Chapter 10
Sinkless Orientation

In this chapter we will study the complexity of *sinkless orientation*, a problem that was introduced in the previous chapter. This is a problem that is understood well: we will design algorithms and show that these are asymptotically optimal.

Recall that sinkless orientation is the problem of orienting the edges of the tree so that each internal node has got at least one outgoing edge. We begin by studying sinkless orientation on paths (or $(2,2)$-biregular trees), and show that we can easily argue about local neighborhoods to prove a tight lower bound result. However, when we switch to $(3,3)$-biregular trees, we will need the round elimination technique to do the same.

10.1 Sinkless Orientation on Paths

We define *sinkless orientation on paths* to be the following bipartite locally verifiable problem $\Pi = (\Sigma, \mathbf{A}, \mathbf{P})$. The alphabet is $\Sigma = \{\mathsf{I}, \mathsf{O}\}$, with the interpretation that I indicates that the edge is oriented towards the active node ("incoming") and O indicates that the edge is oriented away from the active node ("outgoing"). Each active node must label at least one incident edge with O, and thus the active configurations are

$$\mathbf{A} = \big\{ [\mathsf{O}, \mathsf{I}], [\mathsf{O}, \mathsf{O}] \big\}.$$

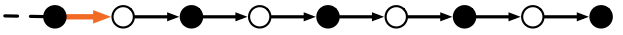

Figure 10.1: Propagation of a sinkless orientation on paths. Orienting a single edge (orange) forces the orientation of the path all the way to the other endpoint.

Each passive node must have at least one incident edge labeled with I, and thus the passive configurations are

$$\mathbf{P} = \big\{ [\mathsf{I}, \mathsf{O}], [\mathsf{I}, \mathsf{I}] \big\}.$$

As previously, nodes of degree 1 are unconstrained; the edges incident to them can be labeled arbitrarily.

10.1.1 Hardness of Sinkless Orientation

We begin by showing that solving sinkless orientation requires $\Omega(n)$ rounds on $(2,2)$-biregular trees.

Lemma 10.1. *Solving sinkless orientation on $(2,2)$-biregular trees in the bipartite* PN*-model requires at least $n/4-1$ rounds, even if the nodes know the value n.*

Let us first see why the lemma is intuitively true. Consider a path, as illustrated in Figure 10.1. Each active node u must choose some label for its incident edges, and at least one of these labels must be O. Then its passive neighbor v over the edge labeled with O must have its other incident edge labeled I. This further implies that the other active neighbor w of v must label its other edge with O. The original output of u *propagates* through the path and the outputs of other nodes far away from u depend on the output of u.

Let us now formalize this intuition.

Proof of Lemma 10.1. Consider any algorithm A running in $T(n) = o(n)$ rounds. Then there exists n_0 such that for all $n \geq n_0$, we have that $T(n) \leq (n-5)/4$. Now fix such an integer n and let $T = T(n)$ denote the running time of the algorithm.

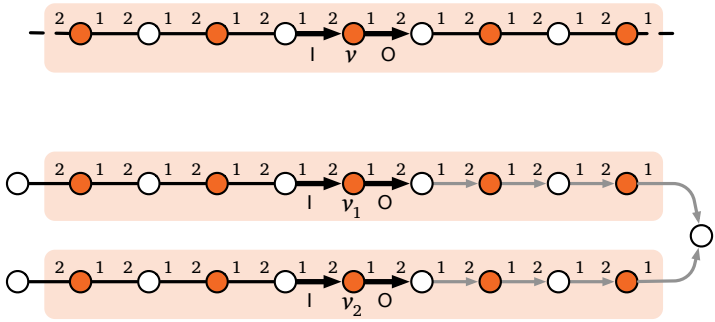

Figure 10.2: Sinkless orientation lower bound. Assume $T = 4$. Top: any algorithm must fix some output labeling with an outgoing edge for a fixed neighborhood $\text{ball}_N(v, 4)$. Bottom: Copying the same 4-neighborhood twice, and arranging the copies towards the same node creates an input N' where the two nodes orient towards the middle. There is no legal way to label the rest of the path.

Consider an active node v in the middle of a path N on n nodes. Let $\text{ball}_N(v, T)$ denote the T-neighborhood of v. Assume that $\text{ball}_N(v, T)$ is consistently port-numbered from *left* to *right*, as illustrated in Figure 10.2. Node v must use the output label O on one of its incident edges; without loss of generality, assume that this is port 1. We can now construct a counterexample N' as follows. Take two copies of $\text{ball}_N(v, T)$, denoted by $\text{ball}_{N'}(v_1, T)$ and $\text{ball}_{N'}(v_2, T)$. In particular, this includes the port-numbering in $\text{ball}_N(v, T)$. Add one new node that is connected to the *right* endpoint of both $\text{ball}_{N'}(v_1, T)$ and $\text{ball}_{N'}(v_2, T)$. Finally, add leaves at the other endpoints of $\text{ball}_{N'}(v_1, T)$ and $\text{ball}_{N'}(v_2, T)$; see Figure 10.2 for an illustration. We claim that the algorithm A fails on N'.

By definition, edges must be labeled alternatively O and I starting from both v_1 and v_2. Therefore we must eventually find an active node labeled [I, I] or a passive node labeled [O, O], an incorrect solution.

The total number of nodes in N' is $n = 2(2T + 1) + 3 = 4T + 5$, giving $T = (n-5)/4$. Thus solving sinkless orientation requires at least $T + 1 = (n-5)/4 + 1 \geq n/4 - 1$ rounds, as required. □

10.1.2 Solving Sinkless Orientation on Paths

The proof of Lemma 10.1 shows that it is impossible to solve sinkless orientation on paths in sublinear number of rounds. Now we will show a linear upper bound: it is possible to solve sinkless orientation once all nodes see an endpoint of the path.

Lemma 10.2. *Sinkless orientation can be solved in* $\lfloor n/2 \rfloor$ *rounds in the bipartite* PN*-model on* $(2,2)$*-biregular trees.*

Proof. The idea of the algorithm is the following: initially send messages from the leaves towards the middle of the path. Orient edges against the incoming messages, i.e., toward the closest leaf. Once the messages reach the midpoint of the path, all edges have been correctly oriented away from the midpoint.

The algorithm works as follows. Initially all non-leaf nodes wait. The leaf nodes send a message to their neighbor and stop. If they are active, they output I on their incident edge. Whenever a node receives a message for the first time, in the next round it sends a message to the other port and stops. If it is an active node, it outputs O in the port from which it received the message, and I in the other port. That is, it orients its incident edges towards the closer leaf. If a node receives two messages in the same round, it is the midpoint of the path; it does not send any further messages. If it is an active node, it outputs O on both of its incident edges.

The algorithm runs in $\lfloor n/2 \rfloor$ rounds: on paths with an even number of nodes, all nodes have received a message in round $n/2-1$, and thus stop in the next round. On paths with an odd number of nodes, the middle node receives two messages in round $\lfloor n/2 \rfloor$ and stops.

It remains to show that our algorithm is correct. All leaf nodes are trivially labeled correctly. Any active non-leaf node always has an incident label O. Now consider a passive node u: there is an active v that sends u a message before stopping. This node will output I on $\{u, v\}$, and thus u is also correctly labeled. □

Theorem 10.3. *The complexity of sinkless orientation on paths is* $\Theta(n)$.

Proof. Follows from Lemmas 10.1 and 10.2. □

10.2 Sinkless Orientation on Trees

In Section 10.1 we saw that if we require that degree-2 nodes have at least one outgoing edge, we arrive at a problem that is hard already in the case of paths. The proof of hardness was a straightforward argument that used local neighborhoods.

However, what happens if we relax the problem slightly and allow any orientation around degree-2 nodes? The proof of hardness from Section 10.1.1 no longer works, but does the problem get easier to solve?

For concreteness, let us consider *trees of maximum degree* 3, that is, both active and passive nodes have degree at most 3; the case of higher degrees is very similar. We define the problem so that nodes of degree 1 and 2 are unconstrained, but nodes of degree 3 must have at least one outgoing edge. We can encode it as follows as a bipartite locally verifiable problem $\Pi_0 = (\Sigma_0, \mathbf{A}_0, \mathbf{P}_0)$:

$$\Sigma_0 = \{\mathsf{O}, \mathsf{I}\},$$
$$\mathbf{A}_0 = \big\{[\mathsf{O}], [\mathsf{I}], [\mathsf{O},\mathsf{O}], [\mathsf{O},\mathsf{I}], [\mathsf{I},\mathsf{I}], [\mathsf{O},\mathsf{I},\mathsf{I}], [\mathsf{O},\mathsf{O},\mathsf{I}], [\mathsf{O},\mathsf{O},\mathsf{O}]\big\},$$
$$\mathbf{P}_0 = \big\{[\mathsf{O}], [\mathsf{I}], [\mathsf{O},\mathsf{O}], [\mathsf{O},\mathsf{I}], [\mathsf{I},\mathsf{I}], [\mathsf{I},\mathsf{O},\mathsf{O}], [\mathsf{I},\mathsf{I},\mathsf{O}], [\mathsf{I},\mathsf{I},\mathsf{I}]\big\}.$$

Here we have listed all possible configurations for nodes of degrees 1, 2, and 3.

10.2.1 Solving Sinkless Orientation on Trees

The algorithm for solving sinkless orientation on trees uses ideas similar to the algorithm on paths: each node u must determine the closest unconstrained node v, i.e., a node of degree 1 or 2, and the path from u to v is oriented away from u. This will make all nodes happy: each active node of degree 3 has an outgoing edge, and all other nodes are unconstrained.

Let us call nodes of degree 1 and 2 *special nodes*. We must be careful in how the nodes choose the special node: the algorithm would fail if two nodes want to orient the same edge in different directions.

The algorithm functions as follows. In the first round, only special nodes are sending messages, broadcasting to each port. Then the special nodes stop and, if they are active nodes, they output I on each edge. Nodes of degree 3 *wake up* in the first round in which they receive at least one message. In the next round they broadcast to each port from which they did not receive a message in the previous round. After sending this message, the nodes stop. If they are active nodes, they orient their incident edges towards the smallest port from which they received a message: output O on that edge, and I on the other edges.

Correctness. Assume that the closest special nodes are at distance t from some node u. Assume that v is one of those nodes, and let $(v_1, v_2, \ldots, v_{t+1})$ denote the unique path from $v = v_1$ to $u = v_{t+1}$. We claim that in each round i, node v_i broadcasts to v_{i+1}. By assumption, v is also one of the closest special nodes to all v_i; otherwise there would be a closer special node to u as well. In particular, there will never be a broadcast from v_{i+1} to v_i, as then v_{i+1} would have a different closer special node. Therefore each v_i will broadcast to v_{i+1} in round i. This implies that in round t, node u will receive a broadcast from v_t.

All nodes that receive a broadcast become happy: Active nodes output O on one of the edges from which they received a broadcast, making them happy. They output I on the other edges, so each passive node is guaranteed that every edge from which it receives a broadcast has the label I.

Time Complexity. It remains to bound the round by which all nodes have received a broadcast. To do this, we observe that each node is at distance $O(\log n)$ from a special node.

Consider a fragment of a 3-regular tree centered around some node v, and assume that there are no special nodes near v. Then at distance 1 from v there are 3 nodes, at distance 2 there are 6 nodes, at distance 3

there are 12 nodes, and so on. In general, if we do not have any special nodes within distance i, then at distance i there are $3 \cdot 2^{i-1} > 2^i$ nodes in the tree. At distance $i = \log_2 n$, we would have more than n nodes. Thus, within distance $\log_2 n$, there has to be a special node. Since each node can stop once it has received a broadcast, the running time of the algorithm is $O(\log n)$.

10.2.2 Roadmap: Next Steps

We have seen that sinkless orientation in trees can be solved in $O(\log n)$ rounds. We would like to now prove a matching lower bound and show that sinkless orientation cannot be solved in $o(\log n)$ rounds. We will apply the round elimination technique from Chapter 9 to do this. However, we will need one further refinement to the round elimination technique that will make our life a lot easier: we can *ignore all non-maximal configurations*. We will explain this idea in detail in Section 10.3, and then we are ready to prove the hardness result in Section 10.4.

10.3 Maximal Output Problems

In Chapter 9 we saw how to use the round elimination technique to construct the *output problem* $\Pi' = \mathrm{re}(\Pi)$ for any given bipartite locally verifiable problem Π. We will now make an observation that allows us to *simplify* the description of output problems. We will change the definition of output problems to include this simplification.

Consider an output problem $\Pi' = (\Sigma, \mathbf{A}, \mathbf{P})$. Recall that Σ now consists of *sets* of labels. Assume that there are two configurations

$$X = [X_1, X_2, \ldots, X_d],$$
$$Y = [Y_1, Y_2, \ldots, Y_d],$$

in $\mathbf{A}$. We say that Y *contains* X if we have $X_i \subseteq Y_i$ for all i.

If Y contains X, then configuration X is redundant; whenever an algorithm solving Π' would like to use the configuration X, it can equally well use Y instead:

- Active nodes are still happy if active nodes switch from X to Y: By assumption, Y is also a configuration in $\mathbf{A}$.

- Passive nodes are still happy if active nodes switch from X to Y: Assume that $Z = [Z_1, Z_2, \ldots, Z_\delta]$ is a passive configuration in $\mathbf{P}$. As this is a passive configuration of $\mathrm{re}(\Pi)$, it means that there is a choice $z_i \in Z_i$ such that $[z_1, z_2, \ldots, z_\delta]$ is an active configuration in the original problem Π. But now if we replace each Z_i with a superset $Z'_i \supseteq Z_i$, then we can still make the same choice $z_i \in Z'_i$, and hence $Z' = [Z'_1, Z'_2, \ldots, Z'_\delta]$ also has to be a passive configuration in $\mathbf{P}$. Therefore replacing a label with its superset is always fine from the perspective of passive nodes, and in particular switching from X to Y is fine.

Therefore we can *omit* all active configurations that are contained in another active configuration and only include the *maximal* configurations, i.e., configurations that are not contained in any other configuration.

We get the following mechanical process for constructing the output problem $\mathrm{re}(\Pi) = (\Sigma, \mathbf{A}, \mathbf{P})$.

(a) Construct the output problem $\mathrm{re}(\Pi) = (\Sigma, \mathbf{A}, \mathbf{P})$ as described in Section 9.2.2.
(b) Remove all non-maximal active configurations from $\mathbf{A}$.
(c) Remove all unused elements from Σ.
(d) Remove all passive configurations containing labels not in Σ.

The resulting problem is exactly as hard to solve as the original problem:

- Since the simplified sets of configurations are subsets of the original sets of configurations, any solution to the simplified problem is a solution to the original problem, and thus the original problem is *at most as hard as* the simplified problem.

- By construction, any algorithm solving the original output problem can solve the simplified problem equally fast, by replacing some

labels by their supersets as appropriate. Therefore the original problem is *at least as hard as* the simplified problem.

We will apply this simplification when we use the round elimination technique to analyze the sinkless orientation problem.

10.4 Hardness of Sinkless Orientation on Trees

We will now show that sinkless orientation requires $\Omega(\log n)$ rounds on $(3,3)$-biregular trees—and therefore also in trees of maximum degree 3, as $(3,3)$-biregular trees are a special case of such trees.

Let us first write down the sinkless orientation problem as a bipartite locally verifiable problem $\Pi_0 = (\Sigma_0, \mathbf{A}_0, \mathbf{P}_0)$ in $(3,3)$-biregular trees; as before, we will only keep track of the configurations for nodes of degree 3, as leaf nodes are unconstrained:

$$\begin{aligned}
\Sigma_0 &= \{\mathsf{O}, \mathsf{I}\},\\
\mathbf{A}_0 &= \left\{ [\mathsf{O}, x, y] \;\middle|\; x, y \in \Sigma \right\},\\
\mathbf{P}_0 &= \left\{ [\mathsf{I}, x, y] \;\middle|\; x, y \in \Sigma \right\}.
\end{aligned}$$

10.4.1 First Step

Lemma 10.4. *Let Π_0 be the sinkless orientation problem. Then the output problem is $\Pi_1 = \mathrm{re}(\Pi_0) = (\Sigma_1, \mathbf{A}_1, \mathbf{P}_1)$, where*

$$\begin{aligned}
\Sigma_1 &= \left\{ \{\mathsf{I}\}, \{\mathsf{O}, \mathsf{I}\} \right\},\\
\mathbf{A}_1 &= \left\{ \left[\{\mathsf{I}\}, \{\mathsf{O}, \mathsf{I}\}, \{\mathsf{O}, \mathsf{I}\} \right] \right\},\\
\mathbf{P}_1 &= \left\{ \left[\{\mathsf{O}, \mathsf{I}\}, x, y \right] \;\middle|\; x, y \in \Sigma_1 \right\}.
\end{aligned}$$

Proof. Let us follow the procedure from Section 10.3. First, we arrive at alphabet Σ_1 that contains all non-empty subsets of Σ_0:

$$\Sigma_1 = \left\{ \{\mathsf{O}\}, \{\mathsf{I}\}, \{\mathsf{O}, \mathsf{I}\} \right\}.$$

The active configurations $\mathbf{A}_1$ consist of all multisets $[X, Y, Z]$ such that no matter how we choose $x \in X$, $y \in Y$, and $z \in Z$, at least one element of the multiset $[x, y, z]$ is I. This happens exactly when at least one of the labels X, Y, and Z is the singleton set $\{\mathsf{I}\}$. We get that

$$\begin{aligned}\mathbf{A}_1 &= \Big\{ [\{\mathsf{I}\}, X, Y] \,\Big|\, X, Y \subseteq \{\mathsf{O}, \mathsf{I}\} \Big\} \\ &= \Big\{ \big[\{\mathsf{I}\}, \{\mathsf{I}\}, \{\mathsf{I}\}\big], \\ &\quad \big[\{\mathsf{I}\}, \{\mathsf{I}\}, \{\mathsf{O}\}\big], \\ &\quad \big[\{\mathsf{I}\}, \{\mathsf{I}\}, \{\mathsf{O}, \mathsf{I}\}\big], \\ &\quad \big[\{\mathsf{I}\}, \{\mathsf{O}\}, \{\mathsf{O}\}\big], \\ &\quad \big[\{\mathsf{I}\}, \{\mathsf{O}\}, \{\mathsf{O}, \mathsf{I}\}\big], \\ &\quad \big[\{\mathsf{I}\}, \{\mathsf{O}, \mathsf{I}\}, \{\mathsf{O}, \mathsf{I}\}\big] \Big\}.\end{aligned}$$

Next we remove all non-maximal configurations. We note that all other active configurations are contained in the configuration

$$\big[\{\mathsf{I}\}, \{\mathsf{O}, \mathsf{I}\}, \{\mathsf{O}, \mathsf{I}\}\big].$$

This becomes the only active configuration:

$$\mathbf{A}_1 = \Big\{ \big[\{\mathsf{I}\}, \{\mathsf{O}, \mathsf{I}\}, \{\mathsf{O}, \mathsf{I}\}\big] \Big\}.$$

Since the label $\{\mathsf{O}\}$ is never used, we may remove it from the alphabet, too: we get that

$$\Sigma_1 = \big\{ \{\mathsf{I}\}, \{\mathsf{O}, \mathsf{I}\} \big\}.$$

The passive configurations are all multisets such that at least one label contains O. Thus the simplified passive configurations are

$$\begin{aligned}\mathbf{P}_1 = \Big\{ &\big[\{\mathsf{O}, \mathsf{I}\}, \{\mathsf{O}, \mathsf{I}\}, \{\mathsf{O}, \mathsf{I}\}\big], \\ &\big[\{\mathsf{O}, \mathsf{I}\}, \{\mathsf{O}, \mathsf{I}\}, \{\mathsf{I}\}\big], \\ &\big[\{\mathsf{O}, \mathsf{I}\}, \{\mathsf{I}\}, \{\mathsf{I}\}\big] \Big\}.\end{aligned}$$

□

10.4.2 Equivalent Formulation

Now let us simplify the notation slightly. We say that a problem Π' is *equivalent* to another problem Π if a solution of Π' can be converted in zero rounds to a solution of Π and vice versa. In particular, equivalent problems are exactly as hard to solve.

Lemma 10.5. *Let Π_0 be the sinkless orientation problem. Then the output problem* $\mathrm{re}(\Pi_0)$ *is equivalent to $\Pi_1 = (\Sigma_1, \mathbf{A}_1, \mathbf{P}_1)$, where*

$$\begin{aligned}
\Sigma_1 &= \{\mathsf{A}, \mathsf{B}\},\\
\mathbf{A}_1 &= \big\{[\mathsf{A}, \mathsf{B}, \mathsf{B}]\big\},\\
\mathbf{P}_1 &= \big\{[\mathsf{B}, x, y] \;\big|\; x, y \in \Sigma_1\big\}.
\end{aligned}$$

Proof. Rename the labels of $\mathrm{re}(\Pi_0)$ as follows to arrive at Π_1:

$$\begin{aligned}
\mathsf{A} &= \{\mathsf{I}\},\\
\mathsf{B} &= \{\mathsf{O}, \mathsf{I}\}.
\end{aligned}$$

□

In what follows, we will use Π_1 and $\mathrm{re}(\Pi_0)$ interchangeably, as they are equivalent.

10.4.3 Fixed Points in Round Elimination

As we will see soon, problem Π_1 is a *fixed point* in round elimination: when round elimination is applied to Π_1, the output problem is again Π_1 (or, more precisely, a problem equivalent to Π_1).

This means that if we assume that Π_1 can be solved in T rounds, then by applying round elimination T times we get a 0-round algorithm for Π_1. It can be shown that Π_1 is *not* 0-round solvable. This would seem to imply that Π_1, and thus sinkless orientation, are not solvable at all, which would contradict the existence of the $O(\log n)$-time algorithm presented in Section 10.2.1!

To resolve this apparent contradiction, we must take a closer look at the assumptions that we have made. The key step in round elimination

happens when a node u simulates the *possible* outputs of its neighbors. The correctness of this step assumes that the possible T-neighborhoods of the neighbors are *independent* given the $(T-1)$-neighborhood of u. When T is so large in comparison with n that the T-neighborhoods of the neighbors put together might already imply the existence of more than n nodes, this assumption no longer holds—see Figure 10.3 for an example.

For the remainder of this chapter we consider algorithms that know the value n, the number of nodes in the graph. This allows us to define a *standard form* for algorithms that run in $T = T(n)$ rounds, where $T(n)$ is some function of n: since n is known, each node can calculate $T(n)$, gather everything up to distance $T(n)$, and simulate any $T(n)$-time algorithm.

In (d, δ)-biregular trees, where $d > 2$, it can be shown that round elimination can be applied if the initial algorithm is assumed to have running time $T(n) = o(\log n)$: this guarantees the independence property in the simulation step. However, round elimination fails for some function $T(n) = \Theta(\log n)$; calculating this threshold is left as Exercise 10.7.

Any problem that can be solved in time $T(n)$ can be solved in time $T(n)$ with an algorithm in the standard form. If the problem is a fixed point and $T(n) = o(\log n)$, we can apply round elimination. We get the following lemma.

Lemma 10.6. *Assume that bipartite locally verifiable problem Π on (d, d)-biregular trees, for $d > 2$, is a fixed point. Then the deterministic complexity of Π in the bipartite* PN*-model is either 0 rounds or $\Omega(\log n)$ rounds, even if the number of nodes n is known.*

10.4.4 Sinkless Orientation Gives a Fixed Point

We will now show that the output problem $\Pi_1 = \mathrm{re}(\Pi_0)$ of the sinkless orientation problem Π_0 is a fixed point, that is, $\mathrm{re}(\Pi_1)$ is a problem equivalent to Π_1 itself. Since this problem cannot be solved in 0 rounds, it requires $\Omega(\log n)$ rounds. As sinkless orientation requires, by defini-

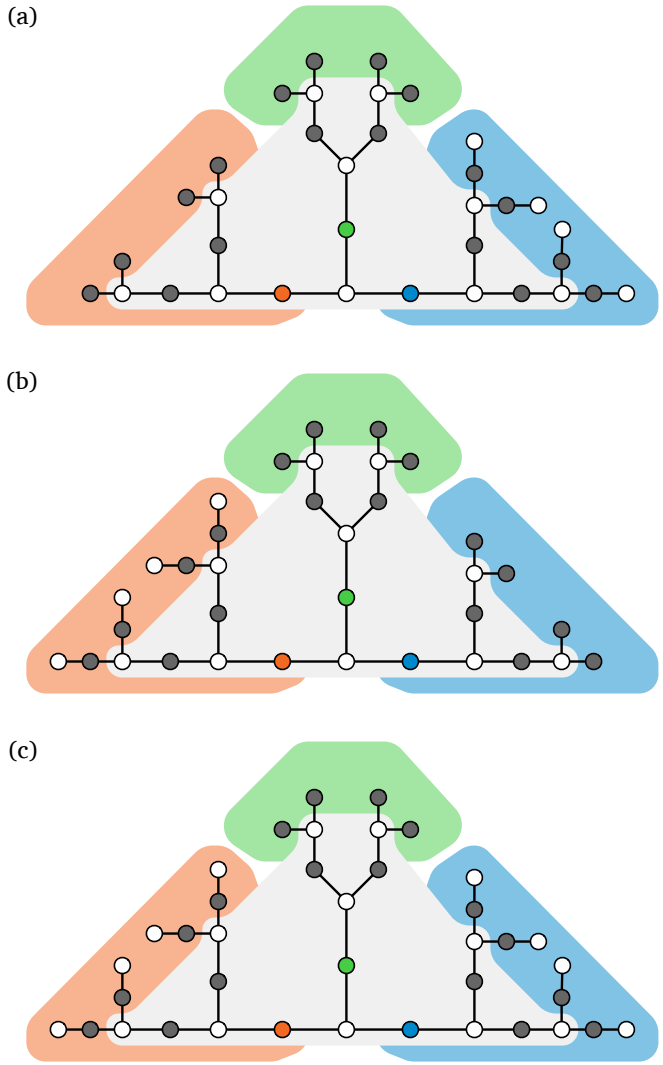

Figure 10.3: If we know that, e.g., $n = 35$, then orange, green, and blue extensions are no longer independent of each other: inputs (a) and (b) are possible but we cannot combine them arbitrarily to form e.g. input (c).

tion, one more round than its output problem, sinkless orientation also requires $\Omega(\log n)$ rounds.

Lemma 10.7. *The output problem* $\Pi_1 = (\Sigma_1, \mathbf{A}_1, \mathbf{P}_1)$ *of sinkless orientation, given by*

$$\begin{aligned}\Sigma_1 &= \{\mathsf{A}, \mathsf{B}\},\\ \mathbf{A}_1 &= \big\{[\mathsf{A}, \mathsf{B}, \mathsf{B}]\big\},\\ \mathbf{P}_1 &= \big\{[\mathsf{B}, x, y] \;\big|\; x, y \in \Sigma_1\big\},\end{aligned}$$

is a fixed point.

Proof. Let $\Pi_2 = \mathrm{re}(\Pi_1) = (\Sigma_2, \mathbf{A}_2, \mathbf{P}_2)$ denote the output problem of Π_1. Again, we have that

$$\Sigma_2 = \big\{\{\mathsf{A}\}, \{\mathsf{B}\}, \{\mathsf{A}, \mathsf{B}\}\big\}.$$

The active configurations $\mathbf{A}_2$ are

$$\mathbf{A}_2 = \Big\{\big[\{\mathsf{B}\}, x, y\big] \;\Big|\; x, y \subseteq \{\mathsf{A}, \mathsf{B}\}\Big\}.$$

That is, one set must be the singleton $\{\mathsf{B}\}$ to satisfy $\mathbf{P}_1$ for *all choices*, and the remaining sets are arbitrary.

Next we determine the maximal configurations. Again, there is a single active configuration that covers the other configurations:

$$\mathbf{A}_2 = \Big\{\big[\{\mathsf{B}\}, \{\mathsf{A}, \mathsf{B}\}, \{\mathsf{A}, \mathsf{B}\}\big]\Big\}.$$

The alphabet is immediately simplified to

$$\Sigma_2 = \big\{\{\mathsf{B}\}, \{\mathsf{A}, \mathsf{B}\}\big\},$$

as the label $\{\mathsf{A}\}$ is never used by any active configuration.

The passive configurations $\mathbf{P}_2$ are all multisets that contain the active configuration $[\mathsf{A}, \mathsf{B}, \mathsf{B}]$. Since A is now only contained in $\{\mathsf{A}, \mathsf{B}\}$, we get

that

$$\mathbf{P}_2 = \Big\{ \big[\{\mathsf{A},\mathsf{B}\}, \{\mathsf{A},\mathsf{B}\}, \{\mathsf{A},\mathsf{B}\}\big], \\ \big[\{\mathsf{A},\mathsf{B}\}, \{\mathsf{A},\mathsf{B}\}, \{\mathsf{B}\}\big], \\ \big[\{\mathsf{A},\mathsf{B}\}, \{\mathsf{B}\}, \{\mathsf{B}\}\big] \Big\}.$$

Now we may do a simple renaming trick to see that Π_2 is equivalent to Π_1: rename $\{\mathsf{B}\} \to \mathsf{A}$ and $\{\mathsf{A},\mathsf{B}\} \to \mathsf{B}$. Written this way, we have that Π_2 is equivalent to the following problem:

$$\begin{aligned} \Sigma_2 &= \{\mathsf{A},\mathsf{B}\}, \\ \mathbf{A}_2 &= \big\{ [\mathsf{A},\mathsf{B},\mathsf{B}] \big\}, \\ \mathbf{P}_2 &= \big\{ [\mathsf{B},x,y] \;\big|\; x,y \in \Sigma_2 \big\}, \end{aligned}$$

which is exactly the same problem as Π_1. □

10.5 Quiz

Calculate the *number of different* 2*-round algorithms* in the PN model on $(3,3)$-biregular trees for bipartite locally verifiable problems with the binary alphabet $\Sigma = \{0,1\}$.

Here two algorithms A_1 and A_2 are considered to be *different* if there is some port-numbered network N and some edge e such that the label edge e in the output of A_1 is different from the label of the same edge e in algorithm A_2. Note that A_1 and A_2 might solve the same problem, they just solved it differently. Please remember to take into account that in a $(3,3)$-biregular tree each node has degree 1 or 3.

Please give your answer in the scientific notation with two significant digits: your answer should be a number of the form $a \cdot 10^b$, where a is rounded to two decimal digits, we have $1 \le a < 10$, and b is a natural number.

10.6 Exercises

Exercise 10.1 (0-round solvability)**.** Prove that the following problems are not 0-round solvable.

(a) $(d+1)$-edge coloring in (d,d)-biregular trees (see Section 9.1.2).

(b) Maximal matching in (d,d)-biregular trees (see Section 9.1.2).

(c) Π_1, the output problem of sinkless orientation, in $(3,3)$-biregular trees (see Section 10.4.2).

▷ *hint A*

Exercise 10.2 (higher degrees)**.** Generalize the sinkless orientation problem from $(3,3)$-biregular trees to $(10,10)$-biregular trees. Apply round elimination and show that you get again a fixed point.

Exercise 10.3 (non-bipartite sinkless orientation)**.** Define non-bipartite sinkless orientation as the following problem $\Pi = (\Sigma, \mathbf{A}, \mathbf{P})$ on $(3,2)$-biregular trees:

$$\begin{aligned}\Sigma &= \{\mathsf{O},\mathsf{I}\},\\ \mathbf{A} &= \left\{ [\mathsf{O},x,y] \;\middle|\; x,y \in \Sigma \right\},\\ \mathbf{P} &= \left\{ [\mathsf{I},\mathsf{O}] \right\}.\end{aligned}$$

Prove that applying round elimination to Π leads to a period-2 point, that is, to a problem Π' such that $\Pi' = \mathrm{re}(\mathrm{re}(\Pi'))$.

Exercise 10.4 (matching lower bound)**.** Let us define *sloppy perfect matching* in trees as a matching such that all non-leaf nodes are matched. Encode this problem as a bipartite locally verifiable problem on $(3,3)$-biregular trees. Show that solving it requires $\Omega(\log n)$ rounds in the PN-model with deterministic algorithms.

Exercise 10.5 (matching upper bound)**.** Consider the sloppy perfect matching problem from Exercise 10.4. Design an algorithm for solving it with a deterministic PN-algorithm on $(3,3)$-biregular trees in $O(\log n)$ rounds.

▷ *hint B*

Exercise 10.6 (sinkless and sourceless)**.** Sinkless and sourceless orientation is the problem of orienting the edges of the graph so that each non-leaf node has at least one outgoing edge *and* at least one incoming edge. Encode the sinkless and sourceless orientation problem as a binary locally verifiable labeling problem on $(5,5)$-biregular trees. Design an algorithm for solving sinkless and sourceless orientation on $(5,5)$-biregular trees.

Exercise 10.7 (failure of round elimination)**.** In Section 10.4.3 we discussed that round elimination fails if in the simulation step the $T(n)$-neighborhoods of the neighbors are dependent from each other. This happens when there exist T-neighborhoods of the neighbors such that the resulting tree would have more than n nodes. Consider (d,d)-biregular trees. Calculate the bound for $F(n)$ such that round elimination fails for algorithms with running time $T(n) \geq F(n)$.

⋆ **Exercise 10.8.** Design an algorithm for solving sinkless and sourceless orientation on $(3,3)$-biregular trees.

10.7 Bibliographic Notes

Brandt et al. [2] introduced the sinkless orientation problem and proved that it cannot be solved in $o(\log \log n)$ rounds with randomized algorithms, while Chang et al. [3] showed that it cannot be solved in $o(\log n)$ rounds with deterministic algorithms. Ghaffari and Su [5] gave matching upper bounds.

Exercises 10.4 and 10.5 are inspired by [1], and Exercises 10.6 and 10.8 are inspired by [4].

10.8 Bibliography

[1] Alkida Balliu, Sebastian Brandt, Yuval Efron, Juho Hirvonen, Yannic Maus, Dennis Olivetti, and Jukka Suomela. Classification of distributed binary labeling problems. In *Proc. 34th International Sympo-*

sium on Distributed Computing (DISC 2020), 2020. arXiv:1911.13294, doi:10.4230/LIPIcs.DISC.2020.17.

[2] Sebastian Brandt, Orr Fischer, Juho Hirvonen, Barbara Keller, Tuomo Lempiäinen, Joel Rybicki, Jukka Suomela, and Jara Uitto. A lower bound for the distributed Lovász local lemma. In *Proc. 48th ACM Symposium on Theory of Computing (STOC 2016)*, 2016. arXiv:1511.00900, doi:10.1145/2897518.2897570.

[3] Yi-Jun Chang, Tsvi Kopelowitz, and Seth Pettie. An exponential separation between randomized and deterministic complexity in the LOCAL model. In *Proc. 57th IEEE Symposium on Foundations of Computer Science (FOCS 2016)*, pages 615–624. IEEE, 2016. arXiv:1602.08166, doi:10.1109/FOCS.2016.72.

[4] Mohsen Ghaffari, Juho Hirvonen, Fabian Kuhn, Yannic Maus, Jukka Suomela, and Jara Uitto. Improved distributed degree splitting and edge coloring. *Distributed Computing*, 33:293–310, 2020. arXiv:1706.04746, doi:10.1007/s00446-018-00346-8.

[5] Mohsen Ghaffari and Hsin-Hao Su. Distributed degree splitting, edge coloring, and orientations. In *Proc. 28th ACM-SIAM Symposium on Discrete Algorithms (SODA 2017)*, 2017. arXiv:1608.03220, doi:10.1137/1.9781611974782.166.

10.9 Hints

A. Show that a 0-round algorithm consists of choosing one active configuration and assigning its labels to the ports. Show that for any way of assigning the outputs to the ports, there exists a port numbering such that the incident edges of a passive node are not labeled according to any passive configuration.

B. Decompose the graph into *layers* $(V_0, V_1, \ldots, V_L)$, where nodes in layer i have distance i to the closest leaf. Then iteratively solve

the problem, starting from layer V_L: match all nodes in layer V_L, V_{L-1}, and so on.