs, but it has some issues. Given an ear decomposition of G, it is possible to produce a SO by orienting each ear as a directed path, thus obtaining 2^k SOs from an ear decomposition with k ears. In general, listing ear decompositions and then obtaining SOs seems a natural approach to our problem. However, two different ear decompositions can lead to the same SO. Figure 1 shows two possible ear decompositions of a graph yielding the same SO: first orient the cycle $\{1, 2, 3, 4, 5\}$ clock-wise in both orientations, then in the left one orient the ears as $(1, 6, 7)$ and $(3, 8, 7, 5)$, whereas in the right one as $(3, 8, 7)$ and $(1, 6, 7, 5)$. It is easy to generalize this example, so that the same SO is obtained by many distinct ear decompositions.

A possible way to list *once* all the SOs would be to consider one edge at a time and employ the algorithm in [7] to check which orientations of that edge will lead to a solution. This approach would yield a recursive algorithm taking $O(m^2)$ time per solution because of the $O(m)$ recursion depth. It is natural to ask whether $O(m)$ time is possible, as each solution requires $O(m)$ to be output.

Our contribution. We present the first algorithm for efficiently listing once all the SOs in a graph G with m edges, with a cost of $O(m)$ time per solution and using $O(m)$ preprocessing time and total space. The cost can be deamortized to obtain $O(m)$ delay with $O(m^2)$ preprocessing time and space, where the delay is the maximum time elapsed between any two consecutive outputs. Using this result, we are able to find all the feasible directions of the road network in $O(K)$ amortized time per solution, using $O(L)$ preprocessing time and total space; also, the cost can be deamortized to obtain $O(K)$ delay using $O(K^2 + L)$ preprocessing time and total space. Furthermore, our approach easily extends to the enumeration of totally cyclic orientations, which are orientations in which every edge is part of a cycle. On a connected graph, these orientations are exactly the SOs [4], otherwise they are combinations of the SOs of each component. Note that SOs are not related to acyclic and cyclic orientations [9, 10], orientations with respectively no cycles or at least one, which require different algorithmic techniques.

In the paper we adopt the following notation for an undirected connected graph $G = (V, E)$ with $|V| = n$ nodes and $|E| = m$ edges. An orientation of G is the directed graph $\overrightarrow{G} = (V, A)$ where for any pair $\{u, v\} \in E$ either $(u, v) \in A$ or $(v, u) \in A$. The orientation $\overrightarrow{G}$ is strong if $\overrightarrow{G}$ is strongly connected. For the sake of clarity, we call *edges* the unordered pairs $\{x, y\}$ (undirected graph), while we call *arcs* the two possible orientations (x, y) and (y, x) (directed graphs).

2 From One-Way Streets to Strong Orientations

We show how to list solutions for the one-way street problem by a reduction to the problem of listing strong orientations, as this gives a cleaner proof of our results. As in [22], we use the notion of mixed graph $G = (V, E, A)$, i.e. a graph with vertices (in V) linked by the edges in E and by the arcs in A. Clearly, both directed and undirected graphs are special cases of mixed graphs, in which $E = \emptyset$ or $A = \emptyset$ respectively. Given the mixed graph $G = (V, E, A)$, we say that node x reaches node y if there is a path from x to y that uses directed edges in their correct orientation and/or undirected edges. G is strongly connected if u reaches v for every pairs of nodes $u, v \in V$, and is 2-edge connected if there are two edge-disjoint paths connecting u and v for every pair of distinct nodes $u, v \in V$. We refer to G as a mixed multigraph when E or A are multisets.

Consider a road network R with N intersections, K one-way roads and $L-K$ two-way roads. We thus model R as a mixed multigraph $M = (V_M, E_M, A_M)$ in which every node in V_M represents a road intersection, E_M is the multiset of edges corresponding to the one-way roads, and A_M is the multiset of directed arcs, that contains (x, y) and (y, x) for each two-way road linking the intersections modeled by x and y (hence, $|V_M| = N$, $|E_M| = K$, and $|A_M| = 2(L - K)$). A strong orientation of M is a direction assignment for the edges in E_M such that the resulting directed multigraph is strongly connected. Any edge $\{u, v\} \in E_M$ has two possible orientations (u, v) and (v, u), representing how the corresponding road is directed. We consider this to hold for self-loops as well.

It is straightforward to see how a strong orientation of M corresponds to a feasible way of directing R. We will map strong orientations of the mixed multigraph M to strong orientations of a suitable graph G.

To this aim, we introduce the following operation on mixed multigraphs:

Definition 1 (contraction of a directed cycle). *Given a mixed multigraph $M = (V_M, E_M, A_M)$ and a set of nodes $C \subseteq V_M$ which form a directed cycle, the contraction of C as a node c modifies M as follows: $V_M = (V_M \setminus C) \cup \{c\}$; for each edge $e \in E_M$ and each arc $a \in A_M$, any endpoint of e and a in C is replaced by c; finally, any* oriented *self-loop on c created this way is removed.*

Note that a contraction can create unoriented selfloops that we preserve along with their endpoints before the contraction.

Lemma 1 shows a useful property of the contraction of a directed 2-cycle, while Lemma 2 shows how to neglect undirected self-loops as well.

Lemma 1. *Let M be a mixed multigraph, and x, y a pair of nodes such that both arcs (x, y) and (y, x) exist in M. Let M' be the mixed multigraph obtained by contracting the directed 2-cycle $C = \{x, y\}$ as a node c. There is a one-to-one correspondence between the strong orientations of M and the ones of M'.*

Proof. Let us show that any SO of M induces a unique SO of M' and vice versa. We remark that all undirected edges of M are preserved in M', although some might have become undirected selfloops, thus we have a mapping from each undirected edge of M to a distinct one of M'. Note that this gives us a bijective mapping of the orientations of M and M', as each orientation is defined by the direction assignment of the undirected edges. Consider now a strong orientation of M: each node can reach/be reached by both x and y, thus in the correspondent orientation of M' each node will reach/be reached by c by construction, making M' strongly connected. Similarly a strong orientation of M' induces a strong orientation M. Thus we have a one-to-one correspondence between strong orientations of M and M'. □

Lemma 2. *Let M' be the multigraph obtained by removing all the k unoriented self-loops in the mixed multigraph M. Each strong orientation of M' corresponds to 2^k unique strong orientations of M, and all strong orientations of M can be found this way.*

Proof. Strong connectivity is not influenced by the removal of self-loops. Thus, removing all self-loop from a strong orientation of M gives us a strong orientation of M'. Moreover, given a strong orientation of M', we can obtain 2^k unique strong orientations of M by assigning arbitrary orientations to any self-loop (recall that each edge, including self-loops, has two possible orientations). Since two orientations obtained in this way from different orientations of M' are clearly distinct, the statement follows. □

Using Lemma 1 and Lemma 2 we transform $M = (V_M, E_M, A_M)$ in a graph $G = (V, E)$, exploiting the fact that all the arcs in A_M form a set of directed cycles of size 2 by construction. Our transformation proceeds as described next.

1. We contract every directed cycle in M to obtain an undirected multigraph M' according to Lemma 1. Note that M' contains only unoriented self-loops.
2. We remove all the self loops in M' according to Lemma 2.
3. From the resulting multigraph M'' we obtain $G = (V, E)$ as follows: for each edge $\{x, y\}$ in M'', we have edges $\{x, z\}$ and $\{z, y\}$ in E, where z is a new dummy node, and V is made of the nodes of M'' plus the new dummy nodes.

Note that $|V| = n \leq |V_M| + |E_M| = 2K$ and, similarly, $|E| = m \leq 2K$ by construction. We now show that this transformation is correct.

Lemma 3. *Let G be the graph obtained by applying the above transformation to a mixed multigraph M modelling a road network. Each strong orientation of G corresponds to 2^k unique strong orientations of M, where k is the number of self-loops removed in the transformation. Each strong orientation of M can be obtained this way.*

Proof. By Lemma 1 and 2, we only need to prove that there is a one-to-one correspondence between the strong orientations of G and the ones of M''. Let d_i denote the dummy node of G introduced by the transformation when "splitting" i-th edge $\{x, y\}$. Given an arbitrary orientation of M'', we define an orientation of G in the following way: if (x, y) is the orientation of $\{x, y\}$, then the orientations of $\{x, d_i\}$ and $\{y, d_i\}$ are (x, d_i) and (d_i, y). This mapping is clearly injective.

It is now sufficient to prove that any strong orientation of G is induced by a strong orientation of M'' and vice versa. Let u, w be two nodes of G. We can assume wlog that neither of them is a dummy node: if, say, $u = d_i$ for edge (x, y) of M'', then we have edges (x, u),(u, y) in G and we can replace u with y. Since G is strongly connected, and only has edges between dummy and non-dummy nodes, there exists a directed path $u = v_1, d_1, \ldots, d_{k-1}, v_k = w$ in G, which alternates non-dummy and dummy nodes. By construction of G and the mapping, it follows that $v_1, v_2, \ldots, v_{k-1}, v_k$ is a directed path from u to v in M''. For the converse, let u, w be two nodes of M''. Since M'' is strongly connected, there is a path $u = v_1, \ldots, v_k = w$ in M''. By construction, G has the path $u = v_1, d_1, \ldots, d_{k-1}, v_k$. □

3 Finding Strong Orientations

In this section we show how to efficiently find all the strong orientations (SOs) of an undirected graph $G = (V, E)$. We assume wlog that $G = (V, E)$ is 2-edge connected: this is a consequence of the following well-known result [21], as otherwise there are no SOs.

Theorem 1 (Robbins' theorem). *A graph G admits a strong orientation iff it is 2-edge connected.*

We introduce the key definitions and properties that will be used to build our algorithm. Using the standard definitions, we call a *cut* of G any bipartition V_1, V_2 of its nodes and we say that an edge $\{x, y\}$ or an arc (x, y) crosses the cut

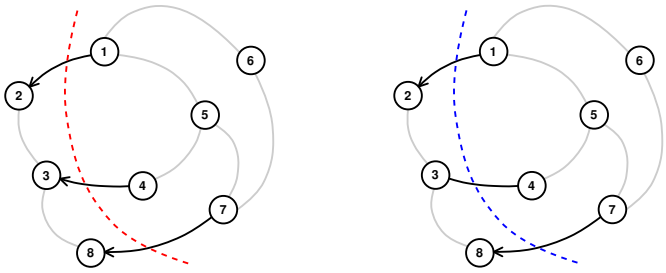

Fig. 2. Two partial orientations of a mixed graph: a ONE-WAY CUT (left) and a FORCING CUT with bound edge $\{3, 4\}$ having $(3, 4)$ as bound direction (right)

if $x \in V_1$ and $y \in V_2$, or vice versa. We define two kinds of cuts which will help us model the problem, namely ONE-WAY CUT and FORCING CUT.

Definition 2 (ONE-WAY CUT). *Given a mixed graph $G = (V, E, A)$, we call a cut V_1, V_2 of V a* ONE-WAY CUT *if the cut is crossed only by arcs, which are all oriented towards V_1 (alternatively, the arcs are all oriented towards V_2).*

We will also exploit another kind of cut that lets us foresee which orientations of which edges will produce a ONE-WAY CUT:

Definition 3 (FORCING CUT). *Given a mixed graph $G = (V, E, A)$, we call a cut V_1, V_2 of V a* FORCING CUT *if the cut is crossed by exactly one undirected edge, called* bound edge*, and by one or more arcs that are all oriented towards V_1. We call* bound direction *the one obtained by orienting the bound edge towards V_2. (The roles of V_1 and V_2 can be interchanged.)*

Note that we cannot have zero arcs in a FORCING CUT of G as otherwise G would not be 2-edge connected.

Lemma 4. *Let $G = (V, E, A)$ be a 2-edge connected mixed graph that has no* ONE-WAY CUT*. Then any node x reaches any other node y.*

Proof. Let us suppose by contradiction that there exist two nodes x, y such that x does not reach y. Let V_x be the set of nodes that are reachable from x. Since $y \notin V_x$, we have that $V_x, V \setminus V_x$ is a cut of the graph. Moreover, by its definition there can be no edge going from a node of V_x to a node of $V \setminus V_x$, so $V_x, V \setminus V_x$ is a ONE-WAY CUT as the graph is connected. □

The above lemma together with Theorem 3, are crucial to understand the idea behind our approach. For this, we need the following known theorem in [3], that extends Robbins' theorem.

Theorem 2 (Boesch and Tindell). *A mixed graph G has a* SO *if and only if G is strongly connected and 2-edge connected.*

We say that a mixed graph can be completed or extended to a SO if there exists a direction assignment for its edges such that the resulting digraph is a SO. By ensuring that our partial orientation never admits a ONE-WAY CUT, we can ensure the existence of a strongly connected extension using Boesch and Tindell's theorem.

Theorem 3. *A 2-edge connected mixed graph $G = (V, E, A)$ can be completed to form a* SO *iff G does not admit a* ONE-WAY CUT*.*

Proof. If G has a ONE-WAY CUT V_1, V_2, clearly it cannot be extended to a SO. Indeed, as all edges between V_1, V_2 are already oriented, the cut will still be a ONE-WAY CUT in any extension, thus nodes in V_2 will not be reachable by nodes in V_1.

To prove the other implication, note that by Lemma 4 we have that in G any node can reach any other node. Moreover we know by hypothesis that G is 2-edge connected. Boesch and Tindell's theorem implies that such a graph has a SO, proving our result. □

Finally, we show how the concept of FORCING CUT is important for the completion of an orientation as a SO. In particular, Theorem 4 extends Lemma 2 in [7].

Lemma 5. *Let $G = (V, E, A)$ be a 2-edge connected mixed graph, and V_1, V_2 a cut of V. Then V_1, V_2 can be turned into a* ONE-WAY CUT *by orienting exactly one undirected edge iff V_1, V_2 is a* FORCING CUT*.*

Proof. The proof follows immediately from the definitions of ONE-WAY CUT and FORCING CUT. □

Theorem 4. *Let $G = (V, E, A)$ be a 2-edge connected mixed graph that has no* ONE-WAY CUT*, and $\{x, y\}$ an undirected edge in E. Then neither of the orientations (x, y) and (y, x) of the edge will create a* ONE-WAY CUT *iff $\{x, y\}$ is not a bound edge.*

Proof. If $\{x, y\}$ is not a bound edge, both orientations lead to a solution. Indeed, any cut crossed by $\{x, y\}$ is not a FORCING CUT, thus by Lemma 5 any orientation of $\{x, y\}$ will not produce a ONE-WAY CUT. If $\{x, y\}$ is a bound edge, then there is a cut V_1, V_2 of V in which all edges are oriented towards V_1 except for $\{x, y\}$. Orienting $\{x, y\}$ towards V_1 will create a ONE-WAY CUT. □

3.1 Algorithm description

The above properties are the guidelines for a simple and efficient algorithm to enumerate the SOs of G. The core idea hinges on bound edges to guarantee that each recursive call either outputs a new SO or yields two calls that will produce at least one new SO each.

The ideas are detailed in Algorithm 1: it is a recursive approach that consists in incrementally exploring all the possible ways of orienting edges of G that

Algorithm 1: Finding all strong orientations (SOs)

Input : Graph $G = (V, E)$.
Output: All SOs of G.
STRONG-ORIENTATIONS$(V, E, \emptyset)$

Function STRONG-ORIENTATIONS(V, E, A)
  $B \leftarrow$ bound edges in mixed graph $G = (V, E, A)$
  $E \leftarrow E \setminus B$
  $A \leftarrow A \cup \{(b, c) : (b, c)$ is the bound direction of $\{b, c\} \in B\}$
  if $E = \emptyset$ **then**
    output SO $\leftarrow \overrightarrow{G} = (V, A)$
  else
    $\{x, y\} \leftarrow$ an arbitrary edge in E
    $E \leftarrow E \setminus \{\{x, y\}\}$
    STRONG-ORIENTATIONS$(V, E, A \cup \{(x, y)\})$
    STRONG-ORIENTATIONS$(V, E, A \cup \{(y, x)\})$

will lead to a solution. In the beginning G is completely undirected, so it will not contain a ONE-WAY CUT. By Theorem 4 we know that the edges that can create a ONE-WAY CUT are exactly all the bound edges; let B be the set of such edges. Each edge in B must be oriented according to its bound direction, as it would otherwise create a ONE-WAY CUT. Note that as a consequence of Boesch and Tindell's theorem [3], if there is at least one SO, then the bound direction does not create a ONE-WAY CUT. For all other edges, we are free to chose any orientation. Thus we orient the edges in B according to their suitable direction, pick an arbitrary edge $\{x, y\}$ (if any), and recur on both possible ways (x, y) and (y, x) of orienting $\{x, y\}$. When there are no more edges that can be oriented we output the current orientation.

It remains to describe how to find the bound edges in B. In any recursive step, our algorithm starts with a mixed graph $G = (V, E, A)$, where A are the edges that have been already directed, and E the ones that have not. We need to find in this graph all the bound edges in E, that is, all the FORCING CUTs of M. As we will show in Lemma 7, these are actually all the undirected edges which are *strong bridges*.

Definition 4 (strong bridge). *Given a mixed graph G, a strong bridge is an edge that, if removed, increases the number of strongly connected components of G.*

Using the algorithm by Italiano et al. [17] we can find all strong bridges in G in $O(|E| + |A|)$ time. The algorithm is intended for directed graphs, but it can also be applied to mixed graphs by considering each undirected edge $\{x, y\}$ as a pair of arcs (x, y), (y, x) with opposite directions, so as to traverse $\{x, y\}$ in both directions: whichever is chosen between (x, y) and (y, x) as a strong bridge,

gives the bound direction to $\{x, y\}$. (Note that (x, y) and (y, x) cannot be both chosen as strong bridges.)

3.2 Correctness

As any edge that is not bound can be oriented in both ways and lead to a solution by Theorem 4, we observe the following fact.

Lemma 6. *Let e be an edge that is not bound in G. Then, orienting any bound edge of G in its forced direction does not make e a bound edge.*

Proof. It follows from the observation that any cut involving e has at least two undirected edges, thus orienting the bound edges cannot affect e.

Lemma 7. *Let $\{x, y\}$ be an undirected edge in a strongly connected mixed graph G. Then $\{x, y\}$ is bound iff it is a strong bridge.*

Proof. We will first prove that if $\{x, y\}$ is a bound edge then it is a strong bridge. Indeed, if V_x, V_y is the FORCING CUT of $\{x, y\}$, where all other edges go from V_x to V_y, then removing $\{x, y\}$ makes nodes in V_y unable to reach nodes in V_x, increasing the number of strongly connected components of G, thus $\{x, y\}$ is a strong bridge.

Suppose now that $\{x, y\}$ is a strong bridge. Let V_x and V_y be the set of nodes reachable from respectively x and y without using the edge $\{x, y\}$. Since $\{x, y\}$ is a strong bridge, either $V_x \neq V$ or $V_y \neq V$. Let V_1 be the set, chosen between V_x and V_y, satisfying the latter disequality. Let $V_2 = V \setminus V_1$ be the complement set, which is nonempty, and consider the cut V_1, V_2: all the arcs in this cut (except $\{x, y\}$) must be oriented towards V_1, as otherwise V_1 would be larger. Hence, V_1, V_2 is a FORCING CUT for $\{x, y\}$ because V_1 has no outgoing edges to V_2 other than $\{x, y\}$ itself. □

Theorem 5. *Given a 2-edge connected graph $G = (V, E)$, our algorithm correctly outputs all the strong orientations of G exactly once.*

Proof. A 2-edge connected mixed graph can be completed to form a SO iff it does not admit a ONE-WAY CUT by Theorem 3. Hence, we prove by induction on $|E|$ that, if $G' = (V, E, A)$ is a mixed graph with no ONE-WAY CUT, our algorithm outputs all the SOs of G' once.

Base case for $|E| = 0$. Then G' is completely oriented and with no ONE-WAY CUT, so by Lemma 4 it is strongly connected. Moreover it has exactly one SO $\overrightarrow{G'} = (V, A)$, which we output.

Inductive step for $|E| > 0$. We can identify all the bound edges in G' and their bound directions by Lemma 7, using the algorithm in [17]. Orienting bound edges in their bound direction does not alter the set of SOs of G', since there is no SO that has a bound edge in the other direction: as each bound edge belongs to a FORCING CUT, orienting that edge otherwise would create a ONE-WAY CUT by Lemma 5. Also, orienting a bound edge in its bound direction cannot create a

new bound edge by Lemma 6. We can thus consider G' as having no bound edges, without loss of generality. If G' has no more undirected edges, we fall back to the base case. Otherwise, given an undirected edge e of G', we know that orienting it either way does not produce any ONE-WAY CUT by Theorem 4. Any SO must have e in either one direction or the other. Let G'_1 and G'_2 be the graphs obtained by orienting e in each way, respectively. Since both G'_1 and G'_2 have a smaller number of undirected edges than G', we know by inductive hypothesis that our algorithm terminates, outputting all the SOs of G'_1 and G'_2 once. Any SO of G' is a SO of either G'_1 and G'_2, and the latter have no intersection as they differ on the orientation of e. Hence, the algorithm produces all the SOs of G' once. □

3.3 Analysis

We now analyze the time and space cost of our algorithm on the graph $G = (V, E)$, with $|V| = n$ and $|E| = m$ assuming wlog that it is connected. We remark that each recursion node which is not a leaf has at least two children, and that every leaf of the computation tree outputs a distinct solution. This gives us a computation tree with no unary nodes[4] and α leaves, where α is the number of solutions. It follows that the total number of recursion nodes is bounded by $2 \cdot \alpha$ and thus the amortized cost per solution of the algorithm is bounded by the cost of a single recursion node.

Consider the structure of Algorithm 1. We show how every step takes $O(m)$ time. Computing bound edges is done in $O(m)$ time by finding the strong bridges and selecting the undirected ones; moreover, the algorithm by Italiano et al. [17] is applied to a directed graph where each undirected edge is represented by two directed arcs, thus finding a strong bridge will immediately give us the bound direction of the corresponding bound edge, making the assignment of bound directions clearly $O(m)$ time. All other steps involve updating or scanning sets of size $O(m)$, which trivially take $O(m)$ time each. The total cost is $O(m \cdot \alpha)$, or equivalently $O(m)$ amortized cost per solution. We remark that this cost is optimal for merely printing each SO.

Finally we show that the space cost is bounded by $O(m)$ as well: indeed, the working space of a single recursion node is $O(m)$, but the information that needs to be passed on to child recursive calls, other than the input, is simply the partial orientation of the graph. If stored as the difference with the partial orientation in the parent node, the space requirement of a root-to-leaf path (and thus of the whole algorithm) is always $O(m)$. Thus the following holds:

Theorem 6. *Given a 2-edge connected graph $G = (V, E)$, Algorithm 1 outputs all the strong orientations of G exactly once, in $O(m)$ amortized time, using $O(m)$ total space.*

We observe that the delay of Algorithm 1 is bounded by the sum of the costs along a leaf-to-root path and a root-to-leaf path. Since the cost of each recursion

[4] This is crucial, as the presence of unary nodes is the reason behind the $O(m^2)$ cost of the approach based on [7], mentioned in the introduction.

node is $O(m)$, and the depth of the computation tree is at most m, we obtain $O(m^2)$ delay. We will now show how the delay can be reduced to $O(m)$ using the Output Queue Method by Uno [27], which suitably accumulates solutions that arrives at an irregular pace to output them in a regular fashion, using a queue of bounded size. The method depends on two parameters: T^*, the maximum cumulative cost in a root-to-leaf path of the recursion tree, and $\bar{T}$, an upper bound on the amortized cost per solution in any subtree of the computation. In our case, the former is $O(m^2)$ as discussed above, and the second is $\Theta(m)$, as each k-size subtree of our binary recursion tree has $\Theta(k)$ leaves (i.e. solutions since there are no unary nodes), and a node takes $O(m)$ time. As a result, using a queue of $O(T^*/\bar{T}) = O(m)$ solutions, we can output each solution with delay $O(\bar{T}) = O(m)$. This takes $O(T^* + \bar{T}) = O(m^2)$ preprocessing time and $O(m \cdot T^*/\bar{T}) = O(m^2)$ space.

Theorem 7. *Given a 2-edge connected graph $G = (V, E)$, there exists an algorithm that outputs all the strong orientations of G exactly once, with $O(m)$ delay, using $O(m^2)$ preprocessing time and total space.*

4 Conclusions

In this paper we considered the problem of finding all feasible ways of directing a connected road network, also known as the one-way street problem, and reduced the latter to the problem of listing all the strong orientations in an undirected connected graph. The bounds are optimal if one wants to print each strong orientation. A referee suggests the interesting open problem of enumerating totally cyclic orientations in 3-edge connected graphs [14] in constant amortized time by listing only the edges that get flipped from orientation to orientation.

References

1. Esther M. Arkin and Refael Hassin. A note on orientations of mixed graphs. *Discrete Applied Mathematics*, 116(3):271 – 278, 2002.
2. Walid Ben-Ameur, Antoine Glorieux, and José Neto. On the most imbalanced orientation of a graph. In *Computing and Combinatorics: 21st International Conference, COCOON 2015*, LNCS, pages 16–29. Springer, 2015.
3. Frank Boesch and Ralph Tindell. Robbins' theorem for mixed multigraphs. *The American Mathematical Monthly*, 87(9):716–719, 1980.
4. Béla Bollobás. *Modern graph theory*. Springer-Verlag, New York, 1998.
5. Rainer E Burkard, Karin Feldbacher, Bettina Klinz, and Gerhard J Woeginger. Minimum-cost strong network orientation problems: Classification, complexity, and algorithms. *Networks*, 33(1):57–70, 1999.
6. Gary Chartrand, Frank Harary, Michelle Schultz, and Curtiss E Wall. Forced orientation number of a graph. *Congressus Numerantium*, pages 183–192, 1994.
7. Fan RK Chung, Michael R Garey, and Robert E Tarjan. Strongly connected orientations of mixed multigraphs. *Networks*, 15(4):477–484, 1985.
8. Vašek Chvátal and Carsten Thomassen. Distances in orientations of graphs. *Journal of Combinatorial Theory, Series B*, 24(1):61 – 75, 1978.

9. Alessio Conte, Roberto Grossi, Andrea Marino, and Romeo Rizzi. Enumerating cyclic orientations of a graph, 2015. IWOCA 2015.
10. Alessio Conte, Roberto Grossi, Andrea Marino, and Romeo Rizzi. Listing acyclic orientations of graphs with single and multiple sources. In *LATIN 2016: Theoretical Informatics - 12th Latin American Symposium, Ensenada, Mexico, April 11-15, 2016, Proceedings*, pages 319–333, 2016.
11. Peter Dankelmann, Ortrud R. Oellermann, and Jian-Liang Wu. Minimum average distance of strong orientations of graphs. *Discrete Applied Mathematics*, 143(13):204 – 212, 2004.
12. Fedor V Fomin, Martín Matamala, and Ivan Rapaport. Complexity of approximating the oriented diameter of chordal graphs. *Journal of Graph Theory*, 45(4):255–269, 2004.
13. Fedor V Fomin, Martın Matamala, Erich Prisner, and Ivan Rapaport. At-free graphs: linear bounds for the oriented diameter. *Discrete Applied Mathematics*, 141(1):135–148, 2004.
14. Komei Fukuda, Alain Prodon, and Tadashi Sakuma. Combinatorics and computer science notes on acyclic orientations and the shelling lemma. *Theoretical Computer Science*, 263(1):9 – 16, 2001.
15. G. Gutin. Minimizing and maximizing the diameter in orientations of graphs. *Graphs and Combinatorics*, 10(2):225–230, 1994.
16. Rafael Hassin and Nimrod Megiddo. On orientations and shortest paths. *Linear Algebra and its Applications*, 114115:589 – 602, 1989. Special Issue Dedicated to Alan J. Hoffman.
17. Giuseppe F Italiano, Luigi Laura, and Federico Santaroni. Finding strong bridges and strong articulation points in linear time. *Theoretical Computer Science*, 447:74–84, 2012.
18. K.M. Koh and E.G. Tay. Optimal orientations of graphs and digraphs: A survey. *Graphs and Combinatorics*, 18(4):745–756, 2002.
19. Martin Laetsch and Sascha Kurz. Bounds for the minimum oriented diameter. *Discrete Mathematics & Theoretical Computer Science*, 14, 2012.
20. Ján Plesník. On the sum of all distances in a graph or digraph. *Journal of Graph Theory*, 8(1):1–21, 1984.
21. H. E. Robbins. A theorem on graphs, with an application to a problem of traffic control. *The American Mathematical Monthly*, 46(5):281–283, 1939.
22. F. S. Roberts. *Graph Theory and its Applications to Problems of Society*. NSF-CBSM Monograph No. 29. SIAM Publications, 1978.
23. Fred S Roberts and Yonghau Xu. On the optimal strongly connected orientations of city street graphs. II: Two east-west avenues or northsouth streets. *Networks*, 19(2):221–233, 1989.
24. Fred S Roberts and Yonghua Xu. On the optimal strongly connected orientations of city street graphs I: Large grids. *SIAM J. on Discrete Math.*, 1(2):199–222, 1988.
25. Fred S Roberts and Yonghua Xu. On the optimal strongly connected orientations of city street graphs. III. Three east–west avenues or north–south streets. *Networks*, 22(2):109–143, 1992.
26. Fred S Roberts and Yonghua Xu. On the optimal strongly connected orientations of city street graphs IV: Four east-west avenues or north-south streets. *Discrete Applied Mathematics*, 49(1):331–356, 1994.
27. Takeaki Uno. Two general methods to reduce delay and change of enumeration algorithms, 2003. NII Technical Report NII-2003-004E, Tokyo, Japan.