Abstract

Given a directed graph $G$ and a parameter $k$, the Long Directed Cycle (LDC) problem asks whether $G$ contains a simple cycle on at least $k$ vertices, while the $k$-Path problems asks whether $G$ contains a simple path on exactly $k$ vertices. Given a deterministic (randomized) algorithm for $k$-Path as a black box, which runs in time $t(G,k)$, we prove that LDC can be solved in deterministic time $O^*(\max\{t(G,2k),4k+o(k)\})$ (randomized time $O^*(\max\{t(G,2k),4^k\})$). In particular, we get that LDC can be solved in randomized time $O^*(4^k)$.

Keywords: algorithms, parameterized complexity, long directed cycle, $k$-path

1. Introduction

We study the Long Directed Cycle (LDC) problem. Given a directed graph $G = (V,E)$ and a parameter $k$, it asks whether $G$ contains a simple cycle on at least $k$ vertices. At first glance, this problem seems quite different from the well-known $k$-Path problem, which asks whether $G$ contains a simple path on exactly $k$ vertices; while $k$-Path seeks a solution whose size is exactly $k$, the size of a solution to LDC can be as large as $|V|$. Indeed, in the context of LDC, Fomin et al. noted that “color-coding, and other techniques applicable to $k$-Path do not seem to work here.”

In this paper, we show that an algorithm for $k$-Path can be used as a black box to solve LDC efficiently. More precisely, suppose that we are given a deterministic (randomized) algorithm $ALG$ that uses $t(G,k)$ time and $s(G,k)$ space, and decides whether $G$ contains a simple path on exactly $k$ vertices directed from $v$ to $u$ for some given vertices $v, u \in V$. Then, we prove that LDC can be solved in deterministic time $O^*(\max\{t(G,2k),4k+o(k)\})$ and $O^*(\max\{s(G,k),4k+o(k)\})$ space (if $ALG$ is deterministic), or in randomized time $O^*(\max\{t(G,2k),4^k\})$ and $O^*(s(G,k))$ space (if $ALG$ is randomized). Somewhat surprisingly, we
show that cases that cannot be efficiently handled by calling an algorithm for $k$-Path, can be efficiently handled by merely using a combination of a simple partitioning step and BFS.

The first parameterized algorithm for LDC, due to Gabow and Nie [2], runs in time $O^*(k^{O(k)})$. Then, Fomin et al. [1] gave a deterministic parameterized algorithm for LDC that runs in time $O^*(8^{k+o(k)})$ using exponential-space. Recently, Fomin et al. [3] and the paper [4] modified the algorithm in [1] to run in deterministic time $O^*(6.75k^{+o(k)})$ using exponential-space. It is known that $k$-Path can be solved in randomized time $O^*(2^{k})$ and polynomial-space [5], and deterministic time $O^*(2.59606^{k})$ and exponential-space [6]. Thus, we immediately obtain that LDC can be solved in randomized time $O^*(4^{k})$ and polynomial-space, and deterministic time $O^*(6.73953^{k})$ and exponential-space.

In the following sections, given a graph $G = (V, E)$ and a set $U \subseteq V$, we let $G[U]$ denote the subgraph of $G$ induced by $U$.

2. Finding Large Solutions in Polynomial-Time

We say that an instance $(G, k)$ of LDC seems difficult if $G$ does not contain a directed cycle on $\ell$ vertices for any $\ell \in \{k, k + 1, \ldots, 2k\}$. Roughly speaking, given such an instance, we are forced to determine whether $G$ contains a large solution. This case, as noted in [2] and [1], seems to be the core of difficulty of LDC. We show, somewhat surprisingly, that under certain conditions, this case can be solved in polynomial-time. More precisely, this section proves the correctness of the following lemma.

**Lemma 1.** Let $(G, k)$ be instance of LDC, and let $(L, R)$ be a partition of $V$. Then, there is a deterministic polynomial-time algorithm, PolyAlg, which satisfies the following conditions.

- If $(G, k)$ seems difficult, and $G$ contains a simple cycle $v_1 \rightarrow v_2 \rightarrow \ldots \rightarrow v_t \rightarrow v_1$ such that $t > 2k$, $v_1, v_2, \ldots, v_k \in L$ and $v_{k+1}, v_{k+2}, \ldots, v_{2k} \in R$, PolyAlg accepts.
- If $G$ does not contain a simple cycle on at least $k$ vertices, PolyAlg rejects.

**Proof.** The pseudocode of PolyAlg is given in Algorithm [1]. Clearly, if the algorithm accepts, there exist two distinct vertices $v$ and $u$ such that $G$ contains two simple internally vertex disjoint paths, $P = (V_P, E_P)$ (from $v$ to $u$) and $P' = (V'_{P'}, E'_{P'})$ (from $u$ to $v$), where $|V_P| = k$. In this case, $G$ contains a simple cycle, which consists of these paths, on at least $k$ vertices. Thus, the second item is correct.

Now, we turn to prove the first item. To this end, suppose that the condition of this item is true. Then, we can let $C = v_1 \rightarrow v_2 \rightarrow \ldots \rightarrow v_t \rightarrow v_1$ be a simple cycle in $G$ such that $t > 2k$, $v_1, v_2, \ldots, v_k \in L$ and $v_{k+1}, v_{k+2}, \ldots, v_{2k} \in R$, which minimizes $t$. We need the following observations.

**Observation 1.** The number of vertices on the shortest path from $v_1$ to $v_k$ in $G[L]$ is exactly $k$.
The choice of the proof of the previous observation. Reach a contradiction in the same manner as it is reached in the last paragraph.\[P(|\varepsilon| = i)\]

The existence of \(C\) implies that we can let \(P = (V_P, E_P)\) denote a path from \(v_1\) to \(v_k\) in \(G[L]\) that minimizes \(|V_P|\), and that we can assume that \(|V_P| \leq k\). We further denote \(P = v_1 \rightarrow u_2 \rightarrow \cdots \rightarrow u_{|V_P|}, \) where \(u_1 = v_1\) and \(u_{|V_P|} = v_k\). It remains to show that \(|V_P| = k\). Suppose, by way of contradiction, that \(|V_P| < k\). Let \(v_i\) be the first vertex on the path \(v_{k+1} \rightarrow v_{k+2} \rightarrow \cdots \rightarrow v_t \rightarrow v_1\) that belongs to \(V_P\). Then, we can define a simple cycle \(C'\) in \(G\) as follows.

- If \(i = 1\): \(C' = v_{k+1} \rightarrow v_{k+2} \rightarrow \cdots \rightarrow v_t \rightarrow (v_1 = u_1) \rightarrow u_2 \rightarrow \cdots \rightarrow (u_{|V_P|} = v_k) \rightarrow v_{k+1}\).
- Else: Let \(j\) be the index such that \(v_i = u_j\). Then, \(C' = v_{k+1} \rightarrow v_{k+2} \rightarrow \cdots \rightarrow v_{i-1} \rightarrow (v_i = u_j) \rightarrow u_{j+1} \rightarrow \cdots \rightarrow (u_{|V_P|} = v_k) \rightarrow v_{k+1}\).

Clearly, the number of vertices of \(C'\) is smaller than \(t\). Therefore, by the choice of \(C\) and since \((G, k)\) is a seemingly difficult instance of LDC, we have that \(C'\) is a cycle on less than \(k\) vertices. However, since \(V_P \subseteq L\) and \(v_{k+1}, v_{k+2}, \ldots, v_k \in R\) (where \(R = V \setminus L\)), we have that \(2k < i\). This implies that \(C'\) is a cycle on at least \(k\) vertices, and thus we have reached a contradiction.\(\square\)

**Observation 2.** Let \(P = (V_P, E_P)\) be a simple path from \(v_1\) to \(v_k\) in \(G[L]\) such that \(|V_P| = k\). Then, \(G[V \setminus (V_P \setminus \{v_1, v_k\})]\) contains a path from \(v_k\) to \(v_1\).

**Proof.** Denote \(P = u_1 \rightarrow u_2 \rightarrow \cdots \rightarrow u_k\), where \(u_1 = v_1\) and \(u_k = v_k\). If \(V_P \cap \{v_{k+1}, v_{k+2}, \ldots, v_t\} = \emptyset\), then the claim is clearly true, since then \(v_k \rightarrow v_{k+1} \rightarrow \cdots \rightarrow v_t \rightarrow v_1\) is a path in \(G[V \setminus (V_P \setminus \{v_1, v_k\})]\). Suppose, by way of contradiction, that \(V_P \cap \{v_{k+1}, v_{k+2}, \ldots, v_t\} \neq \emptyset\). Then, we can let \(v_i\) be the first vertex on the path \(v_{k+1} \rightarrow v_{k+2} \rightarrow \cdots \rightarrow v_t\) that belongs to \(V_P\). Let \(j\) be the index such that \(v_i = u_j\). We have that \(C' = v_{k+1} \rightarrow v_{k+2} \rightarrow \cdots \rightarrow v_t \rightarrow (v_i = u_j) \rightarrow u_{j+1} \rightarrow \cdots \rightarrow (u_k = v_k) \rightarrow v_{k+1}\) is a simple cycle in \(G\). Now, we reach a contradiction in the same manner as it is reached in the last paragraph of the proof of the previous observation.\(\square\)
Consider the iteration of Step 1 that corresponds to $v = v_1$ and $u = v_k$. The first observation implies that the condition of Step 2 is false. Next, the second observation implies that the condition of Step 4 is true, and therefore PolyAlg accepts.

\[ \square \]

3. Computing the Sets $L$ and $R$

In this section we observe that the computation of the sets $L$ and $R$ can merely rely on a simple partitioning step. To this end, we need the following definition and known result.

**Definition 1.** Let $F$ be a set of functions $f : \{1, 2, \ldots, n\} \rightarrow \{0, 1\}$. We say that $F$ is an $(n, t)$-universal set if, for every subset $I \subseteq \{1, 2, \ldots, n\}$ of size $t$ and a function $f' : I \rightarrow \{0, 1\}$, there is a function $f \in F$ such that, for all $i \in I$, $f(i) = f'(i)$.

**Lemma 2 (7).** There is a deterministic algorithm that given a pair of integers $(n, t)$, computes in $O^*(2^{t+o(t)})$ time and space an $(n, t)$-universal set $F \subseteq 2^{\{1, 2, \ldots, n\}}$ of size $O^*(2^{t+o(t)})$.

Now, we turn to prove the following simple observations.

**Observation 3.** Let $(G = (V, E), k)$ be an instance of LDC. Then, there is a deterministic algorithm, $\text{DetLRAlg}$, that uses $O^*(4^k + o(k))$ time and space, and returns a set $S = \{(L, R) : L \subseteq V, R = V \setminus L\}$ of size $O^*(4^k + o(k))$ such that the following condition is satisfied.

- For any simple cycle $v_1 \rightarrow v_2 \rightarrow \ldots \rightarrow v_t \rightarrow v_1$ of $G$ such that $t \geq 2k$, there exists $(L, R) \in S$ such that $v_1, v_2, \ldots, v_k \in L$ and $v_{k+1}, v_{k+2}, \ldots, v_{2k} \in R$.

**Proof.** $\text{DetLRAlg}$ arbitrarily orders $V$, and denotes $V = \{v_1, v_2, \ldots, v_{|V|}\}$ accordingly. It obtains an $(|V|, 2k)$-universal set $F$ by relying on Lemma 2. Then, it defines $L_f = \{v_i \in V : f(i) = 0\}$ and $R_f = V \setminus L$ for each $f \in F$, and lets $S = \{(L_f, R_f) : f \in F\}$. The correctness and running time of the algorithm follow immediately from Definition 1 and Lemma 2.

**Observation 4.** Let $(G = (V, E), k)$ be an instance of LDC. Then, there is a randomized algorithm, $\text{RandLRAlg}$, with polynomial time and space complexities, that returns a partition $(L, R)$ of $V$. Moreover, if $\text{RandLRAlg}$ is called $c \cdot 4^k$ times for some $c \geq 1$, and $G$ contains a simple cycle $v_1 \rightarrow v_2 \rightarrow \ldots \rightarrow v_t \rightarrow v_1$ such that $t \geq 2k$, then with probability at least $(1 - e^{-c})$, at least one of the calls returns a pair $(L, R)$ such that $v_1, v_2, \ldots, v_k \in L$ and $v_{k+1}, v_{k+2}, \ldots, v_{2k} \in R$.

**Proof.** $\text{RandLRAlg}$ initializes $L$ to be an empty set, and $R$ to be $V$. For each $v \in V$, with probability $\frac{1}{2}$ it removes $v$ from $R$ and inserts $v$ into $L$. Then, it returns the resulting pair $(L, R)$, which is clearly a partition of $V$.

To prove the correctness of $\text{RandLRAlg}$, suppose that $G$ contains a simple cycle $v_1 \rightarrow v_2 \rightarrow \ldots \rightarrow v_t \rightarrow v_1$ such that $t \geq 2k$. Then, the probability that
v_1, v_2, \ldots, v_k \in L and v_{k+1}, v_{k+2}, \ldots, v_{2k} \in R is \((\frac{1}{2})^{2k} = \frac{1}{4^k}\). Now, if RandLRAlg is called \(c \cdot 4^k\) times, the probability that none of the calls returns a pair \((L, R)\) such that \(v_1, v_2, \ldots, v_k \in L and v_{k+1}, v_{k+2}, \ldots, v_{2k} \in R\) is \((1 - \frac{1}{4^k})^{c \cdot 4^k} \leq e^{-c} \). □

4. Solving the LDC Problem

We are now ready to solve LDC. The input for our algorithm, LDCAlg, consists of an instance \((G, k)\) of LDC, an algorithm ALG for \(k\)-Path, and an argument \(X \in \{\text{det, rand}\}\) that specifies whether ALG is deterministic or randomized. LDCAlg first determines whether \(G\) contains a simple cycle on \(\ell\) vertices, for any \(\ell \in \{k, k+1, \ldots, 2k\}\) by calling ALG. If no such cycle is found, LDCAlg examines enough pairs \((L, R)\), computed using the algorithm in Observation 3 or 4, and accepts iff PolyAlg accepts one of the resulting inputs \((G, k, L, R)\). The pseudocode of LDCAlg is given in Algorithm 2.

**Algorithm 2** LDCAlg\((G = (V, E), k, ALG, X)\)

1: for \(\ell = k, k+1, \ldots, 2k\) do
2: for all \((u, v) \in E\) do
3: Use ALG to determine whether \(G\) contains a simple path on exactly \(\ell\) vertices directed from \(v\) to \(u\). If the answer is positive, accept.
4: end for
5: end for
6: if \(X = \text{det}\) then
7: Let \(S\) be the set returned by DetLRAlg (see Observation 3), ordered arbitrarily. Moreover, let \(x = |S|\), and let PartitionAlg be a procedure that when called at the \(i^{\text{th}}\) time, returns the \(i^{\text{th}}\) pair \((L, R)\) in \(S\).
8: else
9: Let \(x = 10 \cdot 4^k\), and let PartitionAlg be RandLRAlg (see Observation 4).
10: end if
11: for \(i = 1, 2, \ldots, x\) do
12: Call PartitionAlg to obtain a pair \((L, R)\).
13: end for
14: If PolyAlg\((G, k, L, R)\) accepts: Accept.
15: Reject.

**Theorem 1.** Let ALG be an algorithm that uses \(t(G, k)\) time and \(s(G, k)\) space, and decides whether \(G\) contains a simple path on exactly \(k\) vertices directed from \(v\) to \(u\) for some given vertices \(v, u \in V\). Then, LDCAlg solves LDC in deterministic time \(O^*(\max\{t(G, 2k), 4^{k+o(k)}\})\) and \(O^*(\max\{s(G, k), 4^k\})\) space (if ALG is deterministic), or in randomized time \(O^*(\max\{t(G, 2k), 4^k\})\) and \(O^*(s(G, k))\) space (if ALG is randomized).

**Proof.** First, observe that the time and space complexities of LDCAlg directly follow from the pseudocode, Lemma 1 and Observations 3 and 4. Moreover, by
Lemma 1 and the correctness of ALG, if LDCAlg accepts, it is clearly correct (if \( X = \text{rand} \), we mean that LDCAlg accepts with high probability)\(^\text{3}\).

Now, to complete the proof, suppose that \((G, k)\) is a yes-instance. If \(G\) contains a simple cycle on \(\ell\) vertices for some \(\ell \in \{k, k + 1, \ldots, 2k\}\), then one of the calls to ALG accepts, and therefore LDCAlg accepts (if \(X = \text{rand}\), we mean that LDCAlg accepts with high probability). Thus, we can next assume that \((G, k)\) seems difficult, and let \(C = v_1 \rightarrow v_2 \rightarrow \ldots \rightarrow v_t \rightarrow v_1\) denote a simple cycle in \(G\), where \(t > 2k\). By Observations 3 and 4, there is a call to PartitionAlg where it returns a pair \((L, R)\) such that \(v_1, v_2, \ldots, v_k \in L\) and \(v_{k+1}, v_{k+2}, \ldots, v_{2k} \in R\) (in case \(X = \text{rand}\), we mean that there is such a call with high probability). Then, by Lemma 1 PolyAlg accepts, and therefore LDCAlg accepts.

□

References

[1] F. V. Fomin, D. Lokshtanov, S. Saurabh, Efficient computation of representative sets with applications in parameterized and exact algorithms, in: SODA (see also arXiv:1304.4626), 2014, pp. 142–151.

[2] H. N. Gabow, S. Nie, Finding a low directed cycle, ACM Transactions on Algorithms 4 (2008).

[3] F. V. Fomin, D. Lokshtanov, F. Panolan, S. Saurabh, Representative sets of product families, in: ESA, 2014, pp. 443–454.

[4] H. Shachnai, M. Zehavi, Representative families: a unified tradeoff-based approach, in: ESA, 2014, pp. 786–797.

[5] R. Williams, Finding paths of length \(k\) in \(O^*(2^k)\) time, Inf. Process. Lett. 109 (2009) 315–318.

[6] M. Zehavi, Mixing color coding-related techniques, in: ESA, 2015.

[7] M. Naor, J. L. Schulman, A. Srinivasan, Splitters and near-optimal derandomization, in: FOCS, 1995, pp. 182–191.

---

\(^3\)By iteratively removing edges from \(G\), it is easy to see that one can use ALG not only to determine whether \(G\) contains a simple path on exactly \(\ell\) vertices from \(v\) to \(u\), but also to return such a path. In this manner, even if \(X = \text{rand}\), LDCAlg can be modified to accept only if \((G, k)\) is a yes-instance.