University of Freiburg
Dept. of Computer Science
Prof. Dr. F. Kuhn
Z. Parsaeian, G. Schmid

Distributed Graph Algorithms
Sample Solution Exercise Sheet 6

LCLs in paths and Cycles

In this exercise sheet, we want to talk about what possible complexities **L**ocally **C**heckable **L**abelings (LCLs) can have, when we restrict the networks to only path and cycle topologies.

1 Locally Checkable Labelings

An LCL is a tuple (Σ, r, C) comprised of the following

- A set of outputlabels Σ (e.g. for 3-coloring $\Sigma = \{R, G, B\} \equiv \{1, 2, 3\}$)
- An integer $r \in \mathbb{N}$ called the radius.
- A set C of Σ labeled r-hop balls.

That is, each element of the set C is just a center node, with an arbitrary r-hop neighborhood, where each node in that neighborhood is assigned an output from Σ.
The set C is the set of *allowed* neighborhoods and we call each such neighborhood a configuration. In a solution to our problem, the output-labeled r-hop neighborhood $\mathcal{N}_r(v)$ of every node v in our graph, must be isomorphic to (so must look like) one of the neighborhoods in C.
Two such set of configuraitons are highlighted in Figures 1 and 2 for the MIS and 3 coloring problems in cycles. Note that for both of these problems the radius is equal to one ($r = 1$).

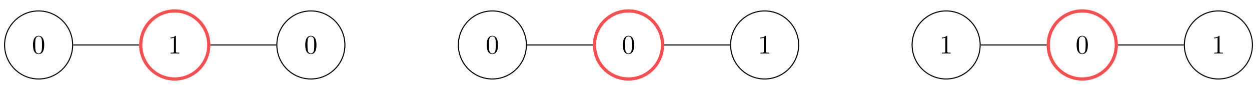


Figure 1: An illustration of the configurations of the MIS problem in cycles. Nodes that are in the MIS output 1 and nodes that are not in the MIS output 0, the center nodes are highlighted in red.

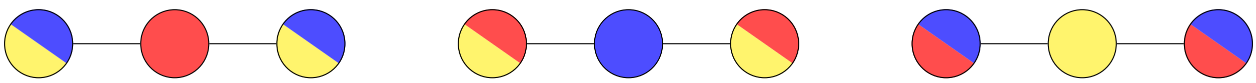


Figure 2: The configurations for the problem of 3 coloring cycles, with colors yellow, blue and red. Two-colored nodes are placeholders, so they represent a node that can either be color 1, or color 2. Each of these 3 configurations represents 4 separate configurations, since the possible combinations of differently colored neighbors have to be turned into explicit versions for the set C.

Task 1: Draw the configurations for 2 coloring cycles and for 2 coloring paths. Note that for paths, we need extra configurations that handle the edgecases of being at the ends of paths.

Sample Solution Task 1

For a proper 2-coloring with colors $\{0, 1\}$, the center node must have a color different from both neighbors. Thus the allowed triples (left, center, right) are exactly:

$$(0, 1, 0) \quad \text{and} \quad (1, 0, 1).$$

In other words, the center is 1 when both neighbors are 0, or the center is 0 when both neighbors are 1. Additionally for the endpoints of paths we need the tuples (endpoint,path-node)

$$(0, 1) \quad \text{and} \quad (1, 0)$$

and (path-node,endpoint)

$$(1, 0) \quad \text{and} \quad (0, 1)$$

For a proper 3-coloring we have triples

$$(1, 0, 1) \quad \text{and} \quad (2, 0, 1) \quad \text{and} \quad (1, 0, 2) \quad \text{and} \quad (2, 0, 2)$$

when considering only those where the center node is 0. Additionally for the endpoints of paths we need the tuples (endpoint,path-node)

$$(0, 1) \quad \text{and} \quad (0, 2)$$

and (path-node,endpoint)

$$(1, 0) \quad \text{and} \quad (2, 1)$$

again only considering the center node to have color 0.

Complexities of LCLs in Paths and Cycles We now want to establish that LCL problems on paths and cycles have only three possible time complexities:

$$O(1), \qquad \Theta(\log^* n), \qquad \Omega(n).$$

In particular, this implies that *no* LCL on cycles can have a complexity such as $\Theta(\log n)$ or any value different from these three.
To simplify the presentation, we make three assumptions. We can prove the same results also without any of these assumptions, but the details become much more technical.

- We restrict attention to cycles, so we do not need to handle the boundary cases at the ends of a path.
- We assume that the checking radius of Π is 1 (i.e., $r = 1$).
- We assume that there is a consistent orientation of the cycle.
- We use the deterministic LOCAL model.

The results we discuss are due to Yi-Jun Chang, Jan Studený, Jukka Suomela, please refer to their work for the full technical details.

2 Gap between $O(\log^* n)$ and $\Omega(n)$

Our goal is to determine whether a given problem is solvable in $O(\log^* n)$ rounds or whether it necessarily requires $\Omega(n)$ rounds.

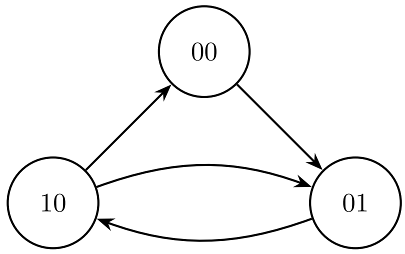


Figure 3: The sequence graph for MIS. Since we now talk about oriented cycles, we need to consider oriented versions of the configuraitons in Figure 1 (the first and third configuration ar symmetric). The first configuraition is $0 \to 1 \to 0$, so this implies an edge from $(0, 1)$ to $(1, 0)$ in the sequence graph. The second configuration is $0 \to 0 \to 1$ this implies an edge from $(0, 0)$ to $(0, 1)$. We can also look at the second configuration from right to left, it then is $1 \to 0 \to 0$ and so it implies an edge from $(1, 0)$ to $(0, 0)$. Lastly the third configuration implies an edge from $(1, 0)$ to $(0, 1)$.

High-level plan. Given an LCL problem Π, we construct a graph that represents all allowed sequences of output labels that may appear along a valid solution on an oriented cycle. We call this graph the *sequence graph* of Π. We then define the notion of a *flexible node* in the sequence graph and prove the following dichotomy:

1. If the sequence graph of Π contains a flexible node, then Π can be solved in $O(\log^* n)$ rounds in oriented cycles.
2. If the sequence graph of Π contains no flexible node, then Π requires $\Omega(n)$ rounds in oriented cycles.

Since one of these two cases always holds, every LCL on paths and cycles is either solvable in $O(\log^* n)$ time or requires $\Omega(n)$ time.

The sequence graph. Let $\Pi = (\Sigma, 1, C)$ be an LCL on cycles. Each configuration in C consists of a triple of labeled nodes (v_l, v_c, v_r), standing for the left, center, and right node, with respective output labels $x_l, x_c, x_r \in \Sigma$.
Think of the edge (v_l, v_c): it is labeled by the tuple (x_l, x_c) in that order. Immediately to its right appears the edge (v_c, v_r), which is labeled (x_c, x_r).
The nodes of the sequence graph correspond exactly to such ordered pairs of labels. For every $(x_1, x_2) \in \Sigma^2$ we create a node in the sequence graph.
Two nodes (x_1, x_2) and (y_1, y_2) in the sequence graph are connected by a directed edge if there exists a configuration in C with

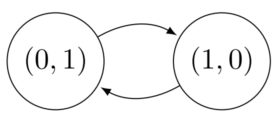
$$x_l = x_1, \qquad x_c = x_2 = y_1, \qquad x_r = y_2.$$

That is, an edge labeled (x_1, x_2) may legally be followed by an edge labeled (y_1, y_2) in some valid local configuration of Π.
This construction captures all globally consistent sequences of edge labels that can appear along a cycle in any valid solution. It is this sequence graph that we will analyze to determine the complexity of Π. For example, we have the sequence graph of MIS in Figure 3

Task 2: Draw the sequence graphs for 3-coloring and 2-coloring.

Sample Solution Task 2

For 2-coloring:



For 3-coloring: Let $\Sigma = \{1, 2, 3\}$. Legal triples are those where the center color differs from each neighbor; neighbors can be equal or different as long as they differ from the center. For example:

$$(1, 2, 1), (1, 2, 3), (3, 2, 1), \ldots$$

Thus edges exist $(a, b) \to (b, c)$ whenever $a \neq b$ and $b \neq c$. In particular:

- For any ordered distinct pair (a, b) there is an edge to (b, a) (the 2-cycle).
- For any triple of distinct colors (a, b, c) there is a 3-cycle $(a, b) \to (b, c) \to (c, a) \to (a, b)$.

Therefore the sequence graph is quite dense on nodes with distinct coordinates; it contains many 2- and 3-cycles. *Remark:* This abundance of cycles will be used to argue flexibility (Task 4).

Walks in the Sequence Graph. A walk in the sequence graph $S_\Pi = (S, E_S)$ is a sequence of output pairs

$$s_1, s_2, s_3, \ldots, s_k \in S,$$

such that for every i, there is a directed edge from s_i to s_{i+1} in S_Π. Note that a walk may revisit the same node multiple times.

We now relate such walks to correct labelings of cycles.
Let $G = (V, E)$ be an oriented cycle, and let $\varphi : V \to \Sigma$ be any correct output labeling for the LCL Π. Consider two consecutive pairs of nodes: u_1, v_1 with v_1 following u_1, and u_2, v_2 with v_2 following u_2. Their output pairs,

$$(x_1, y_1) = (\varphi(u_1), \varphi(v_1)) \quad \text{and} \quad (x_2, y_2) = (\varphi(u_2), \varphi(v_2)),$$

correspond to nodes of the sequence graph S_Π.

Task 3: Assume that the directed distance from u_1 to u_2 along the cycle is k. Show that the labels along the edges between u_1 and u_2 induce a walk in the sequence graph S_Π of length k that starts at (x_1, y_1) and ends at (x_2, y_2).

Sample Solution Task 3

Let G be an oriented cycle and $\varphi : V \to \Sigma$ be a valid labeling for Π. Consider two directed positions $u_1 \to v_1$ and $u_2 \to v_2$ such that v_1 is the k-th successor of u_1 (i.e., directed distance is k). Let the edge labels along the cycle be the ordered pairs $(\varphi(w), \varphi(w^+))$ for each edge $w \to w^+$. The sequence of consecutive edge-label pairs encountered from edge (u_1, v_1) to (u_2, v_2) gives a sequence

$$s_1, s_2, \ldots, s_k$$

where $s_1 = (\varphi(u_1), \varphi(v_1))$ and $s_k = (\varphi(u_2), \varphi(v_2))$, and for each i the transition from $s_i = (x_i, y_i)$ to $s_{i+1} = (x_{i+1}, y_{i+1})$ satisfies $y_i = x_{i+1}$ (they overlap on the center node), and the triple (x_i, y_i, y_{i+1}) is in C. Since (x_i, y_i, y_{i+1}) is in C, by definition of edges in the sequence graph this implies a directed edge $(x_i, y_i) \to (y_i, y_{i+1})$ and so a directed edge $s_i \to s_{i+1}$. Hence these $s_1, \ldots, s_k$ form a directed walk of length k from $(\varphi(u_1), \varphi(v_1))$ to $(\varphi(u_2), \varphi(v_2))$ in S_Π.

Flexible Nodes. We call a node (x_1, x_2) of a sequence graph S *flexible* if there exists a constant $k_0 \in \mathbb{N}$ such that for every $k \geq k_0$ there exists a walk of length k in S that starts and ends at (x_1, x_2).

For example, in the sequence graph of the MIS problem (Figure 3), the node (01) is flexible with $k_0 = 2$. Indeed, the graph contains a 2-cycle between (01) and (10) (namely $(01) \to (10) \to (01)$), and it also contains a 3-cycle $(01) \to (10) \to (00) \to (01)$. Since every integer $k \geq 2$ can be written as a non-negative integer combination

$$k = 2a + 3b,$$

we may obtain a walk of length k by traversing the 2-cycle a times and the 3-cycle b times. Thus (01) is a flexible state.

Task 4: Do the sequence graphs for 3-coloring and 2-coloring contain flexible nodes? If so, identify a flexible node and justify why it is flexible. If not, argue why no node in the sequence graph is flexible.

Sample Solution Task 4

For 2-coloring: The sequence graph is a 2-cycle between $(0,1) \leftrightarrow (1,0)$. From this directed 2-cycle one can make closed walks of any *even* length ≥ 2 that start and end at $(0,1)$ (e.g., go around the 2-cycle m times for length $2m$). But *odd* lengths are impossible (there is no closed walk of odd length because the 2-cycle forces parity). Hence there is no k_0 such that *every* $k \geq k_0$ is realizable. Therefore **no node in the 2-coloring sequence graph is flexible**.

For 3-coloring: Pick any ordered pair (a, b) with $a \neq b$. There is a 2-cycle $(a, b) \to (b, a) \to (a, b)$. Also pick a third color $c \neq a, b$; then the triple (a, b, c) is legal, and the 3-cycle

$$(a, b) \to (b, c) \to (c, a) \to (a, b)$$

exists. Because the sequence graph contains both a 2-cycle and a 3-cycle through (a, b), we can form closed walks whose lengths are any integer representable as $2\alpha + 3\beta$ for nonnegative integers α, β. This is all numbers larger than 2.

We will exploit flexible states as follows.
Consider three nodes u, v, w on the cycle, appearing in this order when traversed clockwise. Assume that the distance from u to v and from v to w is exactly k. Suppose further that there exists a pair (x_1, x_2) such that the sequence graph S_Π contains a walk of length k that starts and ends in (x_1, x_2). We assign the label x_1 to each of the nodes u, v, w and the label x_2 to their immediate successors along the cycle. (So we have (x_1, x_2) at each of the edges after u, v, w)
Because there exists a length-k walk in S_Π from (x_1, x_2) back to (x_1, x_2), we can use the sequence of output pairs along this walk to determine the labels for all nodes between u and v, and likewise for all nodes between v and w. Thus the segments between u, v, and w can be filled in consistently and in parallel by following the labels prescribed by the walk.

2.1 Flexible Nodes Imply $O(\log^* n)$.

We now prove the following claim:

> Let $\Pi = (\Sigma, 1, C)$ be an LCL on cycles whose sequence graph S_Π contains a flexible node. Then Π can be solved deterministically in $O(\log^* n)$ rounds in the LOCAL model.

The key tool will be the notion of *ruling sets*. Informally, a ruling set on a path or cycle is a subset of nodes that split the path/cycle into similarly sized pieces (like in the u, v, w example above).

Exact Ruling Sets One might attempt to define a ruling set as a set $Z \subseteq V$ containing *exactly every k-th node*. However, on paths this is inherently a global problem: determining which node should serve as the "first" node propagates information throughout the graph. In fact, for any fixed k, computing such a set requires $\Omega(n)$ rounds on paths (and on cycles a solution may not even exist if n is not divisible by k).

Task 5: Show that selecting *exactly every k-th node* on a path requires $\Omega(n)$ rounds. (Hint: You may assume k is even and reduce from 2-coloring of paths. Show that, for any constant k, a $o(n)$ algorithm for selecting every k-th node would imply an equally fast deterministic algorithm for 2-coloring a path, which is impossible.)

Sample Solution Task 5

Assume for contradiction that for some fixed,even constant k there is a deterministic LOCAL algorithm A that in $o(n)$ rounds selects exactly every k-th node on any sufficiently long path.
From such an algorithm we will construct an $o(n)$-round algorithm for 2-coloring a path, contradicting known lower bounds (deterministic 2-coloring of a path requires $\Omega(n)$ rounds without unique global symmetry breaking).
We invoke A and obtain a set K, such that K contains every kth node of the path. We have all nodes of K output color 1. Since k is even, there are an odd number of nodes $(k-1)$ between two nodes of k. We can simply have all of these nodes in the middle output a consistent 2-coloring, that respects the choices of the nodes in K in k additional rounds (each node not in K is in some length $k-1$ path envlosed by nodes in K and can therefore infer the correct color).
This takes $o(n)+k$ rounds, which is $o(n)$ since $k \in O(1)$, contradicting the 2-coloring lowerbound.

Because of this inherent global difficulty, we must define ruling sets with some *slack*.

(a, b)-Ruling Sets. A subset $Z \subseteq V$ of the nodes of a cycle is called an (a, b)-*ruling set* (with $1 \leq a \leq b$) if the following hold:

- Any two nodes of Z have distance at least a.
- Every node of the graph, that is not in Z, has a node of Z at distance at most b.

Thus, nodes in Z are neither too close nor too far from each other. For constant a and b, such ruling sets can be computed in $O(\log^* n)$ rounds.

Task 6: Show that for a (k, k)-ruling set Z of a path/cycle, each path of non-ruling set nodes, between two ruling set nodes has length at least $k-1$ and at most $2k$.

Sample Solution Task 6

Let Z be a (k, k)-ruling set on a path or cycle, with $1 \leq k$ integer.
Consider two consecutive nodes of Z along the path/cycle. By the ruling set property,

- Any two nodes of Z have distance at least k. Hence the open segment of non-Z nodes between two consecutive Z nodes has length at least $k-1$ (because distance includes endpoints; e.g., if Z nodes are at positions i and $i+k$, there are $k-1$ nodes strictly between them).
- Every non-Z node has a Z node at distance at most k. Hence in particular every node in the segment between two consecutive Z nodes is within distance k of one of the segment endpoints (which are Z nodes). This implies the segment cannot be longer than $2k$ (otherwise the middle node would be at distance $> k$ from both endpoints).

Task 7: Give a deterministic $O(\log^* n)$-round algorithm for computing a (k, k)-ruling set in cycles for any fixed constant k. (Hint: First compute a $O(\log^* n)$-coloring of the k-th power of the cycle[1] using Linial's coloring algorithm. Then iterate through the colors and select nodes for the ruling set, so that the spacing constraints are satisfied.)

[1]Power graphs should be known from graph theory, but the wikipedia article provides a good reminder.

Sample Solution Task 7

Fix constant k. The algorithm runs in $O(\log^* n)$ rounds and proceeds:

1. Compute a proper vertex coloring of the k-th power of the cycle, C^k, using Linial's $O(\log^* n)$ algorithm. The k-th power C^k has edges between nodes at distance $\leq k$ in the original cycle. Linial's algorithm produces an $O(2k)^2$-coloring in $O(\log^* n)$ rounds, since the degree of C^k is $2k$.

2. Process colors in increasing order. Maintain a set Z initially empty. When processing a color class i, every vertex v of that color checks whether there is already a vertex of Z within distance $k-1$ (in the original cycle). If not, v joins Z. Because all nodes within distance k have different colors in C^k, this choice can be done consistently and locally (nodes that would conflict are prevented by the distance-k constraints ensured by the coloring).

Correctness:

- Any two nodes added to Z are at distance at least k because when a node joins it checked there was no Z node within distance $k-1$.

- Coverage: every node u either is in Z or has some color-processed earlier (or itself) that caused a Z node to appear within distance k; this follows from the greedy processing over the C^k-color classes and the fact that every block of k consecutive vertices contains at least one node whose color is processed early enough to place a Z node nearby.

Each step uses only $O(1)$ local information after the coloring; thus total time is dominated by Linial's $O(\log^* n)$ coloring of C^k, which takes $O(k \cdot \log^* n) = O(\log^* n)$ rounds.

Task 8: Assume Π has a flexible node (x_1, x_2) with flexibility bound $k_0 \in O(1)$, and suppose you have computed a (k_0+1, k_0+1)-ruling set Z on the oriented cycle. Describe how to construct a valid solution to Π in $O(k_0) = O(1)$ additional rounds.

Sample Solution Task 8

Let the nodes of the cycle be partitioned into segments between consecutive Z nodes. By Task 6 each segment length is between k_0 and $2k_0+1$ (constant). For each segment, do the following in parallel (constant-time local procedure):

1. Let the left ruling node and its successor be labeled (assign) x_1, x_2 respectively.

2. Because (x_1, x_2) is flexible, there exists, for the segment length L, a walk of length L in S_Π that starts and ends at (x_1, x_2). Use that walk to assign labels to the vertices along the segment consistently (the walk prescribes the sequence of edge-pairs, hence vertex labels).

3. Do the same independently for all segments in parallel.

All segments can be filled in using only $O(k_0)$ rounds of local information (each segment length is $O(1)$). By Task 3 the output is a valid solution.

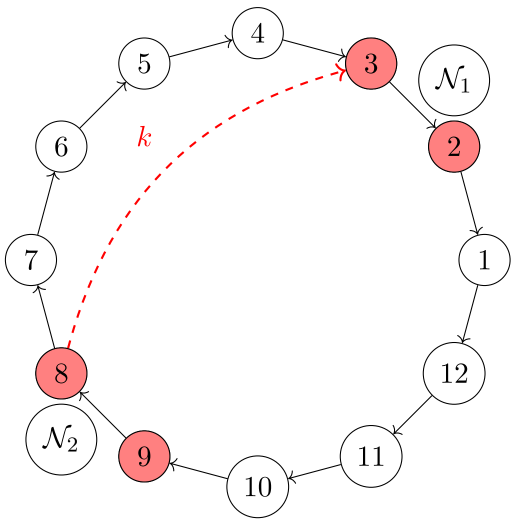
Together Tasks 7 and 8 imply, that any LCL Π with a flexible node in its sequence graph can be solved in $O(\log^* n)$ rounds on oriented cycles.

2.2 No Flexible Node Implies $\Omega(n)$

Next, we show that if the sequence graph S_Π of an LCL Π on oriented cycles contains no flexible node, then Π requires $\Omega(n)$ rounds in the deterministic LOCAL model.

High-Level Idea: If (x_1, x_2) is not a flexible node in the sequence graph, then there are infinitely many integers k for which no walk of length k in the sequence graph starts and ends at (x_1, x_2). For each of these k, this implies that in a solution to Π, there cannot be two pairs of adjacent nodes that output (x_1, x_2) at distance k to each other. (As otherwise by Task 3 there would exist a walk.) To ensure this global constraint, an algorithm must inspect nearly the whole cycle, which forces a lower bound of $\Omega(n)$ rounds.

Illustration. The figure below depicts the idea. Two neighborhoods $\mathcal{N}_1$ (nodes 1–4) and $\mathcal{N}_2$ (nodes 7–10) are shown on the cycle. The algorithm assigns the same label pair (x_1, x_2) to the middle nodes of both neighborhoods (these middle nodes are colored red). The dashed red arrow marks the distance k between the two middle nodes. Since the sequence graph S_Π has no walk of length k that starts and ends at (x_1, x_2), any labeling that places (x_1, x_2) on both middle nodes is invalid, contradicting correctness.



We now make this formal.

Setup. Suppose, for contradiction, that there exists a deterministic LOCAL algorithm A for Π that runs in $o(n)$ rounds on oriented cycles. Since the algorithm has sublinear radius, the output of a pair of consecutive nodes depends only on a neighborhood of radius $o(n)$. Pick n_0 large enough, such that A runs in at most $\frac{n}{C}$ rounds, for all $n \geq n_0$ and some large constant C to be fixed later.

Step 1: Pigeonhole Argument: We need a formal argument, that two such neighborhoods $\mathcal{N}_1$ and $\mathcal{N}_2$ as depicted above must exist.

Task 9: Choose C appropriately (as a function of $|\Sigma|$, the size of the ouptputset), so that there exists an output pair (x_1, x_2) and at least two different $\frac{n}{C}$-hop neighborhoods $\mathcal{N}_1$ and $\mathcal{N}_2$, satisfying the following:

- The set of nodes of the two neighborhoods are assigned disjoint sets of ids.
- In both neighborhoods, there exist two adjacent nodes u, v (v is the successor of u) in the middle, such that A outputs x_1 on u and x_2 on v.

Sample Solution Task 9

There are at most Σ^2 distinct pairs (x_1, x_2), so if we present a T round algorithm with $|\Sigma|^2 + 1$ many different neighborhoods, the algorithm has to output one such pair at least twice.

To that end, we take an oriented path of length $2T + 2$, with middle nodes u, v. If we assign unique ids from $1, \ldots, n$ to these nodes, any T round algorithm A cannot distinguish the local views of u and v from a consistently oriented cycle and hence we can run A on both u and v. We obtain some pair (x_1, x_2).
The number of unique ids used is $2T + 2 < 3T$ (for T sufficiently large). So to be able to repeat this $|\Sigma|^2 + 1$ times using ids $1, \ldots, n$, we choose

$$3T < \frac{n}{|\Sigma|^2 + 1} \to T < \frac{n}{3|\Sigma|^2 + 3}$$

So if we choose $C \geq 3|\Sigma|^2 + 3$, we get that two suitable neighborhoods must exist.

Step 2: No Flexible Node. Take the (x_1, x_2), from step 1, since no node of S_Π is flexible, neither is (x_1, x_2). This means that there exist infinitely many values k such that no walk of length k exists in S_Π starting and ending at (x_1, x_2). We say that such a k is *bad for* (x_1, x_2).

We want to position the two neighborhoods so that their middle nodes are at distance exactly k. However, the radius of each neighborhood depends on the runtime of the algorithm, and the runtime itself depends on the number of nodes in the cycle. Thus, the value of k cannot be chosen arbitrarily: if k is too small, the middle nodes of the two neighborhoods could see each other and the algorithm could not be forced to choose the output (x_1, x_2); if k is too large, we would not have enough nodes in the cycle to place the neighborhoods at distance k. Our construction therefore requires selecting n sufficiently large so that we can choose a value of k that is compatible with both constraints.

Task 10: Use the fact that S_Π does not depend on n and is therefore of finite size. This means *bad* values of k must repeat often.
Then show that starting at some large enough $n_1 \in \mathbb{N}$, for any $n \geq n_1$ and any node (x_1, x_2) of S_Π, there exists a k thats bad for (x_1, x_2) and such that $\frac{n}{6} < k < \frac{n}{3}$.

Sample Solution Task 10

Fix a node $s \in S_\Pi$ and let $L(s)$ be the set of all lengths of closed directed walks that start and end at s. Let $c_1, \ldots, c_m$ be the lengths of all simple directed cycles that can appear on some closed walk at s. Every closed walk at s is a nonnegative integer combination of the c_i, hence

$$L(s) \subseteq \Big\{ \sum_{i=1}^{m} \alpha_i c_i : \alpha_i \in \mathbb{Z}_{\geq 0} \Big\}.$$

Let $g = \gcd(c_1, \ldots, c_m)$. Then every $\ell \in L(s)$ is divisible by g.**If $g = 1$ we would get that s would be flexible.**

Hence $g > 1$, and infinitely many integers are missing. If $g > 1$, every element of $L(s)$ is a multiple of g. Therefore all integers ℓ with $\ell \not\equiv 0 \pmod{g}$ are *not* in $L(s)$, and in particular $L(s)$ misses infinitely many integers.

A missing k in $(n/6, n/3)$ for all large n. Take any sufficiently large cycle length n. The interval

$$I_n = (n/6, n/3)$$

has length $|I_n| = n/6$. If $n \geq 6g$, then $|I_n| \geq g$, so I_n contains at least g consecutive integers. Among any g consecutive integers, at most one is divisible by g. Hence I_n contains at least one integer k with $k \not\equiv 0 \pmod{g}$. Such a k cannot belong to $L(s)$. Therefore, for all $n \geq 6g$, there exists a *missing* integer $k \in (n/6, n/3)$.

Step 3: Construct a Cycle Instance. We pick $n > \max\{n_0, n_1\}$ and use Task 9 to obtain two $\frac{n}{C}$-hop neighborhoods $\mathcal{N}_1$ and $\mathcal{N}_2$, such that A outputs the same output (x_1, x_2) on the middle nodes in both neighborhoods. We now place these in on an oriented cycle, such that the middle two nodes of both neighborhoods are exactly distance k apart (where k is the value from Task 10).

Task 11: Argue that when running A on the above instance, A does not produce a correct solution.

Sample Solution Task 11

Since Σ must contain at leasts two distinct lables (otherwise the problem is trivial) we get $C < \frac{n}{12}$. We call the middle nodes of the neighborhoods u_1, v_1 and u_2, v_2. When placing the two neighborhoods $\mathcal{N}_1$ and $\mathcal{N}_2$ next to each other, the distance between u_1, v_1 and u_2, v_2 is at most $n/6$. So we can add nodes between these two neighborhoods (outside of the n/C hop neighborhood of u_1, v_1, u_2, v_2) to make the distance exactly $k > n/6$. Furthermore, since $k < n/3$ this never takes too mny nodes, so we can arbitrarily add nodes to make the cycle have length exactly n.

Since u_1, v_1, u_2, v_2 still see neighborhoods $\mathcal{N}_1$ and $\mathcal{N}_2$ respectively, we get that A outputs (x_1, x_2) at both of these node pairs. Since these node pairs have distance exactly k in our created instance, and k is bad for (x_1, x_2), A does not produce a valid solution on this instance.

Step 4: Conclude the Lower Bound. This shows that algorithm A is not always correct. Since the argument applies to any sublinear-radius algorithm, we conclude:

Any deterministic LOCAL algorithm for Π requires $\Omega(n)$ rounds.