On the Complexity of Detecting Constrained Negative Cost Cycles *

Longkun Guo\(^1\), Peng Li\(^2\)

\(^1\)College of Mathematics and Computer Science, Fuzhou University, China
\(^2\)Amazon Web Services, Amazon.com, United States

Abstract. Streaming computing systems, composed by networks of processors connected by one-way first-in-first-out channels, are considered as efficient big data processing platforms. Under the bounded-memory models, streaming computing systems can have deadlocks. To avoid them, the \(k\)-length negative cost cycle (\(k\text{LNCC}\)) problem and the fixed-point \(k\)-length negative cost cycle (\(FPk\text{LNCC}\)) problem arise. For a positive integer \(k\) and a directed graph with a length and a cost on each edge, \(k\text{LNCC}\) is to determine whether there exists a negative cost cycle with at least \(k\) edges, while \(FPk\text{LNCC}\) to determine whether there exists such a cycle containing a given vertex (as the fixed point). This paper first showed that \(FPk\text{LNCC}\) is \(\mathcal{NP}\)-complete in multigraph even for \(k = 3\) by reducing from the 3SAT problem. Then after observing two obstacles of extending the proof to \(k\text{LNCC}\) in simple graphs, we gave another reduction from the 3 Occurrence 3-Satisfiability (3O3SAT) problem, by following a similar line as our first proof but with more complicated details. The hardness result is interesting as \(FP2\text{LNCC}\) and \(2\text{LNCC}\) are both polynomial solvable. This paper also closed the open problem proposed by the previous paper \cite{10} whether \(k\text{LNCC}\) admits polynomial-time algorithms.

Keywords: \(\mathcal{NP}\)-complete, 3 occurrence 3-Satisfiability, 3-Satisfiability, \(k\)-length negative cost cycle, fixed point.

1 Introduction

In the era of big data, streaming computing is attracting considerable interest in the research community. It simplifies parallel computing for big data processing as the involved processors (typically FPUs on GPU or FPGAs) can work collaboratively without explicitly managing resource allocation and processors communication. Unlike other parallel computing platforms, data elements in a streaming computing system are processed as a stream over time, while a series of (pipelined) operations/kernel-functions are applied to each element in the stream. Consequently, deadlocks can happen in ways different from those in ordinary parallel computing systems, where applications can deadlock because all

\* This research work is supported by Australian Research Council Discovery Project DP150104871, and Natural Science Foundation of China #61300025.
processes are waiting for some resources (e.g. locks) and create a waiting cycle. To model a streaming computing system, we use vertices in a digraph to represent computing nodes and edges to represent communication channels. As a result, the deadlock detection problem is then transformed to the cycle detection problem in the digraph, as analyzed in paper \[10\]. It is worth noting that in the model the edges are assigned with integral costs, and only negative cost cycles of length at least three can cause deadlocks, which brings the \( k \)-length negative cost cycle problem that is stated below:

**Definition 1 (\( k \)-Length Negative Cost Cycle problem, \( kLNCC \))**

Given a fixed integer \( k \) and a graph \( G = (V, E) \) in which each edge \( e \in E \) is with a cost \( c(e) \rightarrow \mathbb{R} \) and a length \( l(e) = 1 \), \( kLNCC \) is to determine whether there exists a cycle \( O \) with \( l(O) = \sum_{e \in O} l(e) \geq k \) and the cost \( c(O) < 0 \).

Further, in many cases a node in a streaming computing system is only interested in whether its applications will cause deadlock. Then the \( k \)-length negative-cost fixed-point cycle problem arises, as the definition below:

**Definition 2 (Fixed Point \( k \)-Length Negative Cost Cycle problem, \( FPkLNCC \))**

Given a fixed integer \( k \), a graph \( G = (V, E) \) in which each edge \( e \in E \) is with a cost \( c(e) \rightarrow \mathbb{R} \) and a length \( l(e) = 1 \), and a fixed point \( p \in G \), the \( FPkLNCC \) problem is to determine whether there exists a cycle \( O \) containing \( p \), such that \( l(O) = \sum_{e \in O} l(e) \geq k \) and the cost \( c(O) < 0 \).

Note that if \( FPkLNCC \) admits a polynomial-time algorithm then \( kLNCC \) is also polynomially solvable, since we can use the polynomial-time algorithm for \( FPkLNCC \) as a subroutine to solve \( kLNCC \) in polynomial time. That is, if the \( \mathcal{NP} \)-completeness of \( kLNCC \) is proven then it can be immediately concluded that \( FPkLNCC \) is also \( \mathcal{NP} \)-complete.

Throughout this paper, by *walk* we mean an alternating sequence of vertices and connecting edges; by *trail* we mean a walk that does not pass over the same edge twice; by *path* we mean a walk that does not include any vertex twice; and by *cycle* we mean a path that begins and ends on the same vertex.

1.1 Related works

The \( kLNCC \) problem generalizes two well known problems: the negative cycle detection problem of determining whether there exist negative cycles in a given graph; and the \( k \)-cycle problem (or namely the long directed cycle problem \[2\]) of determining whether there exists a cycle with at least \( k \) edges. The former problem is actually \( kLNCC \) of \( k = 2 \), and can be solved in polynomial time with classical algorithms such as the Bellman-Ford algorithm \[15\]. The latter problem, which is to determine whether a given graph contains a cycle \( O \) with \( l(O) \geq k \), is actually \( kLNCC \) when \( c(e) = -1 \) for every \( e \in E \). It is shown the problem is fixed parameter tractable and admits an algorithm with a time complexity \( k^{O(k)}n^{O(1)} \) \[7\]. The runtime is then improved to \( O(c^kn^{O(1)}) \) for a constant \( c > 0 \) by using representative sets \[4\], and later to \( 6.75^k n^{O(1)} \) independently by \[3\] and \[11\]. Comparing to the two known results above that the
negative cycle detection problem is polynomially solvable and the \( k \)-cycle problem admits FPT-algorithms with respect to \( k \), the \( \mathcal{NP} \)-completeness result of 3LNCC is interesting, as 3LNCC is only the combination of the two problems belonging to \( \mathcal{P} \).

### 1.2 Our results

The main result of this paper is to prove the \( \mathcal{NP} \)-completeness of \( k \)LNCC in a simple directed graph. However, as the proof is constructive and complicated, we will first accomplish a much easier task of proving the \( \mathcal{NP} \)-completeness of the \( FPkLNCC \) problem in multigraphs by giving a reduction from the 3-Satisfiability (3SAT) problem, where a multigraph is a graph that allows multiple edges between two nodes.

**Lemma 3** For any fixed integer \( k \geq 3 \), \( FPkLNCC \) is \( \mathcal{NP} \)-complete in a multigraph.

Then following a similar line of the proof of Lemma 3 but with different and more complicated details, we will prove the \( \mathcal{NP} \)-completeness of \( k \)LNCC (and hence \( FPkLNCC \)) in a simple directed graph.

**Theorem 4** For any fixed integer \( k \geq 3 \), \( k \)LNCC is \( \mathcal{NP} \)-complete in a simple graph which allows length-2 cycles.

Note that, strictly a simple graph does not allow length-2 cycles. So we actually can have the following corollary by replacing length-2 cycles with length-3 cycles in the above theorem.

**Corollary 5** For any fixed integer \( k \geq 4 \), \( k \)LNCC is \( \mathcal{NP} \)-complete in a simple graph.

### 2 The \( \mathcal{NP} \)-completeness of \( FPkLNCC \) in Multigraphs

In this section, we will prove Lemma 3 by reducing from the famous 3-Satisfiability (3SAT) problem, which is known to be \( \mathcal{NP} \)-complete \([8]\). In an instance of 3SAT, we are given \( n \) variables \( \{x_1, \ldots , x_n\} \) and \( m \) clauses \( \{C_1, \ldots , C_m\} \), where \( C_i \) is the OR of at most three literals, and each literal is an occurrence of the variable \( x_j \) or its negation. The 3SAT problem answers whether there is an assignment satisfying all the \( m \) clauses.

For any given instance of 3SAT, the key idea of our reduction is to construct a digraph \( G \), such that \( G \) contains a negative cost trial with length at least 3 iff the instance of 3SAT is satisfiable. The construction is composed by the following three parts. First, for each variable \( x_i \) with \( a_i \) occurrences of \( x_i \) and \( b_i \) occurrences of \( \overline{x}_i \) in the clauses, we construct a lobe which contains two vertices,

---

1 The term lobe was used to denote an unit of the auxiliary graph constructed for an instance of SAT, as in \([9]\) and many others \([13, 6]\).
say $y_i$ and $z_i$, and $a_i + b_i$ edges of cost $-2m$ between the two vertices, i.e. $a_i$ copies of edge $(y_i, z_i)$ and $b_i$ copies of $(z_i, y_i)$ (A lobe is as depicted in a dashed circle in Figure 1). For brevity, we say an edge in the lobes is a lobe-edge. Then, for each clause $C_j$, add two vertices $u_i$ and $v_i$, as well as edge $(v_i, u_{i+1})$, $1 \leq i \leq m-1$, with cost 0 and edge $(v_m, u_1)$ with cost $-1$. Last but not the least, for the relationship between the variables and the clauses, say variable $x_j$ occurrences in clause $C_i$, we add two edges with cost $m$ to connect the lobes and the vertices of clauses:

- If $C_i$ contains $x_j$, then add two edges $(u_i, y_j)$, $(z_j, v_i)$;
- If $C_i$ contains $\overline{x}_j$, then add two edges $(u_i, z_j)$, $(y_j, v_i)$.

An example of the construction for a 3SAT instance is as depicted in Figure 1.

![Figure 1. The construction of $G$ for an 3SAT instance $(x_1 \lor x_3) \land (\overline{x}_1 \lor x_2 \lor x_4) \land (x_1 \lor \overline{x}_2 \lor x_3)$.](image)

Then since $FPkLNCC$ is clearly in $NP$, the correctness of Lemma 3 can be immediately obtained from the following lemma:

**Lemma 6** An instance of 3SAT is satisfiable iff there exists a negative-cost closed trail containing $u_1$ and with length at least 3 in its corresponding auxiliary graph $G$.

**Proof.** Suppose there exists a negative-cost trail $O$, which contains $u_1$ but NO negative cost length-2 cycle. Then let $\tau$ be a true assignment for the 3SAT instance according to $O$: if $O$ goes through $(y_i, z_i)$, then set $\tau(x_i) = true$; Otherwise, set $\tau(x_i) = false$. It remains to show such an assignment will satisfy
all the clauses. Firstly, we will first show the path $P = O \setminus \{v_m, u_1\}$ must go through all vertices of $\{v_i|i = 1, \ldots, m\}$. Since $O$ contains $u_1$, $O$ has to go through edge $(v_m, u_1)$, as the edge is the only one entering $u_1$. Then because $O$ is with negative cost and the cost of edge $(v_m, u_1)$ is $m - 1$, $P$ has to go through at least $m$ edges within the $n$ lobes, as only the edges of lobes has a negative cost $-1$. According to the construction of $G$, between two lobe-edges on $P$, there must exist at least an edge of $\{v_i, u_{i+1}|i = 1, \ldots, m-1\}$, since $v_i$ has only one out-going edge $(v_i, u_{i+1})$ while every edge leaving a lobe must enter a vertex of $\{v_i|i = 1, \ldots, m-1\}$. So $P = O \setminus \{v_m, u_1\}$ has to go through all the $m-1$ edges of $\{v_i, u_{i+1}|i = 1, \ldots, m-1\}$, and hence through all vertices of $\{v_i|i = 1, \ldots, m\}$. Secondly, $v_j$ and $v_{j'}$ are two vertices of $\{v_i|i = 1, \ldots, m\}$, such that $P(v_j, v_{j'}) \cap \{v_i|i = 1, \ldots, m\} = \{v_i, v_{j'}\}$. Again because there must be at least an edge of $\{v_i, u_{i+1}|i = 1, \ldots, m-1\}$ between two lobe-edges, there must be at least a lobe-edge appearing on $P(v_j, v_{j'})$, otherwise there will be at most $m-1$ lobe-edges on $O$. That is, there must be exactly a lobe edge, say $(y_i, z_i)$, on $P(v_j, v_{j'})$. Then according to the construction of graph $G$, $x_i$ appears in $C_j$, and hence $\tau(x_i) = true$ satisfies $C_j$. The case for $(z_i, y_i)$ always appears on $P(v_j, v_{j'})$ is similar. Therefore, the 3SAT instance is feasible as it can be satisfied by $\tau$.

Conversely, suppose the instance of 3SAT is satisfiable, and a true assignment is $\tau: x \rightarrow \{true, false\}$. Then for clause $C_k$, there must exist a literal, say $w_k$, such that $\tau(w_k) = true$. If $w_k$ is an occurrence of $x_i$, then set the corresponding subpath as $P_k = u_k - y_i - z_i - v_k$; otherwise set $P_k = u_k - z_i - y_i - v_k$. Then clearly, $P = \{P_k|k = 1, \ldots, m\} \cup \{(v_{h}, u_{h+1})|h = 1, \ldots, m-1\}$ exactly composes a path from $v_1$ to $v_m$ with a cost of $-m$, as it contains $m$ lobe-edge and other edges of cost 0. So $O = P \cup \{v_m, u_1\}$ is a negative cost cycle of length at least 3. Besides, since $\tau(x_i)$ must be either true or false, there exist no length-2 cycles on $P$. This completes the proof.

However the above proof can not be immediately extended to prove the NP-completeness of $kLNCC$, as there are mainly two obstacles. Firstly, in the above proof, besides cycle $O$ which is forced to go through $u_1$, there can exist other negative cycles with length at least three but without going through $u_1$. Thus, in Lemma containing $u_1$ is mandatory for the negative cost cycle. So to extend the proof to $kLNCC$, we have to find a method to prevent negative cycles excepting those going through $u_1$. Secondly, Lemma is only for multigraphs as some of the lobes are already multigraphs. So we also have to find a method to transform the lobes to simple graphs.

3 Extending to the NP-completeness Proof of $kLNCC$

In this section, to avoid the two obstacles as analyzed in the last section, we will prove Theorem by reducing from the 3 occurrence 3SAT (3O3SAT) problem, which is known NP-complete [12]. Similar to 3SAT, in an instance of 3O3SAT we are also given $m$ clauses $\{C_1, \ldots, C_m\}$ and $n$ variables $\{x_1, \ldots, x_n\}$, and the
task is to determine whether there is an assignment satisfying all the \( m \) clauses. The only difference is, each variable \( x_i \) (including both literal \( x_i \) and \( \overline{x}_i \)) appears at most 3 times in all the \( m \) clauses. Further, to simplify the reduction, we assume that the possible occurrences of a variable \( x \) in an instance of 3O3SAT fall in the following three cases:

- **Case 1:** All occurrences of \( x \) are all \( x \);
- **Case 2:** The occurrences of \( x \) are exactly one positive literal and one negative literal;
- **Case 3:** The occurrences of \( x \) are exactly two positive literals and one negative literal.

The above assumption is without loss of generality. We note that there are still two other cases:

- **Case 4:** All occurrences of \( x \) are negative literals;
- **Case 5:** Exactly two occurrences of negative literals and one positive literal.

However, Case 4 and Case 5 can be respectively reduced to Case 1 and Case 3, by replacing occurrences of \( x \) and \( \overline{x} \) respectively with \( y \) and \( \overline{y} \). Therefore, we need only to consider 3O3SAT instances with variables satisfying Case 1-3.

The key idea of the proof is, for any given instance of 3O3SAT, to construct a graph \( G \), such that there exists a cycle \( O \) with \( c(O) < 0 \) and \( l(O) \geq 3 \) in \( G \) if and only if the instance is satisfiable. An important fact used in the construction is that every variable appears at most 3 times in a 3O3SAT instance. In the following, we will show how to construct \( G \) according to clauses, variables, and the relation between clauses and variables.

1. For each \( C_k \):
   Add to \( G \) two vertices \( u_k \) and \( v_k \), as well as edge \((v_k, u_{k+1})\), \( 1 \leq k \leq m - 1 \) with cost 0, and edge \((v_m, u_1)\) with cost \(-1\).

2. For each variable \( x_i \), construct a lobe according to the occurrences of \( x_i \) and \( \overline{x}_i \) (The construction a lobe is as depicted in a dashed circle as in Figure 2):
   - **Case 1:** All occurrences of \( x_i \) in are positive literal \( x_i \), such as \( x_4 \) in Figure 2.
     For the \( j \)th occurrence of \( x_i \), add a directed edge \((y_j^i, z_j^i)\) and assign cost \(-2m\) to it.
   - **Case 2:** Exactly one occurrence for each of positive literal \( x_i \) and negation \( \overline{x}_i \), such as \( x_2 \) in Figure 2.
     (a) Add two vertex \( z_j^i = y_j^i \) and \( y_j^i = z_j^i \), and connect them with directed edges \((y_j^i, z_j^i)\) and \((y_j^i, z_j^i)\).
     (b) Assign edge \((y_j^i, z_j^i)\) with cost \(-2m\) and \((y_j^i, z_j^i)\) with cost \(\frac{1}{m+1}\).
   - **Case 3:** Exactly 2 occurrences of \( x_i \) and one occurrence \( \overline{x}_i \), such as \( x_1 \) in Figure 2.
     (a) For the two positive literals of \( x_i \), say the \( j_1 \)th and \( j_2 \)th occurrence of \( x_i \), \( j_1 < j_2 \), add four vertices \( y_{j_1}^i \), \( y_{j_2}^i \), \( z_{j_1}^i \), \( z_{j_2}^i \), and two directed edges \((y_{j_1}^i, z_{j_1}^i)\), \((y_{j_2}^i, z_{j_2}^i)\) connecting them with cost \(-2m\);
On the Complexity of Detecting Constrained Negative Cost Cycles

(b) For the negation $\pi_i$, say the $j_{th}$ occurrence, set $z_{i1}^{j3} = y_i^{j1}$ and $y_i^{j3} = z_{i2}^{j3}$, and add three directed edges $(y_i^{j1}, y_i^{j2})$, $(y_i^{j2}, z_{i1}^{j3})$, $(z_{i1}^{j1}, z_{i2}^{j3})$ with costs $\frac{1}{2m+2}$, 0 and $\frac{1}{2m+2}$, respectively.

3. For the relation between the variables and the clauses, say $C_k$ is the clause containing the $j$th occurrence of $x_i$, i.e. $C_k$ is the $j$th clause $x_i$ appears in, add directed edges $(u_k, y_i^j)$ and $(z_i^j, v_k)$. If the occurrence of $x_i$ in $C_k$ is a positive literal, assign the newly added edges with cost $m$; Otherwise, assign them with cost 0. Note that no edges will be added between lobes $u_k$, $v_k$ if $x_i$ does not appear in $C_k$.

An example of the construction of $G$ according to $F = (x_1 \lor x_3)(\overline{x_1} \lor x_2 \lor x_4)(x_1 \lor \overline{x_2} \lor x_4)$ is as depicted in Figure 2.

![Fig. 2. The construction of $G$ for an 3O3SAT instance $(x_1 \lor x_3) \land (\overline{x_1} \lor x_2 \lor x_4) \land (x_1 \lor \overline{x_2} \lor x_4)$.](image)

As $kLNCC$ is apparently in $NP$, it remains only to prove the following lemma to complete the proof of Theorem 4.

Lemma 7 An instance of 3O3SAT is feasible iff the corresponding graph $G$ contains a cycle $O$ with $l(O) \geq 3$ and $c(O) < 0$.

Let $U = \{u_i, v_i | i = 1, \ldots, m\}$ be the set of vertices that correspond to the clauses. We first prove a proposition that if a path from $u_h$ to $v_l$ ($h \neq l$) does not enroute any other $u \in U$, the cost of the path is at least $m$.

Proposition 8 Let $P(u, v)$ be a path from $u$ to $v$. For any path $P(u_h, v_l)$ that satisfies $P(u_h, v_l) \cap U = \{u_h, v_l\}$, if $h \neq l$, then $c(P(u_h, v_l)) \geq m$.

Proof. For every edge $(y, z)$ with cost $-2m$, clearly there exists only one edge $e_1$ entering $y$, and only one edge $e_2$ leaving $z$. Furthermore, $e_1 = (u_h, y)$ and
Suppose $P(u_h, v_1), h \neq l$ does not go through any cost $m$ edges. Let $(u_h, y_i)$, $(z_i^0, v_1) \in P(u_h, v_1)$ be the two edges leaving $u_h$ and entering $v_1$, respectively. Then $y_i^1$ and $z_i^0$ must incident to two edges that corresponds to the negation of two variables. Further, the two variable must be identical, since vertices of two distinct lobes will be separated by $U$, and hence for $P(y_i^1, z_i^0) \subset P(u_h, v_1)$, $P(y_i^1, z_i^0) \cap U \neq \emptyset$. This contradicts with $P(u_h, v_1) \cap U = \{u_h, v_1\}$. 

**Proposition 9** In graph $G \setminus e(v_m, u_1)$, every path $P(u, v)$, $u, v \in U$, has a non-negative cost.

**Proof.** Apparently, in $G \setminus e(v_m, u_1)$, every edge with negative cost is exactly with cost $-2m$. Let $(y_i^1, z_i^0)$ be such an edge with cost $-2m$. From the structure of $G$, there exists exactly one edge entering $y_i^1$, and exactly one leaving $z_i^0$, each of which is with exactly cost $m$. So for every path $P(u, v)$, $u, v \in U$, if it contains edge $(y_i^1, z_i^0)$, then it must also go through both the edge entering $y_i^1$ and the edge leaving $z_i^0$. That is, the three edges must all present or all absent in $P(u, v)$, and contribute a total cost $0$. Therefore, $c(P(u, v)) \geq 0$ must hold. 

Now the proof of Lemma 8 is as below:

**Proof.** Suppose that there exists an assignment $\tau: x \to \{true, false\}$ satisfying all the $m$ clauses. Since $c(v_m, u_1) = -1$, we need only to show there exists a $v_m u_1$-path with cost smaller than $1$ by construction such one path. For a satisfied clause $C_k$, there must exist a literal $w_k$ with $\tau(w_k) = true$. If $w_k$ is the $j$th occurrence of $x_i$, then set the corresponding subpath as $P_k = u_k - y_i^0 - z_i^1 - v_k$. We need only to show $P = \{P_k|k = 1, \ldots, m\} \cup \{(v_h, u_{h+1})|h = 1, \ldots, m - 1\}$ exactly composes a path from $u_1$ to $v_m$ with cost smaller than $1$. For the first, $P$ is a path. Because $\tau$ is an assignment, $\tau(x_i) = true$ and $\tau(x_i) = true$ can not both hold, and hence $P$ contains no length-2 cycle. For the cost, according to the construction, if $\tau(w_k) = true$ then the subpath $P_k$ is with cost exactly equal to $0$; otherwise, i.e. $\tau(w_k) = \tau(x_i) = true$, the subpath $P_k$ is with cost exactly equal to $\frac{1}{m+1}$. Meanwhile, $c(e(v_h, u_{h+1})) = 0$ for each $h$. Therefore, the total cost $c(P) \leq \frac{m+1}{m+1} < 1$, where the maximum is attained when all clauses are all satisfied by negative of the variables.

Conversely, assume that there exists a negative-cost cycle $O$, which contains NO negative cost length-2 cycle. According to Proposition 8, $e(v_m, u_1)$ must appears on $O$, so that $c(O) < 0$ can hold. Let $P = O \setminus e(v_m, u_1)$ and $\tau$ be a true assignment according to $P$: if $P$ goes through literal $x_i$, set $\tau(x_i) = false$ and set $\tau(x_i) = true$ otherwise. It remains to show such the assignment according to $P$ satisfies all the clauses. To do this, we shall firstly show $P$ will go through all the vertices of $U$ in the order $u_1 \prec v_1 \prec \cdots \prec u_i \prec v_1 \prec \cdots \prec u_m \prec v_m$; and secondly show that $P(u_h, v_1)$ has to go through exactly a subpath corresponding
to a literal, say \( w \), for which if \( \tau(w) = \text{true} \), then \( C_h \) is satisfied. Then from the fact that \( P \) contains no negative cost length-2 cycle, \( \tau \) is a feasible assignment satisfying all the clauses.

For the first, according to Proposition \( \mathbb{S} \) if \( P(u_i, v_j) \cap U = \{u_i, v_j\} \), then \( j = i \) (i.e. \( v_j = v_i \)) must hold. Since otherwise, according to Proposition \( \mathbb{S} \) \( c(P(u_i, v_j)) \geq m \) must hold; while according to Proposition \( \mathbb{S} \) the other parts of \( P \) is with \( c(P(u_1, u_i)) \geq 0 \) and \( c(P(v_j, v_m)) \geq 0 \). That is, \( c(P) \geq m \). On the other hand, since \( c(e(v_m, u_1)) = -1 \) and \( c(O) < 0 \), we have \( c(P) < 1 \), a contradiction. Furthermore, since there exists only one edge leaving \( v_i \), i.e. \( (v_i, u_{i+1}) \), \( P \) must go through every edge \( e(v_i, u_{i+1}) \) incrementally on \( i \), i.e. in the order of \( u_1 \prec v_1 \prec \cdots \prec u_i \prec v_i \prec \cdots \prec u_m \prec v_m \). For the second, according to the structure of \( G \) and \( c(P) < 1 \), \( c(P(u_h, v_h)) \leq \frac{1}{m+1} \) must hold. Then \( c(P(u_h, v_h)) \) has to go through exactly a subpath corresponding to a literal. \( \Box \)

According to the construction of \( G \), the length-2 cycles only exist in the lobes. Apparently, all the length-2 cycles can be transformed to length-3 cycles by replacing every edge \( (y_j, z_j) \) with two edges \( (y_j, w_j) \) and \( (w_j, z_j) \) of the same cost. Therefore, we immediately have the correctness of Corollary \( \mathbb{A} \).

4 Conclusion

In this paper, we have shown the \( \mathsf{NP} \)-completeness for both the \( k \)-length negative cost cycle (\( k\text{-LNCC} \)) problem and the fixed-point \( k \)-length negative cost cycle (\( \text{FP}_k\text{-LNCC} \)) problem, which have wide applications in parallel computing, particularly in deadlock avoidance for streaming computing systems. In future, we will investigate approximation algorithms for these two problems.

References

1. Richard Bellman. On a routing problem. Technical report, DTIC Document, 1956.
2. Marek Cygan, Fedor V Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. Parameterized Algorithms, volume 4. Springer, 2015.
3. Fedor V Fomin, Daniel Lokshtanov, Fahad Panolan, and Saket Saurabh. Representative sets of product families. In Algorithms-ESA 2014, pages 443-454. Springer, 2014.
4. Fedor V Fomin, Daniel Lokshtanov, and Saket Saurabh. Efficient computation of representative sets with applications in parameterized and exact algorithms. In Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, pages 142–151. SIAM, 2014.
5. Lester Randolph Ford. Network flow theory. 1956.
6. S. Fortune, J. Hopcroft, and J. Wyllie. The directed subgraph homeomorphism problem* 1. Theoretical Computer Science, 10(2):111–121, 1980.
7. Harold N Gabow and Shuxin Nie. Finding a long directed cycle. ACM Transactions on Algorithms (TALG), 4(1):7, 2008.
8. Michael R Gary and David S Johnson. Computers and intractability: A guide to the theory of np-completeness, 1979.
9. Longkun Guo and Hong Shen. On finding min-min disjoint paths. *Algorithmica*, 66(3):641–653, 2013.
10. Peng Li, Kunal Agrawal, Jeremy Buhler, and Roger D. Chamberlain. Deadlock avoidance for streaming computations with filtering. In *Proc. 22nd ACM Symp. on Parallelism in Algorithms and Architectures*, pages 243–252, 2010.
11. Hadas Shachnai and Meirav Zehavi. Faster computation of representative families for uniform matroids with applications. *CoRR*, abs/1402.3547, 2014.
12. Craig A Tovey. A simplified np-complete satisfiability problem. *Discrete Applied Mathematics*, 8(1):85–89, 1984.
13. D. Xu, Y. Chen, Y. Xiong, C. Qiao, and X. He. On the complexity of and algorithms for finding the shortest path with a disjoint counterpart. *IEEE/ACM Transactions on Networking*, 14(1):147–158, 2006.