SIAM J. COMPUT.
Vol. 42, No. 2, pp. 634–661

LINK REVERSAL ROUTING WITH BINARY LINK LABELS: WORK COMPLEXITY*

BERNADETTE CHARRON-BOST†, ANTOINE GAILLARD‡, JENNIFER L. WELCH§, AND JOSEF WIDDER¶

Abstract. Full Reversal and Partial Reversal are two well-known routing algorithms that were introduced by Gafni and Bertsekas [*IEEE Trans. Commun.*, 29 (1981), pp. 11–18]. By reversing the directions of some links of the graph, these algorithms transform a connected input DAG (directed acyclic graph) into an output DAG in which each node has at least one path to a distinguished destination node. We present a generalization of these algorithms, called the link reversal (LR) algorithm, based on a novel formalization that assigns binary labels to the links of the input DAG. We characterize the legal link labelings for which LR is guaranteed to establish routes. Moreover, we give an *exact* expression for the number of steps — called *work complexity* — taken by each node in every execution of LR from any legal input graph. Exact expressions for the per-node work complexity of Full Reversal and Partial Reversal follow from our general formula; this is the first exact expression known for Partial Reversal. Our binary link labels formalism facilitates comparison of the work complexity of certain link labelings — including those corresponding to Full Reversal and Partial Reversal — using game theory. We consider labelings in which all incoming links of a given node i are labeled with the same binary value μ_i. Finding initial labelings that induce good work complexity can be considered as a *game* in which to each node i a player is associated who has strategy μ_i. In this game, one tries to minimize the cost, i.e., the number of steps. Modeling the initial labelings as this game allows us to compare the work complexity of Full Reversal and Partial Reversal in a way that provides a rigorous basis for the intuition that Partial Reversal is better than Full Reversal with respect to work complexity.

Key words. link reversal routing, wireless networks, complexity of algorithms, applications of game theory

AMS subject classifications. 68W15, 68W40, 91A80, 68Q25

DOI. 10.1137/110843095

1. Introduction.

Mobile radio networks, with their ad hoc deployments, node mobility, and wireless communication, pose serious challenges for developing provably correct and efficient applications. A popular algorithm design technique for such systems is link reversal, first proposed by Gafni and Bertsekas [10] for routing. Subsequent routing algorithms based on link reversal were developed (see, e.g., [21, 12]), and extensions were made to solve other problems in mobile ad-hoc networks including mutual exclusion [23, 17, 25] and leader election [15, 9, 11]. Link reversal has also been employed in algorithms for resource allocation [5, 2, 14] and distributed

*Received by the editors August 2, 2011; accepted for publication (in revised form) January 29, 2013; published electronically April 3, 2013. This work was presented in part at ACM SPAA 2009 [7] and Algosensors 2009 [8].

http://www.siam.org/journals/sicomp/42-2/84309.html

†CNRS, LIX, Ecole polytechnique, F-91128 Palaiseau, France (charron@lix.polytechnique.fr).

‡LIX, Ecole polytechnique, F-91128 Palaiseau, France (antoine.gaillard@gmail.com).

§Department of Computer Science and Engineering, Texas A&M University, College Station, TX 77843 (welch@cse.tamu.edu). The research of this author was supported in part by NSF grants 0500265 and 0964696, and by Texas Higher Education Coordinating Board grants ARP 000512-0130-2007 and ARP 000512-0007-2006.

¶TU Wien, Formal Methods in Systems Engineering Group, 1040 Wien, Austria (widder@forsyte.at). The research of this author was supported in part by the French DGA/LIX project, by the Austrian National Research Network S11403-N23 (RiSE) of the Austrian Science Fund (FWF), by FWF projects P20529 and P18264, and by the Vienna Science and Technology Fund (WWTF) grant PROSEED.

queuing [24, 1]. Gafni and Bertsekas [10] considered the following routing problem: the network contains a unique destination node and a virtual direction is associated with each communication link. The link directions should eventually ensure that every node has a directed path to the destination, i.e., that the graph corresponding to the nodes and the directed links is *destination-oriented*. Messages can then be routed through the network from a node to the destination by following the virtual directions of the links.

A common approach to the routing problem is to construct a spanning tree rooted at the destination and to forward messages over the edges of the spanning tree toward the root. An advantage of the link reversal approach over the spanning tree approach is the built-in fault tolerance. In the spanning tree approach, any single failure of a tree link disconnects the tree and prevents nodes downstream from the failed link from communicating with the destination. In contrast, the link reversal approach exploits the redundancy in the entire graph by assigning directions to all links. As long as a path to the destination remains, any number of link failures can be tolerated.

If, due to the failure or movement of one node and the resulting loss of its incident links, a node no longer has a directed path to the destination, then the network should adjust itself to restore the property. The adjustments consist of nodes *reversing* some of their incident links. Nodes should be able to determine autonomously, based only on local information, whether they need to perform reversals.

Gafni and Bertsekas [10] restricted their study to the setting of *acyclic* directed graphs and presented two abstract algorithms, each with a distributed implementation. In both abstract graph algorithms, any node other than the destination that becomes a sink reverses some of its incident links. The algorithms differ by which incident links they choose for reversal. In the *Full Reversal* algorithm, all incident links are reversed, whereas in the *Partial Reversal* algorithm, only those that have not been reversed since the last time this node was a sink are reversed, roughly speaking. The dynamics of link directions in both algorithms is implemented by assigning a *height*, drawn from a totally ordered set, to each node, and considering a link to be directed from the endpoint with larger height to that with smaller height. A link reversal is implemented by increasing the height of a sink.

In more detail, each node has a unique identifier from a totally ordered set as, for instance, the natural numbers. The height for the Full Reversal implementation is an ordered pair consisting of the value of an (unbounded) counter and the unique identifier of the node. The height for the Partial Reversal implementation is an ordered triple consisting of the values of two unbounded counters and the unique identifier of the node. For both implementations, the order under consideration is the lexicographic order. We will refer to these two implementations as the *pair* and *triple* algorithms. A sink applies a function, based on its current height and those of its neighbors, to calculate its new height; for the Full Reversal implementation, the function simply sets the counter to be larger than the counters of all neighbors, whereas the function for the Partial Reversal algorithm increases in a smoother mode.

Gafni and Bertsekas proved that the two resulting distributed algorithms are correct, that is, that they both terminate and, when they do, that the directions on the links form a destination-oriented acyclic graph. For the proof, they consider a generalization of the heights and the functions that manipulate them, and prove correctness for the generalization.

The order-based approach by Gafni and Bertsekas has several advantages, which may account for its popularity. First, it allows the unification of a large class of distributed algorithms, including the pair and triple algorithms, in an elegant way.

As a result, the convergence of the two algorithms is proven in [10] by considering only some abstract properties of the heights and the functions that change them. Moreover, by varying the representation of heights as well as the functions, one may expect to get more algorithms than just the pair and triple algorithms. Finally, the ordered structure of heights gives the acyclicity of the solution for free.

However, when examining the approach closely, two questions arise. First, there is no proof that the pair and triple algorithms actually coincide with Full Reversal and Partial Reversal; while in the case of the pair algorithm the correspondence to Full Reversal is obvious, a formal reasoning for Partial Reversal would be much more involved [6]. In fact, there was no proof that Partial Reversal converges toward a destination-oriented graph until the preliminary version of this paper appeared [7]. Second, even when we assume the exact correspondence between the pair and triple algorithms and the Full Reversal and Partial Reversal algorithms, the height implementation does not seem to be well-adapted to the analysis of the abstract link reversal algorithms, as the height implementation makes the analysis more obscure. Indeed, other than the exact formula for the pair algorithm and the worst-case asymptotic analyses for the pair and triple algorithms due to Busch, Surapaneni, and Tirthapura [3, 4], little is known about the work complexity of these algorithms. In particular, there is no fine-grained comparison of Full Reversal and Partial Reversal that would guide system designers in the choice of implementing one link reversal algorithm over the other.

Binary link labelings and exact work complexity. From the informal specifications of the abstract Full Reversal and Partial Reversal algorithms, we give a formal definition of a generalization of these algorithms. Our generalized algorithm takes as input a directed acyclic graph (DAG) with binary labels on the links and, under certain conditions on the labeling, establishes routes to the destination. More precisely, we give a condition (AC) which ensures that the graph remains acyclic during the whole execution. The Full Reversal (resp., Partial Reversal) algorithm is the specialization of our general algorithm in which all initial link labels are $\mathbb{1}$ (resp., $\mathbb{0}$). We check that both these initial link labelings satisfy (AC) in any acyclic graph, which proves that Full Reversal and Partial Reversal preserve acyclicity. Hence, our formalization allows us to give a simple and unified proof that both algorithms result in a destination-oriented DAG.

Surprisingly, there was no methodical study of the complexity of link reversal algorithms until the work by Busch, Surapaneni, and Tirthapura [3, 4], which analyzed the *work complexity of a node* — the number of reversals performed by the node — for the pair and the triple algorithms.[1] For the pair algorithm, the authors gave an exact expression of the work complexity of each node, while for the triple algorithm they only established an upper bound. Concerning the *global work complexity*, that is, the total number of reversals performed by all the nodes, they thus obtained upper bounds for the pair and triple algorithms, both of which are quadratic in the number of nodes and are asymptotically tight.

Our formalization of the general link reversal algorithm allows us to express the work of each node in any execution of the algorithm on any acyclic input graph. The expression depends only on the initial link-labeled graph and is specialized to simple formulas for Full Reversal and Partial Reversal. In contrast, the work complexity

[1] More precisely, Busch, Surapaneni, and Tirthapura considered the triple algorithm with initial heights which are more general than in [10]. This extension of the triple algorithm is discussed in section 7.

result in [3, 4] is exact only for the pair algorithm (corresponding to Full Reversal). Having an exact and general formula facilitates determining the best and worst graph topologies and, as we discuss next, allows us to conduct a fine-grained comparison of Full Reversal and Partial Reversal based on game theory.

Locally uniform labelings as a game. Interestingly, our algorithm captures more than just the Full Reversal and Partial Reversal algorithms, by considering nonuniform initializations of link labels. We investigate the *locally uniform* (LU) policy for initial link labels: for each node i, all the *incoming* links of i are initially labeled with the same value $\mu_i \in \{\mathbb{0}, \mathbb{1}\}$. In this policy, Full Reversal corresponds to the uniform labeling $\mathbb{1}$ (for each i, $\mu_i = \mathbb{1}$) and Partial Reversal to the uniform labeling $\mathbb{0}$ (for each i, $\mu_i = \mathbb{0}$). Moreover, we will see that for any acyclic graph, LU ensures (AC).

We then use game theory to compare the work done in executions starting with graphs labeled according to LU. As LU is a local policy (each node may be responsible for the labeling of its incoming links), it is natural to associate to each node a player with the two pure strategies $\mathbb{0}$ and $\mathbb{1}$ corresponding to the label choice. Then a *strategy profile* — consisting of a strategy for each player — corresponds to an initial labeled graph. The *cost* incurred by player i is then the work done by i in an execution starting from the initial labeled graph. The *social cost* of a given strategy profile is defined as the global work (the sum of the work over all nodes).

First, we study the influence of i's choice on its own work as well as on the work of the others. From this we observe that the strategy of some node i that reduces the cost incurred by i may increase the cost incurred by other nodes, and also the social cost. These conflicting interests motivate our detailed game-theoretic analysis. We start this analysis by giving a characterization of pure Nash equilibria for this game. As a result we obtain that there always exists a pure Nash equilibrium, namely the profile corresponding to Full Reversal. We show that in terms of social cost (global work complexity) however, this profile is the worst pure Nash equilibrium. Moreover, the social cost of Full Reversal may be larger than the optimal social cost by some factor which depends on the graph. We will see that for certain chains of length n, this factor is $\frac{n+1}{2}$, and so there is no constant bound on the price of anarchy [13] which holds for all graphs.

In contrast, the profile corresponding to Partial Reversal is not necessarily a pure Nash equilibrium. However, we show that its social cost is at most twice the optimal social cost. Hence, even if not optimum, Partial Reversal is never a disastrous strategy compared to others and seems "less risky" than Full Reversal in this respect.

Finally, we consider the *mixed* [18] version of the game, in which each player i's mixed strategy μ_i^* is the probability that μ_i is set to $\mathbb{1}$, and the cost is the expected work. In such a way, we can model random initialization of the link-labeled graph according to LU as a mixed game. We provide a characterization of mixed Nash equilibria and show that, in some sense that we make precise, there are no interesting mixed Nash equilibria other than the pure ones, and thus randomization does not help to find more efficient initial link labelings under the LU policy.

Organization of the paper. In section 2, we formally define the problem and review the previous solutions given by Gafni and Bertsekas [10]. In section 3, we present our general link reversal algorithm and prove its correctness. We also introduce the LU policy for the initial link labeling. The work complexity of the general solution is then analyzed in section 4. In section 5, we first show how our results can be specialized to obtain results for Full Reversal and Partial Reversal, and then we study in more detail the complexity of the LU policy. These findings support a game-theoretic comparison of different initial link labelings, presented in section 6. In section 7, we compare our results to those in [3, 4].

2. The destination orientation problem.

2.1. Notation and terminology. We consider directed graphs, or graphs for short, of the form $G = \langle V \cup \{D\}, E\rangle$: graph G has $n+1$ nodes, one of which is a specific node called the *destination*. For convenience, we refer to the destination as node D, and we let $V = \{1, 2, \ldots, n\}$ be the remaining nodes. The link (i, j) in E is said to be *incident* on both i and j and to be *outgoing* from i and *incoming* to j. A node i is said to be a *sink* if all its incident links are incoming to i.

A *chain* is a sequence C of nodes $i_0, \ldots, i_k$, $k \geq 1$, such that for all m, with $0 \leq m < k$, either (i_m, i_{m+1}) or (i_{m+1}, i_m) is a link in E. A chain $i_0, \ldots, i_k$ is *simple* if $i_j \neq i_\ell$ for $0 \leq j$, $\ell \leq k$, except possibly for $j = 0$ and $\ell = k$. A *path* is a chain $i_0, \ldots, i_k$ such that for all m satisfying $0 \leq m < k$, the link (i_m, i_{m+1}) is in E.

A chain $c = i_0, \ldots, i_k$, where $k \geq 1$, is said to be *closed* if $i_k = i_0$. A *circuit* is a closed chain, and a *cycle* is a closed path. If G has no cycle, then it is *acyclic*.

Graph G is defined to be *routable to destination* D if its destination is D, it is connected, it has no self-loops, and for each (i, j) in E, (j, i) is not in E.[2] When the destination is clear from the context we simply say the graph is routable.

Given two routable graphs $G_1 = \langle V_1 \cup \{D\}, E_1\rangle$ and $G_2 = \langle V_2 \cup \{D\}, E_2\rangle$, G_2 is called a *reorientation* of G_1 if G_1 and G_2 have the same undirected support, in the sense that $V_1 = V_2$, and there is a bijection f from E_1 to E_2 such that $f((i, j))$ is either (i, j) or (j, i). The notion of being a reorientation is symmetric, in that G_1 is a reorientation of G_2 if and only if G_2 is a reorientation of G_1.

Graph G is said to be *D(estination)-oriented* if it has the property that there exists at least one path from each node in V to D. Then we consider the following problem $\mathcal{P}_D$:

$\mathcal{P}_D$: Given a graph G routable to D, find a reorientation of G that is D-oriented.

As shown in [10], if G is acyclic, D-orientation can be characterized by the set of sinks in G.

PROPOSITION 2.1. *Let G be an acyclic graph that is routable to node D. The following conditions are equivalent:* (i) *G is D-oriented,* (ii) *node D is the one and only sink of G.*

This proposition leads to a promising strategy for solving $\mathcal{P}_D$. First, there is a local criterion that implies that the graph is not D-oriented, namely whether a node other than D is a sink. The second part of the strategy is "fighting" sinks — that is, changing the direction of links incident to sinks — while maintaining acyclicity.

In order to implement this strategy in a distributed setting, we make the assumption that a process is associated with each node of the graph, and for each node i in V, the process associated with i can both (a) determine the direction of all the links incident to i, and (b) change the direction of all incoming links incident to i. In this case, any process associated with a sink can locally detect that the graph is not D-oriented and then reverse some of its incoming links in order to no longer be a sink. It remains to verify that this scheme maintains acyclicity and terminates.

In what follows, we work within the context of (a) and (b), and we identify a node i with the process associated with i. Based on (a) and (b), we study various distributed algorithms, according to the link reversal strategy used by sink nodes. We consider the *asynchronous executions* of these link reversal algorithms: if node i is a sink at some moment in an execution, then i is not required to execute a reversal immediately,

[2] The routing algorithms under consideration assign dynamically changing virtual directions to the edges of an undirected graph that models bidirectional communication channels between nodes. Thus it does not make sense for both (i, j) and (j, i) to be in the corresponding directed graph.

but it has only to do so eventually. Hence, a link reversal algorithm admits multiple asynchronous executions depending on the scheduler. Note that because two sinks cannot be neighbors, all the links incident to some sink node i remain stable until i makes a reversal.

2.2. The order-based approach by Gafni and Bertsekas. Following the above strategy, Gafni and Bertsekas [10] provided two distributed algorithms, called *Full Reversal* and *Partial Reversal*:

Full Reversal: At each iteration, some sinks other than D take steps. Each sink that takes a step reverses the direction of all its incident links.

Partial Reversal: Each node i maintains a list containing incident links. Initially, the list is empty. Each time a neighbor j of i makes a reversal, i adds the link between i and j to the list. At each iteration, some sinks other than D take steps. Each sink that takes a step reverses the directions of all its incident links that do not appear in its list, and then empties its list. If there is no such incident link (that is, if the list is "full"), it reverses the directions of all its incident links, and then empties its list.

It is easy to see that in every step of Full Reversal, acyclicity is maintained. In sharp contrast, an elementary proof that Partial Reversal maintains acyclicity is not obvious, and to the best of our knowledge no such proof existed before the proof given in the preliminary version of this paper [7]. (Since then, an alternative proof has been proposed in [22].) Indeed, the argument for proving acyclicity given in [10] is not direct: Gafni and Bertsekas first stated that Full Reversal and Partial Reversal are specializations of a general solution to an order-based problem that is dual to $\mathcal{P}_D$, and then trivially derived acyclicity from the antisymmetry property of any ordering.

In more detail, their approach consists in embedding the directed graph into a total order. Node i maintains a variable, denoted $\mathtt{h}_i$ (for height), whose values are drawn from a totally ordered set $(A, <)$. The ordering between heights determines the *directions* of links by the following basic rule: *if* $\mathtt{h}_i < \mathtt{h}_j$*, then the link connecting* i *and* j *is directed from* j *to* i. Hence, sinks correspond to local minima with respect to heights, and so each node $i \in V$ $(i \neq D)$ is assigned to increase its height, when the latter is a local minimum, according to some function g_i; the height of D never changes. Under the assumption that each node can read the heights of all its neighbors, this defines a distributed algorithm. The class of algorithms obtained by varying the total ordering $(A, <)$ and the update functions g_i is denoted by IH (for Increasing Heights). Gafni and Bertsekas then assumed A to be unbounded and placed a condition on the asymptotic behavior of each g_i. They proved that under these conditions, each IH algorithm solves $\mathcal{P}_D$, and that the resulting D-oriented graph depends only on the input graph.

Under the assumption of distinct identification numbers for nodes, Gafni and Bertsekas claimed that Full Reversal and Partial Reversal coincide with two specific IH algorithms. Namely, Full Reversal is expressed by representing the height of node i as a pair (α_i, id_i) with α_i being an integer and id_i being the unique identifier of i. The relation $<$ is the lexicographical order, and the functions g_i basically correspond to transforming local minima into local maxima. For Partial Reversal, the height of i is a triple $(\alpha_i, \beta_i, id_i)$ where α_i and β_i are integers, and initially, $\alpha_i = 0$ for any node i. The relation $<$ is also the lexicographical order, and the functions g_i transform any local minimum into an intermediate value (not necessarily a local maximum). As already mentioned in the introduction, while in the case of the pair algorithm, the correspondence with Full Reversal is clear, it is more difficult to understand why

the triple algorithm actually implements Partial Reversal, and indeed, no proof of the correspondence has been given in any work on the pair and triple algorithms [10, 3, 4].

3. The link reversal algorithm. In this section, we present a graph algorithm, which we call the *link reversal* algorithm (or LR for short), that encompasses both Full Reversal and Partial Reversal. To explain our graph-based approach, let us first examine more closely how Partial Reversal runs: during each iteration, the directions of some links are reversed by sinks and links are added and removed from the nodes' lists. Starting from an initial graph G_0, the iterations therefore produce a sequence of directed connected graphs with the same support that we denote by $G_0, \ldots, G_\ell, \ldots$. Further, for each node i, we obtain a sequence of lists containing links denoted by $i.list_0, \ldots, i.list_\ell, \ldots$. Consider the graph and the lists after the ℓth iteration. One observes that for any two neighbors i and j, i is in $j.list_\ell$ only if the link (i, j) is directed from i to j in G_ℓ. Hence, for any ℓ, after the ℓth iteration, one cannot have $i \in j.list_\ell$ and $j \in i.list_\ell$. Therefore, for each directed link from i to j in G_ℓ, there are only two possible cases: on the one hand, $i \notin j.list_\ell$ and $j \notin i.list_\ell$, or on the other hand, $i \in j.list_\ell$ and $j \notin i.list_\ell$. Our basic idea is to code these two cases with labels $\mathbb{0}$ and $\mathbb{1}$ on the link from i to j accordingly. We thus propose an algorithm which works on directed graphs with *binary* labels for links, contrary to the IH algorithms which assign *unbounded* labels (so-called heights) to nodes.

In more detail, we consider a graph $G = \langle V \cup \{D\}, E\rangle$ and a function μ from E to $\{\mathbb{0}, \mathbb{1}\}$, which gives the label of either $\mathbb{0}$ or $\mathbb{1}$ to each link of G. If a link is labeled with $\mathbb{1}$, we say that it is marked; otherwise, it is said to be unmarked. We denote the resulting link-labeled graph as $G^\dagger = \langle G, \mu\rangle$. The dagger superscript will be used throughout to indicate such a link-labeled graph. If G is routable, then $G^\dagger$ is said to be a link-labeled routable graph. All the definitions given in section 2.1 above can be clearly extended to link-labeled graphs.

The LR algorithm. For each sink node i other than D, one can apply the following mutually exclusive rules:

R1: If at least one link incident on i is labeled $\mathbb{0}$, then all the links incident on node i that are labeled with $\mathbb{0}$ are reversed, the other incident links are not reversed, and the labels on all the incident links are flipped.

R2: If all the links incident on node i are labeled $\mathbb{1}$, then all the links incident on i are reversed, but none of their labels is changed.

Note that in the context of networks with nodes being able to change the directions and the labels of their incoming links, the LR algorithm admits a distributed implementation over the network. Then we say that node i *takes a step* when R1 or R2 is applied at i. Given a link-labeled routable graph $G^\dagger$, any nonempty set S of sinks in $G^\dagger$ is said to be *applicable to* $G^\dagger$, as each node in S may "simultaneously" take a step. Since two neighboring nodes in $G^\dagger$ cannot both be a sink, the resulting graph depends only on S and $G^\dagger$ and is denoted by $S.G^\dagger$. In case $S = \{i\}$, we write $i.G^\dagger$ for short, instead of $\{i\}.G^\dagger$. By induction, we easily generalize the notion of applicability to $G^\dagger$ for a sequence $\mathbf{S}$ of a nonempty set of nodes, and we denote the resulting graph by $\mathbf{S}.G^\dagger$.

An *execution* of the LR algorithm from a link-labeled routable graph $G_0^\dagger$ is a sequence $G_0^\dagger$, S_1, $G_1^\dagger, \ldots$ $G_{t-1}^\dagger$, $S_t, \ldots$ of alternating link-labeled routable graphs and sets of nodes satisfying the following conditions:

1. For each integer t, $t \geq 1$, S_t is a subset of V that is applicable to $G_{t-1}^\dagger$.
2. For each integer t, $t \geq 1$, $G_t^\dagger$ equals $S_t.G_{t-1}^\dagger$.
3. If the sequence is finite, then it ends with a link-labeled graph that contains no sinks other than D.

For each $t \geq 0$, the transition from $G^\dagger_{t-1}$ to $G^\dagger_t$ is called *iteration* t, and node i in S_t is said to *take a step at iteration* t.

Since each S_t can be any nonempty subset of the sink nodes in $G^\dagger_{t-1}$ other than D, there are multiple possible LR executions starting from the same initial graph: the flexibility for the sets S_t actually captures asynchronous behaviors of the nodes. We may thus model a range of situations, with one extreme being the maximally concurrent situation in which all sinks take a step at each iteration, and the other extreme being a single node taking a step in each iteration. An LR execution $G^\dagger_0$, S_1, $G^\dagger_1, \ldots$ $G^\dagger_{t-1}$, $S_t, \ldots, G^\dagger_k$ that exhibits the maximal amount of parallelism is called *greedy*; i.e., for all t, $1 \leq t \leq k$, S_t consists of all the sinks in $G^\dagger_{t-1}$. Since the algorithm is deterministic, there is exactly one greedy LR execution for a given initial link-labeled routable graph $G^\dagger_0$.

Observe that if all labels are initially equal to $\mathbb{1}$, then all links remain marked during the whole execution, and only rule R2 is executed. The LR algorithm thus coincides with Full Reversal. If initially all labels are $\mathbb{0}$, the above description of the Partial Reversal runs shows that LR executes as Partial Reversal. Our approach, which consists in varying only the initial link labeling, thus unifies Full Reversal and Partial Reversal.

3.1. Convergence and delay-insensitivity. In this section, we prove that LR converges, i.e., from any given routable link-labeled graph, every execution of LR is finite. Moreover, we show that the algorithm is *delay-insensitive* in the sense that the final directed graph generated by the algorithm depends only on the input graph and not on the scheduler. Note that by definition of the LR rules, the final graph has no sink node, except possibly D.

THEOREM 3.1. *Any* LR *execution from a routable link-labeled graph is finite.*

Proof. Assume for the sake of contradiction that there is an LR execution that is infinite. Let X be the set of nodes that take an infinite number of steps, and let Y be its complementary set in $V \cup \{D\}$. By assumption, $X \neq \emptyset$. As node D takes no step, D is in Y and so $Y \neq \emptyset$. Since the underlying undirected graph in the whole execution is connected, there is a node $i \in X$ and a node $j \in Y$ such that either (i, j) or (j, i) is a link of this graph.

Suppose, after the last step of j, that the link between j and i is directed toward j. Then subsequently i never becomes a sink, contradicting the fact that i is in X and thus takes infinitely many steps.

Suppose, after the last step of j, that the link between j and i is directed toward i. Then after at most two subsequent steps by i, the link is directed toward j, and again we get a contradiction. ∎

As for delay-insensitivity, we first prove the following commutativity property.

LEMMA 3.2. *Let S be any subset of V, and let $S = S_1 \cup S_2$ be a bipartitioning of S with S_1 and S_2 both nonempty. If each node in S is a sink of $G^\dagger$, then S_1 is applicable to $S_2.G^\dagger$, S_2 is applicable to $S_1.G^\dagger$, and $S.G^\dagger = S_1.(S_2.G^\dagger) = S_2.(S_1.G^\dagger)$.*

Proof. The result follows at once from rules R1 and R2 since $S_1 \cap S_2 = \emptyset$, and two neighboring nodes cannot both be in S. ∎

THEOREM 3.3. *Let $G^\dagger_0$ be a routable link-labeled graph. A node i in $G^\dagger_0$ takes the same number of steps in any* LR *execution from initial graph $G^\dagger_0$. Moreover, the final graph depends only on $G^\dagger_0$.*

Proof. Let $\mathcal{E} = G^\dagger_0, S_1, G^\dagger_1, S_2, \ldots, G^\dagger_k$ be an LR execution from the initial graph $G^\dagger_0$ in which a sink i other than D is delayed from taking a step; i.e., for

some t_0, $1 \leq t_0 \leq k-1$, node i is a sink of $G^\dagger_{t_0-1}$, i is not in S_{t_0}, and i is in S_{t_0+1}. We now construct another execution $\mathcal{E}'$ in which we move up by one the iteration at which sink i takes its step. For that, let $\tilde{S}_1, \tilde{S}_2, \ldots$ be the sequence of nonempty subsets of V defined as follows:

1. If $S_{t_0+1} = \{i\}$, then

$$\tilde{S}_t = \begin{cases} S_t & \text{if } 1 \leq t \leq t_0 - 1, \\ S_{t_0} \cup \{i\} & \text{if } t = t_0, \\ S_{t+1} & \text{if } t_0 + 1 \leq t \leq k-1; \end{cases}$$

2. else ($S_{t_0+1} \neq \{i\}$), then

$$\tilde{S}_t = \begin{cases} S_t & \text{if } 1 \leq t \leq t_0 - 1 \text{ or } t_0 + 2 \leq t \leq k, \\ S_{t_0} \cup \{i\} & \text{if } t = t_0, \\ S_{t_0+1} \setminus \{i\} & \text{if } t = t_0 + 1. \end{cases}$$

By Lemma 3.2, the sequence $\tilde{S}_1, \tilde{S}_2, \ldots$ is applicable to $G^\dagger_0$ and results in an LR execution $\mathcal{E}'$ from $G^\dagger_0$. Moreover, each node takes the same number of steps in $\mathcal{E}'$ as in $\mathcal{E}$, and the final graphs are the same.

We then define the relation $\succ$ on the (finite) set of LR executions from $G^\dagger_0$ as the smallest transitive relation satisfying $\mathcal{E} \succ \mathcal{E}'$. We immediately observe that for the (unique) greedy LR execution $\mathcal{E}_g$ from $G^\dagger_0$, there is no execution $\mathcal{E}$ such that $\mathcal{E}_g \succ \mathcal{E}$. Moreover, the above argument shows that for any LR execution $\mathcal{E}$ from $G^\dagger_0$ other than the greedy execution $\mathcal{E}_g$, there exists an LR execution $\mathcal{E}'$ with $\mathcal{E} \succ \mathcal{E}'$. Therefore $\mathcal{E}_g$ is the unique minimum of all LR executions from $G^\dagger_0$ with respect to $\succ$. It follows that for each LR execution $\mathcal{E}$ from $G^\dagger_0$, every node i in $G^\dagger_0$ takes the same number of steps in $\mathcal{E}$ as in $\mathcal{E}_g$, and the final graphs of $\mathcal{E}$ and $\mathcal{E}_g$ are the same. □

3.2. The acyclicity issue. We have thus shown that starting with a finite routable input graph, whether acyclic or not, the algorithm converges, i.e., reaches a state where there is no sink node other than D. However, "fighting sinks" is just one part of the required solution, and it remains to study whether acyclicity can be maintained by LR. For that, our strategy will consist in reducing certain properties of the algorithm to invariants of the graph, or more precisely to invariants of circuits.

First, we introduce some additional notation. For a circuit $C^\dagger = i_0, \ldots, i_{k-1}, i_k$, with $k \neq 0$, let $\ell^{\mathbb{1}}(C^\dagger)$ be the number of marked links (i_j, i_{j+1}), and let $r^{\mathbb{1}}(C^\dagger)$ be the number of marked links (i_{j+1}, i_j), for $0 \leq j \leq k-1$. Let $s^{\mathbb{0}}(C^\dagger)$ be the number of nodes in $C^\dagger$ with two distinct incoming unmarked links *relative to* $C^\dagger$. Then we define the following two measures $\xi^+(C^\dagger)$ and $\xi^-(C^\dagger)$:

$$\xi^+(C^\dagger) = \ell^{\mathbb{1}}(C^\dagger) + s^{\mathbb{0}}(C^\dagger) \quad \text{and} \quad \xi^-(C^\dagger) = r^{\mathbb{1}}(C^\dagger) + s^{\mathbb{0}}(C^\dagger).$$

These definitions capture two quantities that are invariant, as we show next. The intuition is that any change to $s^{\mathbb{0}}$ caused by a step is canceled out by a change to $\ell^{\mathbb{1}}$, and similarly any change to $s^{\mathbb{0}}$ is canceled out by a change to $r^{\mathbb{1}}$. The invariance of $\xi^+(C^\dagger)$ and $\xi^-(C^\dagger)$ leads to a simple proof that acyclicity is preserved by LR.

PROPOSITION 3.4. *Let $C^\dagger$ be any simple circuit in $G^\dagger$. For any node i, $\xi^+(i.C^\dagger) = \xi^+(C^\dagger)$ and $\xi^-(i.C^\dagger) = \xi^-(C^\dagger)$.*

Proof. If i is not a node of $C^\dagger$, then $C^\dagger$ remains unchanged, i.e., $i.C^\dagger = C^\dagger$, and the result immediately follows. Otherwise, i is a sink node of $C^\dagger$, and there are three cases to consider depending on the labels of the two links incident to i in $C^\dagger$.

1. None of the two links is marked. Then i executes R1 and both reverses and marks the two links. Hence,
$$\ell^{\mathbb{1}}(i.C^\dagger) = \ell^{\mathbb{1}}(C^\dagger) + 1, \quad r^{\mathbb{1}}(i.C^\dagger) = r^{\mathbb{1}}(C^\dagger) + 1, \quad s^{\mathbb{0}}(i.C^\dagger) = s^{\mathbb{0}}(C^\dagger) - 1.$$
2. The two links are marked. We then consider two subcases:
   (a) Node i executes R1 and hence does not reverse either of them and unmarks both of them. In this case, we have
   $$\ell^{\mathbb{1}}(i.C^\dagger) = \ell^{\mathbb{1}}(C^\dagger) - 1, \quad r^{\mathbb{1}}(i.C^\dagger) = r^{\mathbb{1}}(C^\dagger) - 1, \quad s^{\mathbb{0}}(i.C^\dagger) = s^{\mathbb{0}}(C^\dagger) + 1.$$
   (b) Node i executes R2 and so reverses and marks both of them. We then have
   $$\ell^{\mathbb{1}}(i.C^\dagger) = \ell^{\mathbb{1}}(C^\dagger) - 1 + 1, \quad r^{\mathbb{1}}(i.C^\dagger) = r^{\mathbb{1}}(C^\dagger) - 1 + 1, \quad s^{\mathbb{0}}(i.C^\dagger) = s^{\mathbb{0}}(C^\dagger).$$
3. If only one of the two incoming links of i that belong to $C^\dagger$ is marked, then i executes R1 and thus unmarks the marked link and reverses and marks the other one. We thus have
$$\ell^{\mathbb{1}}(i.C^\dagger) = \ell^{\mathbb{1}}(C^\dagger), \quad r^{\mathbb{1}}(i.C^\dagger) = r^{\mathbb{1}}(C^\dagger), \quad s^{\mathbb{0}}(i.C^\dagger) = s^{\mathbb{0}}(C^\dagger).$$

In all cases we check that $\ell^{\mathbb{1}}(i.C^\dagger) + s^{\mathbb{0}}(i.C^\dagger) = \ell^{\mathbb{1}}(C^\dagger) + s^{\mathbb{0}}(C^\dagger)$ and $r^{\mathbb{1}}(i.C^\dagger) + s^{\mathbb{0}}(i.C^\dagger) = r^{\mathbb{1}}(C^\dagger) + s^{\mathbb{0}}(C^\dagger)$, and we thus conclude that $\xi^+(i.C^\dagger) = \xi^+(C^\dagger)$ and $\xi^-(i.C^\dagger) = \xi^-(C^\dagger)$. □

PROPOSITION 3.5. *If $C^\dagger$ is a circuit such that*

$$\xi^+(C^\dagger) \cdot \xi^-(C^\dagger) > 0, \tag{AC}$$

then $C^\dagger$ is not a cycle.

Proof. Observe that if $C^\dagger$ is a cycle, then $s^{\mathbb{0}}(C^\dagger) = 0$, and either $\ell^{\mathbb{1}}(C^\dagger) = 0$ or $r^{\mathbb{1}}(C^\dagger) = 0$. It follows that condition (AC) is sufficient for guaranteeing that $C^\dagger$ is not a cycle. □

From Propositions 3.4 and 3.5, we immediately derive the following acyclicity result.

THEOREM 3.6. *If in the initial routable graph $G_0^\dagger$ every simple circuit satisfies condition* (AC)*, then all the directed graphs generated in any execution of* LR *starting from $G_0^\dagger$ are acyclic.*

Combining Theorems 3.1 and 3.6, we get the following corollary.

COROLLARY 3.7. *Let G_0 be any routable graph. If the initial link labeling of G_0 is such that all the simple circuits in the resulting labeled graph $G_0^\dagger$ satisfy* (AC)*, then* LR *solves $\mathcal{P}_D$ for G_0.*

In the case of Full Reversal, all labels are initially equal to $\mathbb{1}$, which gives both $\ell^{\mathbb{1}}(C^\dagger) \neq 0$ and $r^{\mathbb{1}}(C^\dagger) \neq 0$ for any circuit $C^\dagger$ that is not a cycle. In contrast, $s^{\mathbb{0}}(C^\dagger) \neq 0$ is guaranteed with the uniform $\mathbb{0}$ labeling (and so with Partial Reversal) when $C^\dagger$ is not a cycle.

More generally, condition (AC) naturally leads to the labeling policy where each node i is allowed to choose $\mu_i \in \{\mathbb{0}, \mathbb{1}\}$ for the initial labels of all its incoming links. We call this policy *locally uniform* and denote it by LU. The point of the LU policy is to ensure condition (AC) for $C^\dagger$ whenever $C^\dagger$ is not a cycle: in $C^\dagger$ there must be at least one node i with two distinct incoming links relative to $C^\dagger$. If $\mu_i = \mathbb{0}$, then $s^{\mathbb{0}}(C^\dagger) > 0$, while if $\mu_i = \mathbb{1}$, then $\ell^{\mathbb{1}}(C^\dagger) > 0$ and $r^{\mathbb{1}}(C^\dagger) > 0$.

Full Reversal and Partial Reversal correspond to the globally uniform choices; i.e., for every node i, $\mu_i = \mathbb{1}$ and $\mu_i = \mathbb{0}$, respectively. The LU initial labeling thus builds a continuum between Full Reversal and Partial Reversal, including other link reversal strategies that we analyze and compare in sections 5.2 and 6.

4. Work complexity. We now analyze the *work complexity* of the LR algorithm. For each execution of LR, we define the *work complexity of a node* in an execution to be the number of reversals performed by that node and define the *global work complexity* to be the sum, over all nodes, of the work complexity of each node. By Theorem 3.3, the work complexity depends only on the initial link-labeled graph and not on executions.

We first extend the definitions for circuits introduced in section 3.2 to chains. Since the initial graph is connected, for each node i there is a simple chain connecting i and D. As LR does not affect the support of the initial graph and so does not affect the connectivity, there is a simple chain connecting D and i with a stable support for each i during any execution of LR. Such a simple chain is called a *D-chain from i*. If each link along a D-chain is directed toward D, then the D-chain is said to be a *D-path*. Further, we will denote by $\mathcal{C}(i, G^\dagger)$ the set of all D-chains from i with the link directions and the link labels inherited from $G^\dagger$.

Due to the statement of the D-orientation problem, we say that for a D-chain from i, the natural orientation is from i to D. For any D-chain $C^\dagger$, we define the *residue* $Res(C^\dagger)$ to be 1 if the first link (with respect to the natural orientation toward D) in $C^\dagger$ is unmarked and against the orientation, and 0 otherwise. As for circuits, we consider the following two quantities $\xi^+(C^\dagger) = \ell^{\mathbb{1}}(C^\dagger) + s^{\mathbb{0}}(C^\dagger)$ and $\xi^-(C^\dagger) = r^{\mathbb{1}}(C^\dagger) + s^{\mathbb{0}}(C^\dagger)$, and we define two additional quantities

$$\delta(C^\dagger) = \xi^+(C^\dagger) - \xi^-(C^\dagger),$$
$$\gamma(C^\dagger) = \xi^+(C^\dagger) + \xi^-(C^\dagger) + Res(C^\dagger).$$

From the very definitions of $\delta(C^\dagger)$ and $\gamma(C^\dagger)$, we immediately derive the following proposition.

PROPOSITION 4.1. *For any D-chain $C^\dagger$, we have $\delta(C^\dagger) \leq \gamma(C^\dagger)$. Moreover, if $C^\dagger$ is a D-path, then $\delta(C^\dagger) = \gamma(C^\dagger)$.*

The previous proposition naturally leads us to consider the nonnegative quantity

$$\omega(C^\dagger) = \gamma(C^\dagger) - \delta(C^\dagger).$$

Now we study how $\gamma(C^\dagger)$ and $\delta(C^\dagger)$ evolve along with executing LR.

PROPOSITION 4.2. *For any sink node i and any D-chain $C^\dagger$, we have*

$$\gamma(i.C^\dagger) = \gamma(C^\dagger).$$

Proof. Let j and e be the first node and the first link in $C^\dagger$, respectively. There are three possible cases:

1. Node i does not belong to $C^\dagger$. Then $i.C^\dagger = C^\dagger$, and thus $\gamma(i.C^\dagger) = \gamma(C^\dagger)$ trivially holds.
2. Node i belongs to $C^\dagger$, and $i \neq j$. The same proof as for Proposition 3.4 shows that in this case, $\xi^+(i.C^\dagger) = \xi^+(C^\dagger)$ and $\xi^-(i.C^\dagger) = \xi^-(C^\dagger)$. Consequently, it remains to show that $Res(i.C^\dagger) = Res(C^\dagger)$. If i is not a neighbor of j, then the result follows immediately from the fact that in LR, nodes change the state of incident links only. Otherwise, i is a neighbor of j, and the link e

between i and j is incoming to i since i is a sink node. Hence, $Res(C^\dagger) = 0$. Depending on the label of e in $C^\dagger$ and on whether R1 or R2 is executed, the link e in $i.C^\dagger$ is either unmarked and incoming to i, or it is outgoing and marked. In both cases we have $Res(i.C^\dagger) = 0$ and thus $Res(i.C^\dagger) = Res(C^\dagger)$ as needed.

3. Node i is in $C^\dagger$ and $i = j$. There are three possible subcases:
   (a) The link e is unmarked in $C^\dagger$. Node i executes R1 and so reverses and marks e. It follows that
   $$Res(i.C^\dagger) = Res(C^\dagger) - 1, \quad \ell^{\mathbb{1}}(i.C^\dagger) = \ell^{\mathbb{1}}(C^\dagger) + 1, \quad r^{\mathbb{1}}(i.C^\dagger) = r^{\mathbb{1}}(C^\dagger).$$
   As e is marked in $i.C^\dagger$, its reversal does not modify the quantity $s^{\mathbb{0}}$, that is, $s^{\mathbb{0}}(i.C^\dagger) = s^{\mathbb{0}}(C^\dagger)$.
   (b) The link e is marked in $C^\dagger$ and i executes R1. According to this rule, i only unmarks e. In this case, $Res(i.C^\dagger) = Res(C^\dagger) + 1$. Moreover, we have
   $$\ell^{\mathbb{1}}(i.C^\dagger) = \ell^{\mathbb{1}}(C^\dagger), \quad r^{\mathbb{1}}(i.C^\dagger) = r^{\mathbb{1}}(C^\dagger) - 1, \quad s^{\mathbb{0}}(i.C^\dagger) = s^{\mathbb{0}}(C^\dagger).$$
   (c) The link e is marked in $C^\dagger$ and i executes R2. Node i reverses and marks e, and so $Res(i.C^\dagger) = Res(C^\dagger)$. Moreover,
   $$\ell^{\mathbb{1}}(i.C^\dagger) = \ell^{\mathbb{1}}(C^\dagger) + 1, \quad r^{\mathbb{1}}(i.C^\dagger) = \ell^{\mathbb{1}}(C^\dagger) - 1, \quad s^{\mathbb{0}}(i.C^\dagger) = s^{\mathbb{0}}(C^\dagger).$$

In all cases we conclude that $\gamma(i.C^\dagger) = \gamma(C^\dagger)$. □

Since $\gamma(C^\dagger)$ never changes, the key point now is to show that $\delta(C^\dagger)$ regularly increases during any execution of LR. To do that, we are led to partition the set of nodes V into three disjoint subsets: $\mathcal{S}_{\mathbb{0}}(G^\dagger)$ is the set of nodes, all of whose incident links are unmarked and incoming, $\mathcal{S}_{\mathbb{1}}(G^\dagger)$ is the set of nodes with no unmarked incoming link, and $\mathcal{D}(G^\dagger)$ consists of the remaining nodes.

PROPOSITION 4.3. *Let i be a sink node and $C^\dagger$ be any D-chain in $G^\dagger$. If i is not the first node in $C^\dagger$, then $\delta(i.C^\dagger) = \delta(C^\dagger)$. Otherwise, $C^\dagger$ is in $\mathcal{C}(i, G^\dagger)$, and $\delta(i.C^\dagger)$ is given by the following rule: if i is in $\mathcal{S}_{\mathbb{1}}(G^\dagger)$, then $\delta(i.C^\dagger) = \delta(C^\dagger) + 2$; else (i.e., $i \in \mathcal{S}_{\mathbb{0}}(G^\dagger) \cup \mathcal{D}(G^\dagger)$), $\delta(i.C^\dagger) = \delta(C^\dagger) + 1$.*

Proof. If i does not belong to $C^\dagger$, then $\delta(i.C^\dagger) = \delta(C^\dagger)$ trivially holds. Moreover, as shown in the proof of Proposition 4.2, we have $\xi^+(i.C^\dagger) = \xi^+(C^\dagger)$ and $\xi^-(i.C^\dagger) = \xi^-(C^\dagger)$ whenever i is a node in $C^\dagger$ but is not the first one. This also gives $\delta(i.C^\dagger) = \delta(C^\dagger)$ in this case.

When i is the first node in $C^\dagger$, there are two cases to consider.

1. $i \in \mathcal{S}_{\mathbb{1}}(G^\dagger)$. In this case, i executes R2 and thus reverses and marks all its incoming links. It follows that $\ell^{\mathbb{1}}(i.C^\dagger) = \ell^{\mathbb{1}}(C^\dagger)+1$ and $r^{\mathbb{1}}(i.C^\dagger) = r^{\mathbb{1}}(C^\dagger)-1$. Thereby, $\delta(i.C^\dagger) = \delta(C^\dagger) + 2$.
2. $i \in \mathcal{S}_{\mathbb{0}}(G^\dagger) \cup \mathcal{D}(G^\dagger)$. Then i necessarily executes R1, and so unmarks but does not reverse all of its marked incoming links, and reverses and marks the others. Thus we have either
   $$\ell^{\mathbb{1}}(i.C^\dagger) = \ell^{\mathbb{1}}(C^\dagger) + 1 \text{ and } r^{\mathbb{1}}(i.C^\dagger) = r^{\mathbb{1}}(C^\dagger)$$
   or
   $$\ell^{\mathbb{1}}(i.C^\dagger) = \ell^{\mathbb{1}}(C^\dagger) \text{ and } r^{\mathbb{1}}(i.C^\dagger) = r^{\mathbb{1}}(C^\dagger) - 1$$
   according to the label of the first link in $C^\dagger$. In both cases, $\delta(i.C^\dagger) = \delta(C^\dagger)+1$.

In all cases the proposition holds. □

Now we study how this node partitioning evolves during the execution of LR.

PROPOSITION 4.4. *If i is a sink node in $G^\dagger$, then $\mathcal{S}_\mathbb{0}(i.G^\dagger) = \mathcal{S}_\mathbb{0}(G^\dagger) \setminus \{i\}$, $\mathcal{S}_\mathbb{1}(i.G^\dagger) = \mathcal{S}_\mathbb{1}(G^\dagger) \cup \big(\mathcal{S}_\mathbb{0}(G^\dagger) \cap \{i\}\big)$, and $\mathcal{D}(i.G^\dagger) = \mathcal{D}(G^\dagger)$.*

Proof. Clearly, the step taken by i in LR cannot affect, in either direction or label, the links incident to nodes that are outside the neighborhood of i. So let j be any node in the neighborhood of i. We study where j migrates in the new node partitioning of $i.G^\dagger$. There are two possible cases:

1. $j = i$. From the LR rules, it follows that $i \in \mathcal{D}(i.G^\dagger)$ or $i \in \mathcal{S}_\mathbb{1}(i.G^\dagger)$ depending on whether $i \in \mathcal{D}(G^\dagger)$ or not.
2. There is a link e from j to i. Since i is a sink node, it follows that $j \notin \mathcal{S}_\mathbb{0}(G^\dagger)$, i.e., $j \in \mathcal{S}_\mathbb{1}(G^\dagger)$ or $j \in \mathcal{D}(G^\dagger)$. In $i.G^\dagger$, there are only two possible configurations for e: e remains an incoming link for i and is unmarked, or e is an outgoing link for i and is marked. In both cases, the new configuration of e cannot modify the status of j, i.e., if $j \in \mathcal{S}_\mathbb{1}(G^\dagger)$, then $j \in \mathcal{S}_\mathbb{1}(v.G^\dagger)$, and if $j \in \mathcal{D}(G^\dagger)$, then $j \in \mathcal{D}(i.G^\dagger)$.

Thereby, when i makes a reversal, our node partitioning remains unchanged, except in the migration of i from $\mathcal{S}_\mathbb{0}$ to $\mathcal{S}_\mathbb{1}$ in the case when i is initially a sink, and all links incident to i are unmarked. ∎

Both Propositions 4.3 and 4.4 trivially extend to the case when several sink nodes make simultaneous reversals:

$$\begin{aligned}
\mathcal{S}_\mathbb{0}(S.G^\dagger) &= \mathcal{S}_\mathbb{0}(G^\dagger) \setminus S,\\
\mathcal{S}_\mathbb{1}(S.G^\dagger) &= \mathcal{S}_\mathbb{1}(G^\dagger) \cup \big(\mathcal{S}_\mathbb{0}(G^\dagger) \cap S\big), \text{ and}\\
\mathcal{D}(S.G^\dagger) &= \mathcal{D}(G^\dagger).
\end{aligned}$$

We need one more definition before stating our main result. For each node i, we consider the set of D-chains from i and define the quantity

$$\omega(i, G^\dagger) = \min_{C^\dagger \in \mathcal{C}(i,G^\dagger)} \big(\omega(C^\dagger)\big)\,.$$

As an immediate consequence of Propositions 4.2 and 4.3, we derive the following lemma.

LEMMA 4.5. *The D-chains that achieve the minimum value of $\omega(C^\dagger)$ are the same throughout the execution.*

We can now prove our major theorem.

THEOREM 4.6. *In any execution of* LR *starting from some routable graph $G_0^\dagger$ whose simple circuits all satisfy* (AC)*, the number of reversals made by any node i is $\omega(i, G_0^\dagger)/2 + 1/2$, $\omega(i, G_0^\dagger)/2$, or $\omega(i, G_0^\dagger)$ if $i \in \mathcal{S}_\mathbb{0}(G_0^\dagger)$, $i \in \mathcal{S}_\mathbb{1}(G_0^\dagger)$, or $i \in \mathcal{D}(G_0^\dagger)$, respectively.*

Proof. Let us consider the following finite sequence of nonnegative integers $\omega(i, G_0^\dagger), \ldots, \omega(i, G_k^\dagger)$, as well as the subsequence $\omega(i, G_0^\dagger), \ldots, \omega(i, G_{k_i}^\dagger)$, obtained when considering only the steps in which i makes a reversal. From Lemma 4.5, and Propositions 4.2 and 4.3, it follows that the sequence $\omega(i, G_0^\dagger), \ldots, \omega(i, G_k^\dagger)$ is nonincreasing and that the subsequence $\omega(i, G_0^\dagger), \ldots, \omega(i, G_{k_i}^\dagger)$ is decreasing. More precisely, we deduce from Propositions 4.2, 4.3, and 4.4 that the subsequence decreases regularly: it decreases either by 1 the first time and then by 2 each later time if $i \in \mathcal{S}_\mathbb{0}(G_0^\dagger)$, or by 2 each time if $i \in \mathcal{S}_\mathbb{1}(G_0^\dagger)$, or otherwise ($i \in \mathcal{D}(G_0^\dagger)$) by 1 each time. Moreover, the first value of this sequence is $\omega(i, G_0^\dagger)$. By Proposition 4.1, $\omega(i, G_k^\dagger) = 0$ since $G_k^\dagger$ is D-oriented. The subsequence thus decreases from $\omega(i, G_0^\dagger)$ to 0, and then we deduce the number of reversals made by i. ∎

In a more convenient form, Theorem 4.6 reads as follows: for each link-labeled routable graph $G_0^\dagger$ whose simple circuits all satisfy (AC), the number of link reversals made by node i in any execution starting from $G_0^\dagger$ is $\min\left\{\Phi(C^\dagger) : C^\dagger \in \mathcal{C}(i, G_0^\dagger)\right\}$, where

$$\Phi(C^\dagger) = \begin{cases} r^{\mathbb{1}}(C^\dagger) + s^{\mathbb{0}}(C^\dagger) + Res(C^\dagger) & \text{if } i \in \mathcal{S}_{\mathbb{0}}(G_0^\dagger) \cup \mathcal{S}_{\mathbb{1}}(G_0^\dagger), \\ 2r^{\mathbb{1}}(C^\dagger) + 2s^{\mathbb{0}}(C^\dagger) + Res(C^\dagger) & \text{if } i \in \mathcal{D}(G_0^\dagger). \end{cases}$$

5. Applications to LU labelings.

5.1. Work complexity of Full Reversal and Partial Reversal. From Theorem 4.6, we deduce the exact formulas for the work complexities of Full Reversal and Partial Reversal. For that, we first introduce additional notation. Given a DAG G, for any node i we denote the set of all D-chains from i by $\mathcal{C}(i, G)$; for any $C \in \mathcal{C}(i, G)$, let $r(C)$ be the number of links in C that are directed away from D, and let $s(C)$ be the number of nodes in C which have two distinct incoming links relative to C. We also define $Res(C)$, the residue of C, to be 1 if the first link of C is directed toward i, and to be 0 otherwise. Note that in the case of the uniform labeling $\mathbb{1}$ (Full Reversal), both $\mathcal{S}_{\mathbb{0}}$ and $\mathcal{D}$ are empty, and so $\mathcal{S}_{\mathbb{1}}$ contains all the nodes other than D. As for the uniform labeling $\mathbb{0}$ (Partial Reversal), $\mathcal{S}_{\mathbb{0}}$ and $\mathcal{S}_{\mathbb{1}}$ consist of all the sinks and all the sources, respectively.

COROLLARY 5.1. *Let G_0 be a routable acyclic graph.*

1. *The number of reversals made by node i in any Full Reversal execution from G_0 is*

$$\min\left\{r(C) \colon C \in \mathcal{C}(i, G_0)\right\}.$$

2. *The number of reversals made by node i in any Partial Reversal execution from G_0 is*

$$\begin{array}{ll} \min\left\{s(C) + Res(C) \colon C \in \mathcal{C}(i, G_0)\right\} & \text{if } i \text{ is a sink or a source in } G, \\ \min\left\{2s(C) + Res(C) \colon C \in \mathcal{C}(i, G_0)\right\} & \text{otherwise.} \end{array}$$

As a consequence of this corollary, we now understand what properties of the input graphs generate steps for Full Reversal and for Partial Reversal. In particular, this leads us to design the two (directed) chains in Figure 5.1 for which there is a large discrepancy between the global work complexities of Full Reversal and Partial Reversal, respectively. Indeed, for the upper chain, the number of steps taken by Full Reversal is $\frac{n\cdot(n+1)}{2}$, while it is n for Partial Reversal. In contrast, the global work complexity of Full Reversal for the lower chain is n, while that of Partial Reversal is $2n - 2$. Hence, Partial Reversal is worse by about a factor of 2 for this graph. The results of section 6.3 imply that Partial Reversal is worse by at most a factor of 2 in any graph.

5.2. Comparing LU link labelings. The findings discussed above do not allow us to conclusively decide whether Full Reversal or Partial Reversal is the better algorithm. Interestingly, most of the literature on link reversal routing (see, e.g., [21, 15, 9]) uses algorithms that operate similarly to Partial Reversal. So there seems to be an implicit understanding that Partial Reversal is better. However, there is no formal approach underpinning this intuition.

We now consider the 2^n possible initial LU link labelings, and we compare the work complexity for two *adjacent* LU labelings, i.e., two LU labelings that differ on

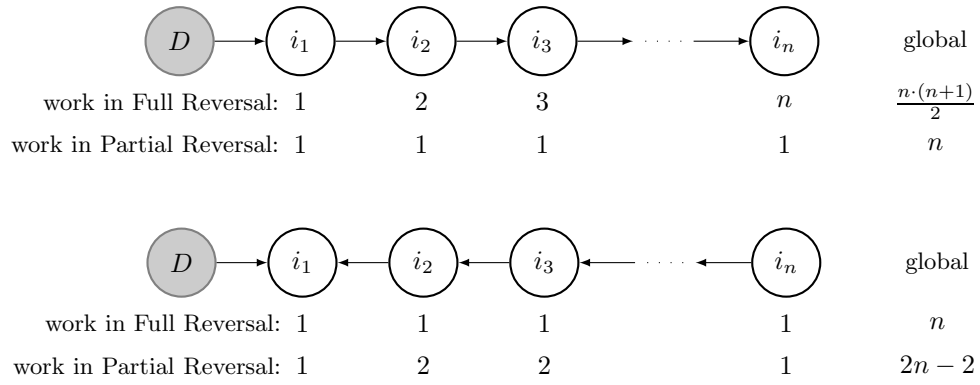


FIG. 5.1. *Example chains showing that neither Full Reversal nor Partial Reversal performs better in general.*

the labels of the incoming links of a single node. Step by step, we can then compare Full Reversal (all links are labeled with $\mathbb{1}$) and Partial Reversal (all links are labeled with $\mathbb{0}$).

Let the set of vectors $M = \{\mathbb{0}, \mathbb{1}\}^n$; $\vec{\mathbb{0}}$ and $\vec{\mathbb{1}}$ denote the vectors in M whose components all are $\mathbb{0}$ and $\mathbb{1}$, respectively. For any vector $\vec{\mu} \in M$, let μ_i denote its ith component. We consider a routable acyclic graph $G = \langle V \cup \{D\}, E\rangle$. The link-labeled graph in which, for each node $i \in V$, all incoming links of i are labeled with μ_i, and the links incoming to D are labeled with $\mathbb{0}$,[3] is denoted by $G^{\vec{\mu}}$. Similarly, let $C^{\vec{\mu}}$ denote the labeled D-chain from some node i in $G^{\vec{\mu}}$ when C is a D-chain from i. As every circuit of $G^{\vec{\mu}}$ satisfies (AC), Theorem 4.6 provides node i's work in executions of LR starting with $G^{\vec{\mu}}$, denoted by $\sigma_i(\vec{\mu})$.

We now investigate the influence of i's choice for μ_i first on its own work, and then on the work of the other nodes. We denote by $(\vec{\mu}_{-i}, s)$ the vector in M which results from replacing the ith component of $\vec{\mu}$ by s. Further, we call a node i a *good node* if there is a path from i to D; otherwise, i is called a *bad node*.

First, observe that for any node i and any vector $\vec{\mu} \in M$, we have $i \in \mathcal{S}_{\mathbb{1}}(G^{(\vec{\mu}_{-i},\mathbb{1})})$. Moreover, if i is neither a sink nor a source, then $i \in \mathcal{D}(G^{(\vec{\mu}_{-i},\mathbb{0})})$. Combining this remark with Theorem 4.6, we easily show the following proposition which formally establishes that, from a selfish viewpoint, it is in the interest of i to choose $\mu_i = \mathbb{1}$.

PROPOSITION 5.2. *For every node $i \in V$, and any vector $\vec{\mu} \in M$, we have*

$$\sigma_i(\vec{\mu}_{-i}, \mathbb{0}) \geq \sigma_i(\vec{\mu}_{-i}, \mathbb{1}).$$

Further, if i is a sink or a source, then

$$\sigma_i(\vec{\mu}_{-i}, \mathbb{0}) = \sigma_i(\vec{\mu}_{-i}, \mathbb{1}).$$

Proof. There are three cases to consider:

1. If i is a source in G, then $G^{(\vec{\mu}_{-i},\mathbb{0})} = G^{(\vec{\mu}_{-i},\mathbb{1})}$ and the proposition follows.
2. If i is a sink in G, then for all D-chains from i, denoted by C, we have

$$Res(C^{(\vec{\mu}_{-i},\mathbb{0})}) = Res(C^{(\vec{\mu}_{-i},\mathbb{1})}) + 1,$$
$$r^{\mathbb{1}}(C^{(\vec{\mu}_{-i},\mathbb{0})}) = r^{\mathbb{1}}(C^{(\vec{\mu}_{-i},\mathbb{1})}) - 1.$$

[3] Theorem 4.6 shows that this choice has no influence on the work, and labeling with $\mathbb{1}$ instead leads to the same results.

We have by the partitioning that $i \in \mathcal{S}_{\mathbb{0}}(G^{(\vec{\mu}_{-i},\mathbb{0})})$ and $i \in \mathcal{S}_{\mathbb{1}}(G^{(\vec{\mu}_{-i},\mathbb{1})})$. Consequently, from Theorem 4.6 it follows that $\sigma_i(\vec{\mu}_{-i},\mathbb{0}) = \sigma_i(\vec{\mu}_{-i},\mathbb{1})$, and the proposition follows also in this case.

3. Otherwise, i has incoming and outgoing links in G. For each D-chain from i, denoted by C, we have two possibilities:
   (a) If the link in C incident to i is directed away from i, then $C^{(\vec{\mu}_{-i},\mathbb{0})} = C^{(\vec{\mu}_{-i},\mathbb{1})}$, and consequently $r^{\mathbb{1}}(C^{(\vec{\mu}_{-i},\mathbb{0})}) = r^{\mathbb{1}}(C^{(\vec{\mu}_{-i},\mathbb{1})})$, $s^{\mathbb{0}}(C^{(\vec{\mu}_{-i},\mathbb{0})}) = s^{\mathbb{0}}(C^{(\vec{\mu}_{-i},\mathbb{1})})$, and $Res(C^{(\vec{\mu}_{-i},\mathbb{0})}) = Res(C^{(\vec{\mu}_{-i},\mathbb{1})})$.
   (b) Otherwise, the link in C incident to i is directed toward i. We have $r^{\mathbb{1}}(C^{(\vec{\mu}_{-i},\mathbb{0})}) = r^{\mathbb{1}}(C^{(\vec{\mu}_{-i},\mathbb{1})}) - 1$, $s^{\mathbb{0}}(C^{(\vec{\mu}_{-i},\mathbb{0})}) = s^{\mathbb{0}}(C^{(\vec{\mu}_{-i},\mathbb{1})})$, and further $Res(C^{(\vec{\mu}_{-i},\mathbb{0})}) = Res(C^{(\vec{\mu}_{-i},\mathbb{1})}) + 1$. In this case we obtain
   $$2r^{\mathbb{1}}(C^{(\vec{\mu}_{-i},\mathbb{0})}) + 2s^{\mathbb{0}}(C^{(\vec{\mu}_{-i},\mathbb{0})}) + 1 \geq r^{\mathbb{1}}(C^{(\vec{\mu}_{-i},\mathbb{0})}) + 1 + s^{\mathbb{0}}(C^{(\vec{\mu}_{-i},\mathbb{0})}).$$

   In both situations (a) and (b) the following inequality holds:
   (5.1)
   $$2r^{\mathbb{1}}(C^{(\vec{\mu}_{-i},\mathbb{0})}) + 2s^{\mathbb{0}}(C^{(\vec{\mu}_{-i},\mathbb{0})}) + Res(C^{(\vec{\mu}_{-i},\mathbb{0})}) \geq r^{\mathbb{1}}(C^{(\vec{\mu}_{-i},\mathbb{1})}) + s^{\mathbb{0}}(C^{(\vec{\mu}_{-i},\mathbb{1})}).$$

   As i has incoming and outgoing links in G, it follows that $i \in \mathcal{S}_{\mathbb{1}}(G^{(\vec{\mu}_{-i},\mathbb{1})})$ and $i \in \mathcal{D}(G^{(\vec{\mu}_{-i},\mathbb{0})})$. From this, it follows by Theorem 4.6 that

   $$\sigma_i(\vec{\mu}_{-i},\mathbb{1}) = \min_{C^{(\vec{\mu}_{-i},\mathbb{1})} \in \mathcal{C}(i,G^{(\vec{\mu}_{-i},\mathbb{1})})} \left(r^{\mathbb{1}}(C^{(\vec{\mu}_{-i},\mathbb{1})}) + s^{\mathbb{0}}(C^{(\vec{\mu}_{-i},\mathbb{1})}) \right)$$

   and

   $$\sigma_i(\vec{\mu}_{-i},\mathbb{0}) = \min_{C^{(\vec{\mu}_{-i},\mathbb{0})} \in \mathcal{C}(i,G^{(\vec{\mu}_{-i},\mathbb{0})})} \left(2r^{\mathbb{1}}(C^{(\vec{\mu}_{-i},\mathbb{0})}) + 2s^{\mathbb{0}}(C^{(\vec{\mu}_{-i},\mathbb{0})}) + Res(C^{(\vec{\mu}_{-i},\mathbb{0})}) \right).$$

   Combining this with (5.1) proves the proposition in this case.

In all the cases the proposition holds. □

We show next that the damage caused by the choice of $\vec{\mu}_i = \mathbb{0}$ on the work of i is limited by the factor 2.

PROPOSITION 5.3. *For every node $i \in V$, and any vector $\vec{\mu} \in M$,*

$$\sigma_i(\vec{\mu}_{-i},\mathbb{0}) \leq 2 \cdot \sigma_i(\vec{\mu}_{-i},\mathbb{1}).$$

Proof. For sinks and sources the proposition follows from Proposition 5.2. We now consider a node i that is neither a sink nor a source. For each D-chain from i, denoted C, we have two cases:

(a) If the link in C incident to i is directed away from i, we have $Res(C^{(\vec{\mu}_{-i},\mathbb{0})}) = 0$, $r^{\mathbb{1}}(C^{(\vec{\mu}_{-i},\mathbb{1})}) = r^{\mathbb{1}}(C^{(\vec{\mu}_{-i},\mathbb{0})})$, and $s^{\mathbb{0}}(C^{(\vec{\mu}_{-i},\mathbb{0})}) = s^{\mathbb{0}}(C^{(\vec{\mu}_{-i},\mathbb{1})})$.
(b) Otherwise, the link in C incident to i is directed toward i. In this case, we have $Res(C^{(\vec{\mu}_{-i},\mathbb{0})}) = 1$, $r^{\mathbb{1}}(C^{(\vec{\mu}_{-i},\mathbb{1})}) = r^{\mathbb{1}}(C^{(\vec{\mu}_{-i},\mathbb{0})}) + 1$, and $s^{\mathbb{0}}(C^{(\vec{\mu}_{-i},\mathbb{0})}) = s^{\mathbb{0}}(C^{(\vec{\mu}_{-i},\mathbb{1})})$.

Trivially, we obtain

$$2r^{\mathbb{1}}(C^{(\vec{\mu}_{-i},\mathbb{0})}) + 2s^{\mathbb{0}}(C^{(\vec{\mu}_{-i},\mathbb{0})}) + 1 \leq 2 \cdot \left(r^{\mathbb{1}}(C^{(\vec{\mu}_{-i},\mathbb{0})}) + 1 + s^{\mathbb{0}}(C^{(\vec{\mu}_{-i},\mathbb{0})}) \right),$$

from which it follows for both cases (a) and (b) that

$$2r^{\mathbb{1}}(C^{(\vec{\mu}_{-i},\mathbb{0})}) + 2s^{\mathbb{0}}(C^{(\vec{\mu}_{-i},\mathbb{0})}) + Res(C^{(\vec{\mu}_{-i},\mathbb{0})}) \leq 2 \cdot \left(r^{\mathbb{1}}(C^{(\vec{\mu}_{-i},\mathbb{1})}) + s^{\mathbb{0}}(C^{(\vec{\mu}_{-i},\mathbb{1})}) \right). \quad (5.2)$$

As node i has both incoming and outgoing links in G, it follows that $i \in \mathcal{S}_{\mathbb{1}}(G^{(\vec{\mu}_{-i},\mathbb{1})})$ and $i \in \mathcal{D}(G^{(\vec{\mu}_{-i},\mathbb{0})})$. From this, it follows by Theorem 4.6 that

$$\sigma_i(\vec{\mu}_{-i},\mathbb{1}) = \min_{C^{(\vec{\mu}_{-i},\mathbb{1})} \in \mathcal{C}(i,G^{(\vec{\mu}_{-i},\mathbb{1})})} \left(r^{\mathbb{1}}(C^{(\vec{\mu}_{-i},\mathbb{1})}) + s^{\mathbb{0}}(C^{(\vec{\mu}_{-i},\mathbb{1})}) \right)$$

and

$$\sigma_i(\vec{\mu}_{-i},\mathbb{0}) = \min_{C^{(\vec{\mu}_{-i},\mathbb{0})} \in \mathcal{C}(i,G^{(\vec{\mu}_{-i},\mathbb{0})})} \left(2r^{\mathbb{1}}(C^{(\vec{\mu}_{-i},\mathbb{0})}) + 2s^{\mathbb{0}}(C^{(\vec{\mu}_{-i},\mathbb{0})}) + Res(C^{(\vec{\mu}_{-i},\mathbb{0})}) \right).$$

Combining this with (5.2) finally proves the proposition. ∎

On the contrary, the next proposition states that if a node j other than i changes its incoming labels from $\mathbb{0}$ to $\mathbb{1}$, the work performed by i either stays the same or increases.

PROPOSITION 5.4. *For any two different nodes i and j, and any vector $\vec{\mu} \in M$,*

$$\sigma_i(\vec{\mu}_{-j},\mathbb{0}) \leq \sigma_i(\vec{\mu}_{-j},\mathbb{1}).$$

Proof. As μ_j has influence only on the labels of incoming links of j, and because $j \neq i$, for each D-chain C from i, $Res(C^{(\vec{\mu}_{-j},\mathbb{0})}) = Res(C^{(\vec{\mu}_{-j},\mathbb{1})})$. If j is not part of a D-chain from i, the labels of its incoming links have no influence on i's work, and we are done. So we suppose that j is a node on a D-chain C from i, and we distinguish four cases:

- If j has two outgoing links relative to C, then we have $C^{(\vec{\mu}_{-j},\mathbb{0})} = C^{(\vec{\mu}_{-j},\mathbb{1})}$. We obtain $s^{\mathbb{0}}(C^{(\vec{\mu}_{-j},\mathbb{0})}) + r^{\mathbb{1}}(C^{(\vec{\mu}_{-j},\mathbb{0})}) = s^{\mathbb{0}}(C^{(\vec{\mu}_{-j},\mathbb{1})}) + r^{\mathbb{1}}(C^{(\vec{\mu}_{-j},\mathbb{1})})$.
- j has two incoming links relative to C. If $\mu_j = \mathbb{0}$, then $s^{\mathbb{0}}$ is augmented; otherwise, the link directed toward i augments $r^{\mathbb{1}}$. We obtain $s^{\mathbb{0}}(C^{(\vec{\mu}_{-j},\mathbb{0})}) = s^{\mathbb{0}}(C^{(\vec{\mu}_{-j},\mathbb{1})}) + 1$ and $r^{\mathbb{1}}(C^{(\vec{\mu}_{-j},\mathbb{0})}) = r^{\mathbb{1}}(C^{(\vec{\mu}_{-j},\mathbb{1})}) - 1$ and further obtain $s^{\mathbb{0}}(C^{(\vec{\mu}_{-j},\mathbb{0})}) + r^{\mathbb{1}}(C^{(\vec{\mu}_{-j},\mathbb{0})}) = s^{\mathbb{0}}(C^{(\vec{\mu}_{-j},\mathbb{1})}) + r^{\mathbb{1}}(C^{(\vec{\mu}_{-j},\mathbb{1})})$.
- j has one incoming link relative to C, and it is directed toward D. The mark of this link has influence only on $\ell^{\mathbb{1}}$. Again, we obtain $s^{\mathbb{0}}(C^{(\vec{\mu}_{-j},\mathbb{0})}) + r^{\mathbb{1}}(C^{(\vec{\mu}_{-j},\mathbb{0})}) = s^{\mathbb{0}}(C^{(\vec{\mu}_{-j},\mathbb{1})}) + r^{\mathbb{1}}(C^{(\vec{\mu}_{-j},\mathbb{1})})$.
- j has one incoming link relative to C, and it is directed toward i. The mark of this link has influence only on $r^{\mathbb{1}}$. We obtain $r^{\mathbb{1}}(C^{(\vec{\mu}_{-j},\mathbb{0})}) + 1 = r^{\mathbb{1}}(C^{(\vec{\mu}_{-j},\mathbb{1})})$.

Thus, we obtain $s^{\mathbb{0}}(C^{(\vec{\mu}_{-j},\mathbb{0})}) + r^{\mathbb{1}}(C^{(\vec{\mu}_{-j},\mathbb{0})}) \leq s^{\mathbb{0}}(C^{(\vec{\mu}_{-j},\mathbb{1})}) + r^{\mathbb{1}}(C^{(\vec{\mu}_{-j},\mathbb{1})})$ for all chains C, and the proposition follows. ∎

Repeated application of Proposition 5.4 leads to the following corollary.

COROLLARY 5.5. *For every node $i \in V$, and any two vectors $\vec{\mu}$ and $\vec{\nu}$ in M such that for all $j \in V$, $\mu_j \leq \nu_j$ and $\mu_i = \nu_i$, it holds that $\sigma_i(\vec{\mu}) \leq \sigma_i(\vec{\nu})$.*

Finally, we show that when using the LU policy, whether a node has to perform reversals depends *solely* on the initial directed graph G (and not on $G^{\vec{\mu}}$). In particular, if there is a labeling in M for which node i performs no reversal, then for every labeling in M, i makes no reversal.

PROPOSITION 5.6. *For each node $i \in V$, the following statements are equivalent:*

(1) *for all $\vec{\mu} \in M : \sigma_i(\vec{\mu}) = 0$.*
(2) *there is a $\vec{\mu} \in M : \sigma_i(\vec{\mu}) = 0$.*
(3) *i is a good node (i.e., there is a path from i to D).*

Proof. Trivially, (1) implies (2).

If (2) holds for some $\vec{\mu} \in M$, then Theorem 4.6 shows that there is a D-chain from i, denoted by C, such that $C^{\vec{\mu}}$ satisfies $r^{\mathbb{1}}(C^{\vec{\mu}}) + s^{\mathbb{0}}(C^{\vec{\mu}}) + Res(C^{\vec{\mu}}) = \mathbb{0}$. Therefore $r^{\mathbb{1}}(C^{\vec{\mu}}) = s^{\mathbb{0}}(C^{\vec{\mu}}) = Res(C^{\vec{\mu}}) = 0$. We now argue that C is a path from i to D.

Suppose, by way of contradiction, that there is a link e in C directed toward i. As $r^{\mathbb{1}}(C^{\vec{\mu}}) = \mathbb{0}$, all links, including e, that are directed toward i must be unmarked. Moreover, from $Res(C^{\vec{\mu}}) = 0$ we deduce that the first link in $C^{\vec{\mu}}$ is directed toward D. Hence the latter link and e are directed in the opposite direction, and so there exists a node j with two distinct incoming links relative to $C^{\vec{\mu}}$. Since $r^{\mathbb{1}}(C^{\vec{\mu}}) = 0$, $\vec{\mu}_j = \mathbb{0}$. Therefore, node j has two unmarked incoming links relative to $C^{\vec{\mu}}$, and so $s^{\mathbb{0}}(C^{\vec{\mu}}) > 0$. This provides the required contradiction and shows that (2) implies (3).

If (3) holds, then there exists a simple chain C which is a path from i toward D. From the definitions of $r^{\mathbb{1}}$, $s^{\mathbb{0}}$, and Res, we immediately obtain that for any vector $\vec{\mu} \in M$, $r^{\mathbb{1}}(C^{\vec{\mu}}) = s^{\mathbb{0}}(C^{\vec{\mu}}) = Res(C^{\vec{\mu}}) = 0$. This shows that (3) implies (1). ∎

6. The *LU* policy as a game. By Propositions 5.2, the work by node i may be decreased by labeling its incoming links with $\mathbb{1}$, while Proposition 5.4 shows that this might increase the work of other nodes. This kind of conflicting interests is exactly what is handled by game theory. We are thus naturally led to model the LU policy as a game: to each node i we associate a player who chooses a value in $\{\mathbb{0}, \mathbb{1}\}$ for μ_i, and the cost incurred by player i is given by the work done by node i. In doing so, game-theoretic analysis allows us to manage the contradictory interests of the players/nodes and to precisely compare the different initial LU labelings. In particular, we provide a rigorous comparison of Full Reversal and Partial Reversal, which correspond to the special vectors $\vec{\mathbb{1}}$ and $\vec{\mathbb{0}}$, respectively.

6.1. Game-theoretic definitions. The LU game consists of the set V of n *players*. Each player i selects a *pure strategy* $\mu_i \in \{\mathbb{0}, \mathbb{1}\}$, and we form the vector $\vec{\mu} = (\mu_1, \mu_2, \ldots, \mu_n)$, called a *pure profile*. For a given routable graph, the profile $\vec{\mu}$ entirely determines the work for each player/node. We naturally consider for each player i the *cost* $\sigma_i(\vec{\mu})$ incurred by node i, which is the work that i has to perform in LR starting from $G^{\vec{\mu}}$.

We first recall some basic notions from game theory [20, 19]. A pure Nash equilibrium is a pure profile $\vec{\mu}$ of the game where no player can prefer a different strategy if the current strategies of the other players are fixed:

$$\forall i \in V, \quad \forall s \in \{\mathbb{0}, \mathbb{1}\} : \ \sigma_i(\vec{\mu}_{-i}, s) \geq \sigma_i(\vec{\mu}).$$

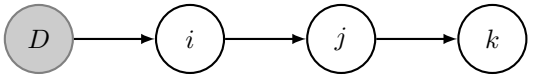
A pure Nash equilibrium thus represents a profile where, from a local and selfish viewpoint, each player i has no motivation to change its strategy. A pure Nash equilibrium is based on local conditions and so is not necessarily the best possible profile, in the sense that all players incur minimum cost. Such a best pure profile $\vec{\mu}$ is called a *global optimum*:

$$\forall i \in V, \quad \forall \vec{\nu} \in M : \sigma_i(\vec{\mu}) \leq \sigma_i(\vec{\nu}).$$

Further, we investigate whether the LU game is a *potential game* [16]: a function $P : M \to \mathbb{R}$ is a potential function for a game if

$$\forall i \in V, \quad \forall \vec{\mu} \in M, \quad \forall s \in \{\mathbb{0}, \mathbb{1}\} : \ \sigma_i(\vec{\mu}) - \sigma_i(\vec{\mu}_{-i}, s) > 0 \leftrightarrow P(\vec{\mu}) - P(\vec{\mu}_{-i}, s) > 0.$$

A game is then called a potential game if it admits a potential function. It was shown by Monderer and Shapley [16] that every potential game has a pure Nash equilibrium. In general, however, the existence of a pure Nash equilibrium is not guaranteed. Interestingly, Proposition 5.2 exactly expresses the fact that Full Reversal is a pure Nash equilibrium. However, we show that the LU game is not a potential game.



		j = $\mathbb{0}$	j = $\mathbb{1}$
i	$\mathbb{0}$	$(1,1)$ U	$(1,1)$ V
	$\mathbb{1}$	$(1,3)$ W	$(1,2)$ X

FIG. 6.1. *An example graph, with a table of the subgame of players i and j.*

To evaluate the overall quality of a pure profile $\vec{\mu}$, we consider the *social cost*, which is the sum of the costs incurred by all players:

$$SC(\vec{\mu}) = \sum_{i \in V} \sigma_i(\vec{\mu}).$$

Clearly, $SC(\vec{\mu})$ is the global work complexity of the LR algorithm starting from $G^{\vec{\mu}}$. While a global optimum does not necessarily exist, there is always a pure profile with minimum social cost among all profiles.

Further, we are interested in measuring the increase in the social cost when the profiles result from selfish choices — that is, pure Nash equilibria — instead of the choices of the profiles with minimum social cost. Koutsoupias and Papadimitriou [13] introduced the *price of anarchy*, which captures this notion. In this paper we consider the pure price of anarchy, which is defined as the ratio between the worst social cost of a pure Nash equilibrium and the minimum social cost among all profiles. In the following we show that, with respect to the social cost, Full Reversal corresponds to the worst pure Nash equilibrium. For our game, finding the price of anarchy thus coincides with studying how much worse Full Reversal is than a profile with minimum social cost.

6.2. Properties of strategy profiles. We now show that the LU game is not a potential game, using the example given in Figure 6.1: the pairs in the table are the cost pairs $(\sigma_i(\vec{\mu}), \sigma_j(\vec{\mu}))$ for the profiles $\vec{\mu} = (\mu_i, \mu_j)$. As μ_k has no influence on the cost incurred by i and j, we may omit it in the following discussion. The uppercase letters represent values of a function $f : M \to \mathbb{R}$. We just have to show that no such function f can be a potential function. As $\sigma_i(\mathbb{1}, \mathbb{0}) = \sigma_i(\mathbb{0}, \mathbb{0})$, we require from the definition of a potential function that $U = W$. Similarly, $\sigma_i(\mathbb{1}, \mathbb{1}) = \sigma_i(\mathbb{0}, \mathbb{1})$, $\sigma_j(\mathbb{0}, \mathbb{0}) = \sigma_j(\mathbb{0}, \mathbb{1})$, which implies $V = X$, $U = V$, and consequently $W = X$. However, as $\sigma_j(\mathbb{1}, \mathbb{0}) > \sigma_j(\mathbb{1}, \mathbb{1})$ we require $W > X$, a contradiction. We thus conclude that for certain graphs, there is no potential function, and thus LU is not a potential game.

We continue our investigation of the LU game by the following theorem, which provides the exact characterization of pure Nash equilibria.

THEOREM 6.1. *The profile $\vec{\mu}$ is a pure Nash equilibrium if and only if for all nodes i it holds that i is a sink, or i is a source, or $\mu_i = \mathbb{1}$, or $\sigma_i(\vec{\mu}) \leq 1$.*

Proof. First we show that if i is a sink in G, or i is a source in G, or $\mu_i = \mathbb{1}$, or $\sigma_i(\vec{\mu}) \leq 1$, then for all s in $\{\mathbb{0}, \mathbb{1}\}$, $\sigma_i(\vec{\mu}_{-i}, s) \geq \sigma_i(\vec{\mu})$. If $\mu_i = \mathbb{1}$, then the claim is a straightforward consequence of the first part of Proposition 5.2. If i is a source or a sink in G, then the claim directly follows from the second part of Proposition 5.2. Finally, suppose $\sigma_i(\vec{\mu}) \leq 1$, that, is $\sigma_i(\vec{\mu}) = 0$ or $\sigma_i(\vec{\mu}) = 1$. If $\sigma_i(\vec{\mu}) = 0$, then clearly, i's work is already minimum. If $\sigma_i(\vec{\mu}) = 1$, then Proposition 5.6 implies that i is a

bad node, and that for any initial link labeling, i must take at least one step. We conclude that if i is a sink or a source, $\mu_i = \mathbb{1}$, or $\sigma_i(\vec{\mu}) \leq 1$, then for any s in $\{\mathbb{0}, \mathbb{1}\}$, we obtain $\sigma_i(\vec{\mu}_{-i}, s) \geq \sigma_i(\vec{\mu})$.

Consequently, if for each node i one of these requirements holds, then for each i we have that for all s in $\{\mathbb{0}, \mathbb{1}\}$, $\sigma_i(\vec{\mu}_{-i}, s) \geq \sigma_i(\vec{\mu})$; i.e., $\vec{\mu}$ is a pure Nash equilibrium.

Conversely, assume that $\vec{\mu}$ is a pure Nash equilibrium. Further, assume by way of contradiction that there is some node i which is neither a sink nor a source in G, $\mu_i = \mathbb{0}$, and $\sigma_i(\vec{\mu}) > 1$. We will show that changing node i's strategy to $\mathbb{1}$ causes i to take fewer than $\sigma_i(\vec{\mu})$ steps, implying that $\vec{\mu}$ is not a pure Nash equilibrium. As $i \in \mathcal{D}(G^{\vec{\mu}})$ and $\sigma_i(\vec{\mu}) > 1$, Theorem 4.6 implies that for every D-chain C from i,

$$2r^{\mathbb{1}}(C^{\vec{\mu}}) + 2s^{\mathbb{0}}(C^{\vec{\mu}}) + Res(C^{\vec{\mu}}) > 1.$$

Consequently, $r^{\mathbb{1}}(C^{\vec{\mu}}) > 0$ or $s^{\mathbb{0}}(C^{\vec{\mu}}) > 0$. We now show that

$$2r^{\mathbb{1}}(C^{\vec{\mu}}) + 2s^{\mathbb{0}}(C^{\vec{\mu}}) + Res(C^{\vec{\mu}}) > r^{\mathbb{1}}(C^{(\vec{\mu}_{-i},\mathbb{1})}) + s^{\mathbb{0}}(C^{(\vec{\mu}_{-i},\mathbb{1})}). \tag{6.1}$$

There are two cases to consider:

1. The first link in C is directed away from i. We have

   $$r^{\mathbb{1}}(C^{\vec{\mu}}) = r^{\mathbb{1}}(C^{(\vec{\mu}_{-i},\mathbb{1})}), \quad s^{\mathbb{0}}(C^{\vec{\mu}}) = s^{\mathbb{0}}(C^{(\vec{\mu}_{-i},\mathbb{1})}), \quad \text{and } Res(C^{\vec{\mu}}) = 0.$$

   From the fact that $r^{\mathbb{1}}(C^{\vec{\mu}}) > 0$ or $s^{\mathbb{0}}(C^{\vec{\mu}}) > 0$, we derive that (6.1) holds.
2. The first link in C is directed toward i. Then

   $$r^{\mathbb{1}}(C^{\vec{\mu}}) + 1 = r^{\mathbb{1}}(C^{(\vec{\mu}_{-i},\mathbb{1})}), \quad s^{\mathbb{0}}(C^{\vec{\mu}}) = s^{\mathbb{0}}(C^{(\vec{\mu}_{-i},\mathbb{1})}), \quad \text{and } Res(C^{\vec{\mu}}) = 1.$$

   Since $r^{\mathbb{1}}(C^{\vec{\mu}}) > 0$ or $s^{\mathbb{0}}(C^{\vec{\mu}}) > 0$, we obtain

   $$2r^{\mathbb{1}}(C^{\vec{\mu}}) + 2s^{\mathbb{0}}(C^{\vec{\mu}}) + 1 > r^{\mathbb{1}}(C^{\vec{\mu}}) + 1 + s^{\mathbb{0}}(C^{\vec{\mu}}),$$

   which implies that (6.1) holds also in this case.

We observe that $i \in \mathcal{S}_{\mathbb{1}}(G^{(\vec{\mu}_{-i},\mathbb{1})})$ and $Res(C^{(\vec{\mu}_{-i},\mathbb{1})}) = 0$. From Theorem 4.6 and as $i \in \mathcal{D}(G^{\vec{\mu}})$, it follows that

$$\sigma_i(\vec{\mu}) = \min_{C^{\vec{\mu}} \in \mathcal{C}(i, G^{\vec{\mu}})} \left(2r^{\mathbb{1}}(C^{\vec{\mu}}) + 2s^{\mathbb{0}}(C^{\vec{\mu}}) + Res(C^{\vec{\mu}})\right).$$

As $i \in \mathcal{S}_{\mathbb{1}}(G^{(\vec{\mu}_{-i},\mathbb{1})})$, we obtain

$$\sigma_i(\vec{\mu}_{-i}, \mathbb{1}) = \min_{C^{(\vec{\mu}_{-i},\mathbb{1})} \in \mathcal{C}(i, G^{(\vec{\mu}_{-i},\mathbb{1})})} \left(r^{\mathbb{1}}(C^{(\vec{\mu}_{-i},\mathbb{1})}) + s^{\mathbb{0}}(C^{(\vec{\mu}_{-i},\mathbb{1})})\right).$$

Combined with inequality (6.1), this implies that $\sigma_i(\vec{\mu}) > \sigma_i(\vec{\mu}_{-i}, \mathbb{1})$, and $\vec{\mu}$ is not a pure Nash equilibrium, which provides the required contradiction. ∎

The conditions in Theorem 6.1 are rather different in nature: whether some node is a sink or a source solely depends on the graph and not on the strategy, contrary to the condition $\mu_i = \mathbb{1}$. The final condition $\sigma_i(\vec{\mu}) \leq 1$ is hybrid: Proposition 5.6 shows that good nodes never take steps independently of the profile and thus always satisfy $\sigma_i(\vec{\mu}) = 0$. Further, Theorem 4.6 shows that under LU, a bad node which is a neighbor of a good node always incurs a cost equal to 1. Such nodes thus satisfy this condition for any profile $\vec{\mu}$. Other nodes may satisfy the latter condition, but

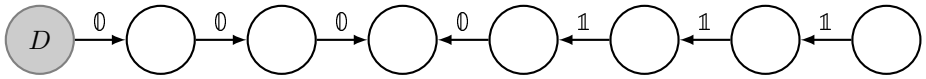


FIG. 6.2. *A global optimum other than* $\vec{\mathbb{0}}$ *and* $\vec{\mathbb{1}}$.

this depends on both the directed graph and the strategies of the other players, as exemplified in Figure 6.2, where each node makes exactly one reversal.

Theorem 6.1 implies that for the graph given in Figure 6.2, the indicated labeling is a pure Nash equilibrium. In fact, it is a global optimum as for each node i, $\sigma_i(\vec{\mu}) = 1$. Figure 6.2 thus provides an example of a graph for which neither Full Reversal nor Partial Reversal is the best strategy. More generally, determining the best strategies requires the knowledge of the whole graph, which is not available to the nodes in a distributed setting. So from a practical viewpoint, the flexibility offered by LU in the initializations of link labels is not interesting a priori. However, the LU labelings form a continuum between Full Reversal and Partial Reversal that helps significantly in the analysis and comparison of Full Reversal and Partial Reversal, as shown below.

Contrary to the profile $\vec{\mathbb{1}}$, the profile $\vec{\mathbb{0}}$ is not necessarily a pure Nash equilibrium: for instance, consider the directed graph (D-chain) in Figure 6.2 and apply Theorem 6.1. However, we show that from a global viewpoint, i.e., with respect to the social cost, $\vec{\mathbb{0}}$ indeed has advantageous properties compared to $\vec{\mathbb{1}}$. We start by proving that $\vec{\mathbb{1}}$ is the worst pure Nash equilibrium in terms of social cost.

THEOREM 6.2. *If the profile* $\vec{\mu}$ *is a pure Nash equilibrium, then for all nodes* i,

$$\sigma_i(\vec{\mu}) \leq \sigma_i(\vec{\mathbb{1}}).$$

Proof. By Theorem 6.1, for every node i it holds that (1) i a sink or a source, or (2) $\mu_i = \mathbb{1}$, or (3) $\sigma_i(\vec{\mu}) \leq 1$. We consider the following three cases:

1. Node i is a sink or a source in graph G. From Proposition 5.2 it follows that $\sigma_i(\vec{\mathbb{1}}_{-i}, \mu_i) = \sigma_i(\vec{\mathbb{1}})$. By Corollary 5.5, $\sigma_i(\vec{\mu}) \leq \sigma_i(\vec{\mathbb{1}}_{-i}, \mu_i)$ and, consequently, $\sigma_i(\vec{\mu}) \leq \sigma_i(\vec{\mathbb{1}})$.
2. $\mu_i = \mathbb{1}$. In this case, Corollary 5.5 directly implies that $\sigma_i(\vec{\mu}) \leq \sigma_i(\vec{\mathbb{1}})$.
3. $\sigma_i(\vec{\mu}) \leq 1$. We consider the two cases $\sigma_i(\vec{\mu}) = 0$ and $\sigma_i(\vec{\mu}) = 1$. If $\sigma_i(\vec{\mu}) = 0$, then the theorem trivially holds. If $\sigma_i(\vec{\mu}) = 1$, then as a direct consequence of Proposition 5.6, we obtain $\sigma_i(\vec{\mu}) \leq \sigma_i(\vec{\mathbb{1}})$.

In all three cases, the theorem holds. □

COROLLARY 6.3. *If the profile* $\vec{\mu}$ *is a pure Nash equilibrium, then* $SC(\vec{\mu}) \leq SC(\vec{\mathbb{1}})$.

We now bound the ratio between the social cost of $\vec{\mathbb{1}}$ and the social cost of any profile. Combining the two latter results, we derive an upper bound on the pure price of anarchy, which we prove to be tight.

First, we introduce additional notation. Let n_B be the number of bad nodes, and let R denote the maximum of the works of nodes in Full Reversal, i.e., for $\vec{\mathbb{1}}$. Let us recall that Corollary 5.1 gives R as the following function of the graph:

$$R = \max_{i \in V} \left(\min_{C \in \mathcal{C}(i,G)} (r(C)) \right).$$

THEOREM 6.4. *For all* $\vec{\mu} \in M$ *it holds that*

$$\frac{SC(\vec{\mathbb{1}})}{SC(\vec{\mu})} \leq R \cdot \left(1 - \frac{R-1}{2n_B}\right).$$

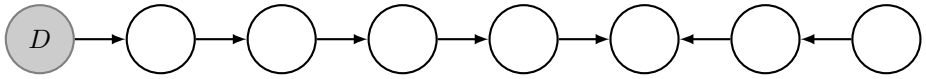


FIG. 6.3. *Chain that maximizes the social cost of Full Reversal for $R = 5$ and $n_B = 7$.*

Proof. Let n_ℓ denote the number of nodes i with $\sigma_i(\vec{\mathbb{1}}) = \ell$ for $0 \leq \ell \leq R$. The number of bad nodes n_B is then given by

$$n_B = \sum_{\ell=1}^{R} n_\ell.$$

Moreover, $V = \mathcal{S}_{\mathbb{1}}(G^{\vec{\mathbb{1}}})$, and we obtain

$$SC(\vec{\mathbb{1}}) = \sum_{\ell=1}^{R} \ell \cdot n_\ell. \tag{6.2}$$

For given R and n_B, one maximizes $SC(\vec{\mathbb{1}})$ by constructing a graph in which as many nodes as possible make R steps.

Since the number of steps by a node i in Full Reversal is the maximum, over all D-chains from i, of the number of links directed away from D, it follows that a node with work x has only neighbors with work x, $x+1$, or $x-1$. Thus a node with work R can only be reached from D by a chain of nodes, along which the work increases by at most one at each node. Thus, $SC(\vec{\mathbb{1}})$ is maximized by a chain in which the R links closest to D are directed away from D, and the remaining $n_B - R$ links are directed toward D (cf. the chain given in Figure 6.3). In such a chain, the $n_B - (R-1)$ nodes farthest from D all have work R, while the other nodes have work $1, 2, \ldots, R-1$, respectively. Consequently, the social cost is bounded by

$$SC(\vec{\mathbb{1}}) \leq R \cdot (n_B - (R-1)) + \sum_{\ell=1}^{R-1} \ell = R \cdot (n_B - R + 1) + \frac{R \cdot (R-1)}{2}.$$

Besides, from Proposition 5.6, it follows that for any pure profile $\vec{\mu}$, each bad node must take at least one step, and consequently that

$$SC(\vec{\mu}) \geq n_B.$$

We obtain that

$$\frac{SC(\vec{\mathbb{1}})}{SC(\vec{\mu})} \leq R - \frac{R \cdot (R-1)}{2n_B}$$

and the theorem follows. □

COROLLARY 6.5. *The pure price of anarchy is less than or equal to $R \cdot \left(1 - \frac{R-1}{2n_B}\right)$.*

6.3. Partial Reversal: A socially conscious choice. Now, we focus on the pure profile $\vec{\mathbb{0}}$ (Partial Reversal) in the LU game. First, we establish that when $\vec{\mathbb{0}}$ is a pure Nash equilibrium, it is a global optimum. As a result, we show that the bound on the pure price of anarchy in Corollary 6.5 is actually tight. Then, we establish that the social cost of $\vec{\mathbb{0}}$ is at most twice the optimal social cost.

THEOREM 6.6. *If $\vec{\mathbb{0}}$ is a pure Nash equilibrium, then $\vec{\mathbb{0}}$ is a global optimum.*

Proof. We show that if $\vec{\mathbb{0}}$ is a pure Nash equilibrium, then for any node i and any profile $\vec{\mu}$, $\sigma_i(\vec{\mathbb{0}}) \leq \sigma_i(\vec{\mu})$. There are two cases to consider:

1. Node i has both incoming and outgoing links. Since $\vec{\mathbb{0}}$ is a pure Nash equilibrium, it follows from Theorem 6.1 that $\sigma_i(\vec{\mathbb{0}}) \leq 1$. By Proposition 5.6, it holds that $\sigma_i(\vec{\mathbb{0}}) \leq \sigma_i(\vec{\mu})$.
2. Node i is a sink or a source. By Proposition 5.2, $\sigma_i(\vec{\mathbb{0}}_{-i}, \mu_i) = \sigma_i(\vec{\mathbb{0}})$. From Corollary 5.5, it follows that $\sigma_i(\vec{\mathbb{0}}_{-i}, \mu_i) \leq \sigma_i(\vec{\mu})$, and thus $\sigma_i(\vec{\mathbb{0}}) \leq \sigma_i(\vec{\mu})$.

In both cases, the theorem holds. ∎

Consider a graph which is a chain with D as an extremity, and all links are oriented away from D. From Theorem 4.6, we obtain $SC(\vec{\mathbb{1}}) = \frac{n\cdot(n+1)}{2}$, while $SC(\vec{\mathbb{0}}) = n$. The profile $\vec{\mathbb{0}}$ is a pure Nash equilibrium since for each node i, $\sigma_i(\vec{\mathbb{0}}) = 1$, and so $\vec{\mathbb{0}}$ achieves the minimum social cost. Thereby the pure price of anarchy is equal to $\frac{n+1}{2}$. Since $n = n_B = R$, the bound in Corollary 6.5 is tight.

Hence the social cost of $\vec{\mathbb{1}}$ (Full Reversal) may be worse than $\vec{\mathbb{0}}$ (Partial Reversal) by some factor $\Theta(n)$. In Figure 5.1, we gave an example of a graph for which the social cost of Partial Reversal is less than twice the social cost of Full Reversal. We are now in position to prove that in fact, this maximal ratio of 2 holds for any graph.

THEOREM 6.7. *For any profile $\vec{\mu} \in M$, it holds that $SC(\vec{\mathbb{0}}) \leq 2 \cdot SC(\vec{\mu})$.*

Proof. Let i be any node in V. From Corollary 5.5, it follows that $\sigma_i(\vec{\mathbb{0}}_{-i}, \mu_i) \leq \sigma_i(\vec{\mu})$. Moreover, by Proposition 5.3, $\sigma_i(\vec{\mathbb{0}}) \leq 2 \cdot \sigma_i(\vec{\mathbb{0}}_{-i}, \mu_i)$. Consequently, $\sigma_i(\vec{\mathbb{0}}) \leq 2 \cdot \sigma_i(\vec{\mu})$, and the theorem follows. ∎

6.4. Properties of mixed strategy profiles. Until now, we have studied deterministic strategies in order to compare the work complexity of different initial link labelings. A natural question is whether random link labelings have interesting properties. To investigate this, we consider the mixed version of the LU game: by assigning a probability to each pure strategy of a player, one obtains a *mixed strategy* and considers the expected cost of this profile. As we still assume locally uniform labelings, (AC) is satisfied.

More precisely, each player i selects a mixed strategy $\mu_i^* \in [0,1]$ in the sense that strategy $\mathbb{1}$ is chosen with probability μ_i^*. The vector $\vec{\mu}^* = (\mu_1^*, \mu_2^*, \ldots, \mu_n^*)$ is called a *mixed profile*. We denote by M^* the set of all mixed strategy profiles. Given a mixed profile $\vec{\mu}^*$, a specific pure profile $\vec{\mu}$ occurs with probability $P(\vec{\mu}, \vec{\mu}^*)$; its value is given by

$$P(\vec{\mu}, \vec{\mu}^*) = \prod_{i:\mu_i=\mathbb{1}} \mu_i^* \cdot \prod_{i:\mu_i=\mathbb{0}} (1-\mu_i^*) = \prod_{i=1}^{n} \left(1 - \mu_i + (-1)^{1-\mu_i} \cdot \mu_i^*\right). \tag{6.3}$$

From this, we obtain the expected cost incurred by player i in the mixed profile $\vec{\mu}^*$ as

$$\bar{\sigma}_i(\vec{\mu}^*) = \sum_{\vec{\mu} \in M} P(\vec{\mu}, \vec{\mu}^*) \cdot \sigma_i(\vec{\mu}). \tag{6.4}$$

The mixed profile $\vec{\mu}^*$ is a *(mixed) Nash equilibrium* if

$$\forall i \in V, \quad \forall p \in [0,1] : \ \bar{\sigma}_i(\vec{\mu}^*_{-i}, p) \geq \bar{\sigma}_i(\vec{\mu}^*).$$

Nash [18] showed that every finite mixed game has an equilibrium. As shown above, the LU game always has a pure Nash equilibrium.

A natural question is whether there are mixed Nash equilibria that are different from pure ones. To provide some insight toward answering this question in general, we first consider a pure equilibrium $\vec{\mu}$ and some initial sink j. We will show that from $\vec{\mu}$ we can find additional mixed equilibria just by varying the mixed strategy of j. From

Proposition 5.2, we see that if the other strategies are fixed, the cost incurred by an initial sink is the same no matter which pure strategy the sink uses ($\mathbb{0}$ or $\mathbb{1}$). Further, if some initial sink j is on some D-chain from another node i, then changing j's strategy from $\mathbb{0}$ to $\mathbb{1}$ causes $s^{\mathbb{0}}$ to decrease by 1 and $r^{\mathbb{1}}$ to increase by 1. Thus, Theorem 4.6 implies that j's strategy has no influence on the costs incurred by any node in the pure game. As the expected cost in the mixed game is just the weighted sum of the costs in the pure game, we obtain for any mixed profile $\vec{\nu}^*$, any player i, and any mixed strategy $p \in [0,1]$ that $\bar{\sigma}_i(\vec{\nu}^*_{-j}, p) = \bar{\sigma}_i(\vec{\nu}^*)$ if j is an initial sink. Consequently, if j is an initial sink and the function $y \mapsto \bar{\sigma}_i(\vec{\nu}^*_{-i}, y)$ is minimized at some u, then for any $p \in [0,1]$, the function $y \mapsto \bar{\sigma}_i((\vec{\nu}^*_{-j}, p)_{-i}, y)$ is minimized at u. It follows that if $\vec{\mu}$ is a pure Nash equilibrium, $\vec{\mu}^*$ is the mixed equivalent[4] of $\vec{\mu}$, and j is an initial sink, then for any $p \in [0,1]$ the mixed profile $(\vec{\mu}^*_{-j}, p)$ is a mixed equilibrium. As there is always a pure equilibrium in the LU game (namely $\vec{\mathbb{1}}$), it follows that for any initial graph with at least one sink other than D, there are (infinitely many) mixed equilibria different from pure ones.

However, such additional equilibria are not particularly interesting, as they provide additional favorable choices only for players whose choices do not influence the cost incurred by any player. We now show that generally all mixed equilibria that are different from pure ones are not interesting and that therefore randomization does not help here.

As the cost incurred by player i in a mixed profile is the weighted sum of the costs incurred by player i in all the pure strategy profiles, we directly obtain the following from Proposition 5.2.

PROPOSITION 6.8. *For any node i and any mixed profile $\vec{\mu}^* \in M^*$,*

$$\bar{\sigma}_i(\vec{\mu}^*_{-i}, 0) \geq \bar{\sigma}_i(\vec{\mu}^*_{-i}, 1).$$

Using this corollary, we can prove the following theorem, which characterizes the nature of the mixed Nash equilibria in this game.

THEOREM 6.9. *If $\vec{\mu}^*$ is a mixed Nash equilibrium and there is some $i \in V$ with $\mu^*_i \in]0,1[$, then for any $p \in [0,1]$, $(\vec{\mu}^*_{-i}, p)$ is a mixed Nash equilibrium.*

Proof. Consider by way of contradiction that $\vec{\mu}^*$ is a mixed Nash equilibrium and i is a player with μ^*_i in $]0,1[$, but there is some $p \in [0,1]$ such that $\vec{\nu}^* = (\vec{\mu}^*_{-i}, p)$ is not a mixed Nash equilibrium. As $\vec{\nu}^*$ is not an equilibrium, there must be some index j and some $q \in [0,1]$ such that $\bar{\sigma}_j(\vec{\nu}^*_{-j}, q) < \bar{\sigma}_j(\vec{\nu}^*)$. The function f defined by $f(u) = \bar{\sigma}_i(\vec{\mu}^*_{-i}, u)$ is linear due to (6.3) and (6.4). From this, and since $\vec{\mu}^*$ is a mixed Nash equilibrium—that is, f has its global minimum at $\mu^*_i \in]0,1[$—it follows that the function f is constant. Therefore, from $\bar{\sigma}_j(\vec{\nu}^*_{-j}, q) < \bar{\sigma}_j(\vec{\nu}^*)$ it follows that $j \neq i$. We obtain

$$\exists j \in V, \quad \exists q \in [0,1] \colon i \neq j \ \wedge \ \bar{\sigma}_j(\vec{\nu}^*_{-j}, q) < \bar{\sigma}_j(\vec{\nu}^*). \tag{6.5}$$

We now fix such a node j. To derive a contradiction, we consider the expected cost incurred by j in $\vec{\mu}^*$, $\vec{\nu}^*$, and $(\vec{\nu}^*_{-j}, q)$ for various values of $q \in [0,1]$. These profiles differ only in the strategies of i and j. Therefore, we may limit ourselves to considering a subgame between these two players, where $x \in [0,1]$ and $y \in [0,1]$ represent the possible mixed strategies of i and j, respectively. We denote by

$$\hat{\sigma}_j(x,y) = \bar{\sigma}_j\left(\left(\vec{\mu}^*_{-i}, x\right)_{-j}, y\right)$$

[4]That is, if $\mu_i = \mathbb{1}$, then $\mu^*_i = 1$, and otherwise $\mu^*_i = 0$.

the expected cost incurred by j in the subgame when i plays x and j plays y.

To derive an explicit formula for $\hat{\sigma}_j(x,y)$, we first define the pure version of the subgame, where i and j have the pure strategies $s_i \in \{0,1\}$ and $s_j \in \{0,1\}$, respectively. The cost incurred by j in the pure subgame is the expected cost of the original n-player game when i plays s_i and j plays s_j, while the others play as in any of the profiles under consideration (see, e.g., $\vec{\mu}^*$). For the four pure strategy profiles of this subgame, we denote the cost incurred by j as

$$a = \hat{\sigma}_j(0,0), \quad b = \hat{\sigma}_j(1,0), \quad c = \hat{\sigma}_j(0,1), \quad d = \hat{\sigma}_j(1,1).$$

From these costs, $\hat{\sigma}_j(x,y)$ is obtained by weighting the cost incurred by j in a pure strategy profile by the probability that this profile occurs, given x and y:

$$\begin{aligned}\hat{\sigma}_j(x,y) &= (1-x)\cdot(1-y)\cdot a + (1-x)\cdot y \cdot b + x\cdot(1-y)\cdot c + x\cdot y\cdot d \\ &= (a-b+d-c)\cdot x\cdot y + (b-a)\cdot y + (c-a)\cdot x + a.\end{aligned}$$

Evidently, this function is linear in y, and y's coefficient can be rewritten as

$$(b-a)\cdot(1-x) + (d-c)\cdot x. \tag{6.6}$$

Because by Proposition 6.8, $b-a \leq 0$ and $d-c \leq 0$, it follows from (6.6) for any x that $y \mapsto \hat{\sigma}_j(x,y)$ is nonincreasing. If the function g defined by $g(y) = \hat{\sigma}_j(\nu_i^*, y) = \bar{\sigma}_j(\vec{\nu}^*_{-j}, y)$ is constant, then we reach a contradiction to (6.5).

So we consider the other case, that is, g is strictly decreasing. From this, it follows by (6.6) that $a > b$ or $c > d$. From this and (6.6) it follows for any $x \in]0,1[$ that the function $y \mapsto \hat{\sigma}_j(x,y)$ is strictly decreasing. In particular, as $\mu_i^* \in]0,1[$, the function h defined by $h(y) = \hat{\sigma}_j(\mu_i^*, y) = \bar{\sigma}_j(\vec{\mu}^*_{-j}, y)$ is strictly decreasing and therefore minimized at 1. As $\vec{\mu}^*$ is a Nash equilibrium, h is minimized at μ_j^* and since h is strictly decreasing, it follows that $\mu_j^* = 1$. The function g is also strictly decreasing and minimized at 1, and since $\mu_j^* = \nu_j^*$, it follows that ν_j^* minimizes g. By the definition of g, this means that for any $r \in [0,1]$ we have $\bar{\sigma}_j(\vec{\nu}^*_{-j}, r) \geq \bar{\sigma}_j(\vec{\nu}^*)$, which contradicts (6.5). ∎

7. Comparison with previous work. Busch, Surapaneni, and Tirthapura [3, 4] initiated the study of the work complexity of the link reversal algorithms presented in [10]. More precisely, they considered the pair and triple algorithms in their analysis. Regarding the triple algorithm, they considered any initializations of the triples, while — as discussed in section 2.2 — Gafni and Bertsekas [10] claimed correspondence between Partial Reversal and the triple algorithm only for special initializations of the latter, namely, if for each process i the height $(\alpha_i, \beta_i, id_i)$ initially satisfies $\alpha_i = 0$. In fact, in the case where initially there are different α values, executions of the triple algorithm are similar to Partial Reversal executions only after a preliminary phase during which the α values are aligned. This preliminary phase is reflected in the work complexity theorems by Busch, Surapaneni, and Tirthapura by a value α^* that is the difference between the minimal and maximal initial α values. If one wants to obtain results for Partial Reversal from their theorems, one has to set α^* to 0; i.e., all the initial α values are equal (under the assumption of the correspondence between the triple algorithm and Partial Reversal). We give a detailed comparison of these specializations to our results in this section.

Busch, Surapaneni, and Tirthapura analyzed the total number of reversals as a function of n_B, the number of nodes that have no directed path to D in the initial

graph (called *bad* nodes). For the pair algorithm, Busch, Surapaneni, and Tirthapura determined the exact number of reversals performed by any node. They partitioned the set of initially bad nodes into pair-layers[5] and proved that if a node belongs to the ℓth pair-layer, then it performs exactly ℓ reversals. Their definition of pair-layer is such that a node i is in the ℓth pair-layer if and only if our formula for i's work equals ℓ. Thus our results are the same as those in [3, 4] for the work complexity of Full Reversal. Busch, Surapaneni, and Tirthapura pointed out that in any graph, there are at most n_B pair-layers. Since n_B is never more than n, they concluded that the global work complexity is at most n^2. In our approach, an n^2 upper bound on the global work complexity follows from the fact that there are at most n links in a path from a node to the destination. As in [3, 4], we derive from our formula a particular family of graphs in which $\Omega(n^2)$ reversals are done: the graphs are chains of $n+1$ nodes in which D is at one end of the chain and all links are directed away from D (cf. the upper chain of Figure 5.1).

For the triple algorithm, Busch, Surapaneni, and Tirthapura calculated only upper and lower bounds on the number of reversals performed by any node. Moreover, their lower bound is defined only for certain graphs. In more detail, they partitioned the initially bad nodes into "levels" and showed that the level index of a node is an upper bound on the number of reversals performed by the node. It can be shown that for nodes that are initially sinks or sources, their upper bound is at least twice as large as our exact work expression, and as such is not tight.

For the lower bound, they partitioned the initially bad nodes into triple-layers[5] and showed that the number of reversals performed by a node is at least half the index of the triple-layer to which the node belongs. It can be shown that their lower bound is at most half as large as our exact work expression for nodes in $\mathcal{S}_{\mathbb{1}}$ or $\mathcal{S}_{\mathbb{0}}$, and is at most one-fourth as large as that for nodes in $\mathcal{D}$. Not all graphs can be decomposed into triple-layers, and thus their lower bounds make sense only for nodes in a subset of all routable acyclic graphs. In contrast, our results give an exact formula for the work complexity of all nodes in all routable acyclic graphs.

Arguments similar to those just given show that from the analysis by Busch, Surapaneni, and Tirthapura as well as from ours, one can conclude that the total number of reversals is $O(n^2)$ for both the (appropriately initialized) triple algorithm and Partial Reversal. Moreover, Corollary 5.1 establishes a formal understanding of the graphs used by Busch, Surapaneni, and Tirthapura to prove that their bounds are asymptotically tight. In particular, the worst case graph for the global work complexity of Partial Reversal given in Figure 7.1 is obtained by maximizing the number of nodes with two distinct incoming links (denoted by s in Corollary 5.1). Graphs constructed as the example from Figure 7.1 consisting of n nodes lead to a global work complexity of $\Omega(n^2)$.

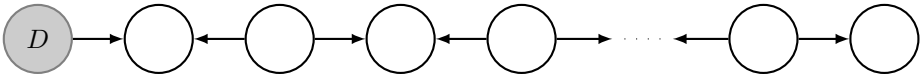


FIG. 7.1. *A worst case graph for global work complexity of Partial Reversal.*

In summary, our analysis and that of Busch, Surapaneni, and Tirthapura lead to exact expressions of the work complexity of Full Reversal. While we also obtain an

[5] In fact, Busch, Surapaneni, and Tirthapura used the same term "layer" for two different notions in the analysis of the pair and triple algorithms.

exact expression of the work complexity of Partial Reversal, the analysis by Busch, Surapaneni, and Tirthapura provides only an $O(n^2)$ upper bound, and a lower bound for some specific graphs. However, their bounds hold for the "general" triple algorithm with arbitrary initializations of the α values, which captures more than just the abstract Partial Reversal algorithm. For a different kind of generalization, namely, LR with initial labelings satisfying (AC), we give exact expressions for the work complexity of every node.

Further, section 6 gives a formal underpinning of the intuition we got from the examples in Figure 5.1: there are graphs where the work complexity for Full Reversal is much larger than for Partial Reversal (namely, by a factor $\frac{n+1}{2}$), while for all graphs, the work complexity of Partial Reversal is at most equal to twice the work complexity of Full Reversal. In other words, contrary to Full Reversal, Partial Reversal is never a disastrous strategy even if not optimum, and so appears as "less risky" than Full Reversal. Therefore, it may seem as if our results contradict the results by Busch and Tirthapura who stated in their conclusions [4] that "the Full Reversal algorithm outperforms the Partial Reversal algorithm in the worst case." However, they actually refer to the pair algorithm and the triple algorithm with arbitrary α initializations. The triple algorithm with arbitrary α initializations does not correspond to Partial Reversal, but rather to Partial Reversal preceded by a preliminary phase, which may be quite penalizing since it may incur an additional work complexity of $\alpha^* \cdot n$ (recall that α^* denotes the difference between the minimal and the maximal initial α values). Hence the pair algorithm outperforms the triple algorithm with arbitrary α initializations in general, and that explains the discrepancy between the conclusion in [4] and our conclusion concerning Full Reversal and Partial Reversal.

8. Conclusions. We presented a formal definition of the Full Reversal and Partial Reversal algorithms introduced by Gafni and Bertsekas [10]. This formalization unifies the two algorithms and consists of a general link reversal algorithm which assigns binary labels on the links and decides which links to reverse based on the current labeling. For our general algorithm, we have established an exact expression of the work complexity, thus giving the work complexity of both Full Reversal and Partial Reversal.

Using binary link labels very naturally leads to the use of game theory for analyzing the impact of the choice of initial labelings. Our main findings are that Full Reversal is a pure Nash equilibrium but has the worst social cost among all pure Nash equilibria. In fact, the price of anarchy — determined by the social cost of Full Reversal — is proportional to the number of nodes in the graph. On the other hand, although Partial Reversal is not necessarily a pure Nash equilibrium, its social cost is never more than twice the optimal. Game theory thus provides us with a formal basis to further compare the work complexity of Full Reversal and Partial Reversal.

Full Reversal and Partial Reversal were originally presented in a way hinting that superior performance could be obtained from Partial Reversal, and most of the work based on link reversal routing uses algorithms [21, 15, 9] whose dynamics are similar to Partial Reversal. However, no rigorous analysis has been yet given to support this intuition. Our game-theoretic study provides a rigorous basis for comparing Partial Reversal with Full Reversal and explains why Partial Reversal is actually preferable to Full Reversal.

Acknowledgments. We thank Matthias Függer for his helpful comments on the proof of Theorem 3.3. We are also grateful to the anonymous reviewers, whose constructive comments helped us to improve the presentation of our results.

REFERENCES

[1] H. Attiya, V. Gramoli, and A. Milani, *A provably starvation-free distributed directory protocol*, in Proceedings of the 12th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS), Lecture Notes in Comput. Sci. 6366, Springer, Berlin, 2010, pp. 405–419.

[2] V. C. Barbosa and E. Gafni, *Concurrency in heavily loaded neighborhood-constrained systems*, ACM Trans. Program. Lang. Syst., 11 (1989), pp. 562–584.

[3] C. Busch, S. Surapaneni, and S. Tirthapura, *Analysis of link reversal routing algorithms for mobile ad hoc networks*, in Proceedings of the 15th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA), ACM, New York, 2003, pp. 210–219.

[4] C. Busch and S. Tirthapura, *Analysis of link reversal routing algorithms*, SIAM J. Comput., 35 (2005), pp. 305–326.

[5] K. M. Chandy and J. Misra, *The drinking philosopher's problem*, ACM Trans. Program. Lang. Syst., 6 (1984), pp. 632–646.

[6] B. Charron-Bost, *Private communication*, 2007.

[7] B. Charron-Bost, A. Gaillard, J. L. Welch, and J. Widder, *Routing without ordering*, in Proceedings of the 21st ACM Symposium on Parallelism in Algorithms and Architectures (SPAA), ACM, New York, 2009, pp. 145–153.

[8] B. Charron-Bost, J. L. Welch, and J. Widder, *Link reversal: How to play better to work less*, in Proceedings of the 5th International Workshop on Algorithmic Aspects of Wireless Sensor Networks (Algosensors), Lecture Notes in Comput. Sci. 5304, Springer, Berlin, 2009, pp. 88–101.

[9] A. Derhab and N. Badache, *A self-stabilizing leader election algorithm in highly dynamic ad hoc mobile networks*, IEEE Trans. Parallel Distrib. Syst., 19 (2008), pp. 926–939.

[10] E. M. Gafni and D. P. Bertsekas, *Distributed algorithms for generating loop-free routes in networks with frequently changing topology*, IEEE Trans. Commun., 29 (1981), pp. 11–18.

[11] R. Ingram, P. Shields, J. E. Walter, and J. L. Welch, *An asynchronous leader election algorithm for dynamic networks*, in Proceedings of the IEEE International Parallel and Distributed Processing Symposium, IEEE Press, Piscataway, NJ, 2009, pp. 1–12.

[12] Y.-B. Ko and N. H. Vaidya, *Geotora: A protocol for geocasting in mobile ad hoc networks*, in Proceedings of the 2000 International Conference on Network Protocols (ICNP), IEEE Press, Piscataway, NJ, 2000, pp. 240–250.

[13] E. Koutsoupias and C. H. Papadimitriou, *Worst-case equilibria*, in Proceedings of the 16th Annual Symposium on Theoretical Aspects of Computer Science (STACS), Lecture Notes in Comput. Sci. 1563, Springer, Berlin, 1999, pp. 404–413.

[14] Y. Malka, S. Moran, and S. Zaks, *A lower bound on the period length of a distributed scheduler*, Algorithmica, 10 (1993), pp. 383–398.

[15] N. Malpani, J. L. Welch, and N. Vaidya, *Leader election algorithms for mobile ad hoc networks*, in Proceedings of the 4th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communication, ACM, New York, 2000, pp. 96–103.

[16] D. Monderer and L. S. Shapley, *Potential games*, Games and Economic Behavior, 14 (1996), pp. 124–143.

[17] M. Naimi, M. Trehel, and A. Arnold, *A log(n) distributed mutual exclusion algorithm based on path reversal*, J. Parallel Distrib. Comput., 34 (1996), pp. 1–13.

[18] J. Nash, *Non-cooperative games*, Ann. of Math., 54 (1951), pp. 286–295.

[19] N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani, eds., *Algorithmic Game Theory*, Cambridge University Press, New York, 2007.

[20] M. J. Osborne, *An Introduction to Game Theory*, Oxford University Press, New York, 2003.

[21] V. D. Park and M. S. Corson, *A highly adaptive distributed routing algorithm for mobile wireless networks*, in Proceedings of the 16th Conference on Computer Communications (Infocom), IEEE Press, Piscataway, NJ, 1997, pp. 1405–1413.

[22] T. Radeva and N. A. Lynch, *Partial reversal acyclicity*, in Proceedings of the 30th Annual ACM Symposium on Principles of Distributed Computing (PODC), ACM, New York, 2011, pp. 353–354.

[23] K. Raymond, *A tree-based algorithm for distributed mutual exclusion*, ACM Trans. Comput. Syst., 7 (1989), pp. 61–77.

[24] S. Tirthapura and M. Herlihy, *Self-stabilizing distributed queuing*, IEEE Trans. Parallel Distrib. Syst., 17 (2006), pp. 646–655.

[25] J. E. Walter, J. L. Welch, and N. H. Vaidya, *A mutual exclusion algorithm for ad hoc mobile networks*, Wireless Networks, 7 (2001), pp. 585–600.