Relations between automata and the simple $k$-path problem

Ran Ben-Basat     Ariel Gabizon\textsuperscript{*}
Technion, Haifa

January 28, 2014

Abstract

Let $G$ be a directed graph on $n$ vertices. A $k$-path in $G$ is a path $p = v_1 \rightarrow \ldots \rightarrow v_k$ in $G$. Given an integer $k \leq n$, the SIMPLE k-PATH problem asks whether there exists a simple $k$-path in $G$. In case $G$ is weighted, the MIN-WT SIMPLE k-PATH problem asks for a simple $k$-path in $G$ of minimal weight. The fastest currently known deterministic algorithm for MIN-WT SIMPLE k-PATH by Fomin, Lokshtanov and Saurabh \cite{5} runs in time $O(2^{0.851k} \cdot n^{O(1)} \cdot \log W)$ for graphs with integer weights in the range $[-W, W]$. This is also the best currently known deterministic algorithm for SIMPLE k-PATH - where the running time is the same without the log $W$ factor.

We define $L_k(n) \subseteq [n]^k$ to be the set of words of length $k$ whose symbols are all distinct. We show that an explicit construction of a non-deterministic automaton (NFA) of size $f(k) \cdot n^{O(1)}$ for $L_k(n)$ implies an algorithm of running time $O(f(k) \cdot n^{O(1)} \cdot \log W)$ for MIN-WT SIMPLE k-PATH when the weights are non-negative or the constructed NFA is acyclic as a directed graph. We show that the algorithm of Kneis et al. \cite{9} and its derandomization by Chen et al. \cite{8} for SIMPLE k-PATH can be used to construct an acyclic NFA for $L_k(n)$ of size $O^*(4^{k+o(k)})$.

We show, on the other hand, that any NFA for $L_k(n)$ must have size at least $2^k$. We thus propose closing this gap and determining the smallest NFA for $L_k(n)$ as an interesting open problem that might lead to faster algorithms for MIN-WT SIMPLE k-PATH.

We use a relation between SIMPLE k-PATH and non-deterministic xor automata (NXA) to give another direction for a deterministic algorithm with running time $O^*(2^k)$ for SIMPLE k-PATH.

1 Introduction

Let us recall the classic Travelling Salesman Problem (TSP): Given a complete undirected weighted graph $G$ on $n$ vertices we wish to find a cycle of minimal weight passing through all vertices. A parameterized version of this problem is sometimes called $k$-TSP (cf. \cite{11}). Here the salesman wants to visit only $k$ out of the $n$ cities (he does not insist on which $k$) while minimizing the total travel time. Note that in general the optimal route may not be simple. It will be convenient to formally define a more general problem, where the desired end point and starting point are given as part of the input and the graph can be directed. What we get is a problem referred to in \cite{3} as the k-STROLL problem.

\textsuperscript{*}The research leading to these results has received funding from the European Community’s Seventh Framework Programme (FP7/2007-2013) under grant agreement number 257575.

\textsuperscript{1}There seem to be inconsistencies in the literature on whether $k$-TSP insists on a cycle, or a fixed starting point.
**k-STROLL**

- **Input:** Directed graph \( G = (V, E) \), vertices \( s, t \in V \), weight function \( w : E \to \mathbb{R} \).
- **Parameter:** \( k \in \mathbb{N} \).
- **Problem:** Find a minimal weight path from \( s \) to \( t \) that visits at least \( k \) distinct vertices (counting \( s \) and \( t \)).

The k-TOUR problem [3] is a special case of k-STROLL where \( s = t \).

A related problem that has received much attention is that of determining whether there exists a *simple* \( k \)-path in a graph, and if so returning such a path of minimal weight. Here we define a \( k \)-path in a graph to be a path of the form \( v_1 \to \ldots \to v_k \) (i.e., the number of *vertices* in the path is \( k \)). Let us define the unweighted and weighted versions of this problem.

**SIMPLE \( k \)-PATH**

- **Input:** Directed graph \( G = (V, E) \).
- **Parameter:** \( k \in \mathbb{N} \).
- **Problem:** Determine if there exists a simple \( k \)-path in \( G \), and if so return such a path.

**MIN-WT SIMPLE \( k \)-PATH**

- **Input:** Directed graph \( G = (V, E) \), weight function \( w : E \to \mathbb{R} \).
- **Parameter:** \( k \in \mathbb{N} \).
- **Problem:** Determine if there exists a simple \( k \)-path in \( G \), and if so return such a path of minimal weight.

For vertices \( s, t \in V \), an \((s, t)\)-\( k \)-path is a \( k \)-path beginning in \( s \) and ending in \( t \). Let us now also define MIN-WT SIMPLE \((s, t)\)-\( k \)-PATH to be the version of MIN-WT SIMPLE \( k \)-PATH where we give as additional input vertices \( s, t \in V \) and ask for a simple \((s, t)\)-\( k \)-path of minimal weight. Though an optimal solution for k-STROLL is not necessarily a *simple* path, the problem is easily reducible to MIN-WT SIMPLE \((s, t)\)-\( k \)-PATH: Given \( G \) compute the complete graph \( G' \) on the same set of vertices, where the weight of the directed edge \((u, v)\) is the weight of the minimal weight path between \( u \) to \( v \) in \( G \). A simple \((s, t)\)-\( k \)-path \( p' \) in \( G' \) of minimal weight corresponds to an \((s, t)\)-\( k \)-path \( p \) in \( G \) passing through \( k \) distinct vertices of minimal weight: Replace an edge \((u, v)\) in \( p' \) by the shortest path from \( u \) to \( v \) in \( G \). This connection gives more motivation for solving MIN-WT SIMPLE \( k \)-PATH (it seems that all known algorithms for MIN-WT SIMPLE \( k \)-PATH can be adapted to solve MIN-WT SIMPLE \((s, t)\)-\( k \)-PATH with the same running time).

The purpose of this paper is to propose a direction for obtaining faster deterministic algorithms for MIN-WT SIMPLE \( k \)-PATH via a connection to automata theory.

### 1.1 Previous results on MIN-WT SIMPLE \( k \)-PATH and our results

Alon, Yuster and Zwick [11] gave the first deterministic algorithm for MIN-WT SIMPLE \( k \)-PATH running in time \( O(2^{O(k)} \cdot n^{O(1)} \cdot \log W) \), where we assume the weights of the graph are integers in the range \([-W, W]\).

The current state of the art is by Fomin, Lokshtanov and Saurabh [5] giving a deterministic algorithm running in time \( O(2.851^k \cdot n^{O(1)} \cdot \log W) \).
Definition 1.1 (The language $L_k(n)$). Fix positive integers $k \leq n$. We define $L_k(n) \subseteq [n]^k$ to be the set of words $w_1 \cdots w_k \in [n]^k$ such that $w_1, \ldots, w_k$ are all distinct.

Our main result is to show that a non-deterministic finite automaton (NFA) for the language $L_k(n)$ implies an algorithm for MIN-WT SIMPLE $k$-PATH whose running time is close to the size of the NFA. In fact, Theorem 3.4 in Section 3 gives a general connection between constructing compact NFAs and finding minimal-weight paths satisfying a certain constraint (in our case the constraint is being simple of length $k$).

We state the result formally for MIN-WT SIMPLE $(s, t)$-$k$-PATH. Note that MIN-WT SIMPLE $k$-PATH can be easily reduced to MIN-WT SIMPLE $(s, t)$-$k$-PATH by adding a start vertex $s$ that has outgoing edges to all vertices, and a target vertex $t$ that has ingoing edges from all vertices.

The following theorem uses notation regarding NFAs from Definition 2.1. We note in particular that by the size of an NFA we mean the total number of states and transitions it contains.

Theorem 1.2. Fix integers $k \leq n$. Suppose we can construct an NFA $M$ of size $s$ with $L(M) = L_k(n)$ in time $O(s)$. Then we can solve MIN-WT SIMPLE $(s, t)$-$k$-PATH on graphs with $n$ vertices and non-negative integer weights of size at most $W$ in time $O(s \cdot \log s \cdot n^2 \cdot \log W)$.

In case $M$ is a directed acyclic graph we can solve MIN-WT SIMPLE $(s, t)$-$k$-PATH on graphs with $n$ vertices and integer weights in the range $[-W, W]$ in time $O(s \cdot n^2 \cdot \log W)$.

In Section 5 we show that the algorithms of Kneis et al. [9] and Chen et al. [8] for SIMPLE k-PATH can be used to construct an acyclic NFA for $L_k(n)$ of size $4^k \cdot k^{O(\log^2 k)}$ in time $O(4^k \cdot k^{O(\log^2 k)})$. In Section 4 we show that any NFA for $L_k(n)$ must have at least $2^k$ states. We thus find closing this gap to be an interesting problem that could lead to a faster deterministic algorithm for MIN-WT SIMPLE $k$-PATH.

A non-deterministic XOR automata (NXA) is an NFA where the acceptance condition is that a word has an odd number of accepting paths, rather than at least one. In Section 6 we show that a small set of NXAs of size $O^*(2^k)$ can be constructed such that the union of their languages is $L_k(n)$. This construction is in fact related to a randomized algorithm for SIMPLE $k$-PATH of Abasi and Bshouty [2]. We use this to give an $O^*(8^k)$ randomized algorithm for SIMPLE $k$-PATH. The algorithm could be derandomized and its running time improved potentially to $O^*(2^k)$ if a certain set of matrices could be explicitly constructed and a faster algorithm for checking the emptiness of an NXA were devised. See Section 6 for details.

2 Preliminaries

We formally define non-deterministic automata. It will be convenient to allow the transitions of the automaton to be weighted.

Definition 2.1 (NFA). A non-deterministic finite automaton (NFA) $M$ over alphabet $\Sigma$ is a labeled directed graph $M = \langle Q, \Delta, q_0, F \rangle$ where

- $Q$ is the set of vertices. We refer to the elements of $Q$ as ‘states’.
- $\Delta$ is the set of edges. We refer to elements of $\Delta$ as ‘transitions’ and suggestively use the notation $(u \to v)$ rather than $(u, v)$.
- Each transition $e \in \Delta$ is labeled with an element of $\Sigma$.
- $q_0$ is an element of $Q$ which is the ‘start state’ of $M$.
- $F \subseteq Q$ is the set of ‘accepting states’.


At times $M$ will be a weighted graph. That is, we will also have a weight function $w : \Delta \rightarrow \mathbb{R}$. For a word $w = w_1 \cdots w_t \in [n]^t$, we define $M(w) \subseteq V$ to be the ‘subset of states reach by $w$’ in the usual way for NFAs. One subtlety: If while reading a word we reach a state where we cannot progress by reading the next symbol, this run is rejected and the state we are at is not added to $M(w)$. We define the language of $M$, denoted $L(M)$ by

$$L(M) \triangleq \{ w \in [n]^* | M(w) \cap F \neq \emptyset \}.$$  

It will be convenient to define the size of $M$, denoted size($M$), as the sum of the number of states and transitions in $M$. That is, size($M$) $\triangleq$ $|Q| + |\Delta|$.

Finally, we say $M$ is acyclic if it is acyclic as a directed graph.

**Definition 2.2 (Intersection NFA).** Given NFAs $M_1 =< Q_1, \Delta_1, q_0^1, F_1 >$ and $M_2 =< Q_2, \Delta_2, q_0^2, F_2 >$ over the same alphabet $\Sigma$ we define the intersection NFA

$$M_1 \cap M_2 \triangleq < Q_1 \times Q_2, \Delta, < q_0^1, q_0^2 >, F_1 \times F_2 >$$

over $\Sigma$, where the set of transitions $\Delta$ is defined as follows. For every pair of transitions $(u_1 \rightarrow v_1) \in \Delta_1$ and $(u_2 \rightarrow v_2) \in \Delta_2$ that are both labeled by the same element $a \in \Sigma$, we have a transition $(< u_1, u_2 > \rightarrow < v_1, v_2 >) \in \Delta$ labeled $a$.

It is known that

**Fact 2.3.** $L(M_1 \cap M_2) = L(M_1) \cap L(M_2)$.

### 3 Finding automata-constrained shortest paths

The purpose of this section is to establish a general connection between algorithms for finding minimal weight paths satisfying a certain constraint and NFAs representing the constraint.

The following definition and straightforward lemma formally convert a graph into an automaton accepting the paths of the graph.

**Definition 3.1 (The path automaton).** Let $G = < V, E >$ be a directed graph. Fix $s, t \in V$.

The NFA

$$M(G,s,t) \triangleq < Q = V \cup \{ q_0 \}, \Delta = E \cup \{ (q_0, s) \}, q_0, F = \{ t \} >$$

with alphabet $\Sigma = V$ is defined with the following labeling of transitions. The transition $(q_0 \rightarrow s)$ will be labeled $s$. For each $(u,v) \in E$ the transition $(u \rightarrow v)$ is labeled with the source vertex $u \in V$ of the edge.

**Lemma 3.2.** Let $G = < V, E >$ be a directed graph. Fix $s, t \in V$. Then $L(M(G,s,t))$ is precisely the set of words $p = s \cdot v_1 \cdots v_m \cdot t$ such that $s \rightarrow v_1 \rightarrow \cdots \rightarrow v_m \rightarrow t$ is a path from $s$ to $t$ in $G$.

**Definition 3.3 (Paths accepted by an NFA).** Fix an NFA $M$ with alphabet $\Sigma$, and a directed graph $G = < \Sigma, E >$. Let $p = v_1 \rightarrow v_2 \rightarrow \ldots \rightarrow v_t$ be a (directed) path in $G$. Identify $p$ with the word $v_1 \cdots v_t \in \Sigma^\ast$. We say the path $p$ is accepted by $M$ if $p \in L(M)$. Or in words, running the NFA $M$ with the word $p$ can end in an accepting state.

The following theorem states that if we have an NFA of a certain size capturing a certain constraint on a path, we have an algorithm for finding the shortest path satisfying the constraint whose running time is similar to the size of the NFA.
Theorem 3.4. Fix any NFA $M$ with alphabet $\Sigma = [n]$. There is an algorithm that, given as input a directed weighted graph $G = ([n], E, w)$ with integer weights and vertices $s, t \in [n]$, returns an $(s, t)$-path in $G$ that is accepted by $M$ of minimal weight. The running time of the algorithm is at most $O(\text{size}^2(M) \cdot n^3)$. The running time can be improved to

- $O(\text{size}(M) \cdot \log(\text{size}(M)) \cdot n^2)$ when $G$ only contains non-negative weights.
- $O(\text{size}(M) \cdot n^2)$ when $M$ is acyclic.

All running times assume $O(1)$ arithmetic operations on weights and without this assumption require an additional $\log W$ factor when the weights are in the range $[−W, W]$.

Proof. First note that we can convert $M$ to an NFA with one accepting state while at most doubling its size. Let us assume from now on that $M$ indeed has a unique accepting state. Let $M(G, s, t)$ be the NFA from Definition 2.2. We construct the intersection NFA $N \triangleq M \cap M(G, s, t)$ as in Definition 2.2. Now we add weights to the transitions according to $G$. More precisely, transitions $(q_1 \rightarrow q_2) \in \Delta$ with $(u, v) \in E$ will be given weight $w(u, v)$. All other transitions (simply ones where the second coordinate shifts from the start state of $M(G, s, t)$ to $s$) will be given weight 0.

Note that when looking at $N$ as a weighted directed graph, the paths from its start to accept state exactly correspond to the $(s, t)$-paths in $G$ accepted by $M$. (It is possible that a certain $(s, t)$-path in $G$ corresponds to many accepting paths in $N$). Now note that the weight of any accepting path in $N$ of a word $s \cdot v_1 \cdots v_m \cdot t$ is the same as the weight of the path $s \rightarrow v_1 \rightarrow \ldots \rightarrow v_m \rightarrow t$ in $G$. Thus, running a shortest path algorithm on $N$ from the start to accept state will give us an $(s, t)$-path in $G$ that is accepted by $M$ and is of minimal weight among the $(s, t)$-paths in $G$ accepted by $M$. Note that $N$ has at most $\text{size}(M) \cdot (n + 1)$ vertices and at most $\text{size}(M) \cdot n^2$ edges. Running the Bellman-Ford algorithm would give us time $O(|V| \cdot |E|) = O(\text{size}^2(M) \cdot n^3)$. In case $G$ has non-negative weights we can use Fredman and Tarjan’s implementation of Dijkstra’s algorithm [6] to get time $O(|E| + |V| \cdot \log |V|) = O(\text{size}(M) \cdot \log(\text{size}(M)) \cdot n^2)$. In case $M$ is acyclic so is $N$ and we can use topological sort to get time $O(\text{size}(M) \cdot n^2)$. \qed

Theorem 3.2 now follows from Theorem 3.4 by considering NFAs whose language is $L_k(n)$.

4 A Lower bound for the NFA size of $L_k(n)$

The following theorem of Gliaster and Shallit [7] gives a method to lower bound the NFA size of a language.

Theorem 4.1. Fix a language $L \subseteq [n]^*$. Suppose we have elements $x_1, \ldots, x_t, y_1, \ldots, y_t \in [n]^*$ such that

- For all $i \in [t]$, $x_i \cdot y_i \in L$.
- For all $i \neq j \in [t]$, $x_i \cdot y_j \notin L$.

Then any NFA for $L$ has at least $t$ states.

Theorem 4.2. Fix any integers $k \leq n$. Then any NFA for $L_k(n)$ has at least $2^k$ states.

Proof. For every subset $S = \{i_1, \ldots, i_d\} \subseteq [k]$, let $x_S \in [n]^k$ be the word $x_S = i_1 \cdots i_d$. For every $S \subseteq [k]$ define $y_S = x_S$. It is clear that for every $S \subseteq [k]$, $x_S \cdot y_S \in L$. And for every $S \neq T \subseteq [k]$ $x_S \cdot y_T \notin L$. Now the claim follows from Theorem 4.1. \qed
5 NFA construction for $L_k(n)$

In this section we give an explicit construction of an NFA for the language $L_k(n)$ of size $O^*(4^{k+o(k)})$. The NFA construction and analysis closely correspond to the algorithm for SIMPLE k-PATH of [9] and its derandomization using universal sets by [8]. For this purpose we now define universal sets.

**Definition 5.1 ((n, k)-universal set).** A set of strings $U \subseteq \{0, 1\}^n$ is an $(n,k)$-universal set if for every $S \subseteq [n]$ of size $k$, and every $a \in \{0, 1\}^k$ we have $x \in T$ such that $x|_S = a$. Equivalently, an $(n,k)$-universal set is a set $T$ of subsets of $[n]$ such that for every $S \subseteq [n]$ of size $k$ and every $S' \subseteq S$ we have $T \in U$ such that $T \cap S = S'$.

Naor, Schulman and Srinivasan [10] gave an almost optimal construction of universal sets.

**Claim 5.2.** [10] Fix integers $k \leq n$. There is a deterministic algorithm of running time $O(2^k \cdot k^{O(\log k)} \cdot \log n)$ that constructs an $(n,k)$-universal set of size $2^k \cdot k^{O(\log k)} \cdot \log n$.

We now state the main result of this section.

**Theorem 5.3.** Fix integers $k \leq n$. An acyclic NFA $M$ of size $O^*(4^k \cdot k^{O(\log^2 k)})$ for $L_k(n)$ can be constructed in time $O^*(4^k \cdot k^{O(\log^2 k)})$.

Before proving the theorem we state a technical claim that will be used in the analysis.

**Claim 5.4.** For a positive integer $k$ look at the sum

$$s(k) = k + \lceil k/2 \rceil + \lceil k/2 \rceil/2 + \ldots + 1.$$ 

Then

- $s(k) \leq 2k + 2 \cdot \log k$  
- The number of summands in $s(k)$ is at most $\log k + 1$.

We proceed with the proof of Theorem 5.3.

**Proof.** The following definition will be convenient for the proof. For a subset $S \subseteq [n]$ we define the language $L_k(n,S) \triangleq L_k(n) \cap S^k$. In words, $L_k(n,S)$ is simply the set of words in $w \in [n]^k$ whose symbols are all distinct, and are also all in $S$. Fix any positive integer $n$. For every $1 \leq k \leq n$ and $S \subseteq [n]$ we construct an NFA $M_{k,S}$ for $L_k(n,S)$ by induction on $k$ as follows.

For $k = 1$, given $w \in [n]^k$ the $M_{k,S}$ will simply check if $w_1 \in S$ and if $|w| = 1$. Such $M_{k,S}$ of size $3 \cdot n$ can be constructed. Now assume we have a construction of an NFA $M_{k',S}$ for every $1 \leq k' < k$ and $S \subseteq [n]$. Before constructing $M_{k,S}$, let us construct as a component an NFA for a simpler language. Fix disjoint subsets $S_1, S_2 \subseteq [n]$. We will define an NFA $M_{k,S_1,S_2}$ that accepts exactly the words $w \in L_k(n)$ whose first $\lceil k/2 \rceil$ symbols are in $S_1$, and last $\lceil k/2 \rceil$ symbols are in $S_2$. $M_{k,S_1,S_2}$ can be constructed as follows. $M_{k,S_1,S_2}$ will consist of a copy of a $M_{\lceil k/2 \rceil,S_1}$ that reads the first $\lceil k/2 \rceil$ symbols of $w$, followed by a copy of $M_{\lceil k/2 \rceil,S_2}$ that reads the last $\lceil k/2 \rceil$ symbols of $w$.

Now, given $S \subseteq [n]$ we construct $M_{k,S}$ as follows. Fix an $(n,k)$-universal set $U$ of size $|U| = 2^k \cdot k^{O(\log k)} \cdot \log n$ obtained from Theorem 5.2. For every set $T \in U$ we put an $\varepsilon$-transition from the start state of $M_{k,S}$ to a copy of the NFA $M_{k,S \cap T,S \cap T}$. Thus, $M_{k,S}$ accepts a word $w$ if and only if one of the automata $\{M_{k,S \cap T,S \cap T}\}_{T \in U}$ accepts $w$. Let us show that indeed $L(M_{k,S}) = L_k(n,S)$. Note that for any disjoint subsets $S_1, S_2 \subseteq S$, $M_{k,S_1,S_2}$ accepts a subset of $L_k(n,S)$. Hence, it is clear that $M_{k,S}$ does not accept any words outside of $L_k(n,S)$. Now, fix a word $w \in L_k(n,S)$ and let us show that one the machines $\{M_{k,S \cap T,S \cap T}\}_{T \in U}$ accepts it. Let $S_1 \subseteq S$ be the set of the first $\lceil k/2 \rceil$ symbols
that appear in \( w \). Let \( S_2 \subseteq S \) be the set of the last \( \lfloor k/2 \rfloor \) symbols that appear in \( w \). Note that as \( w \in L_k(n, S) \), \( S_1 \) and \( S_2 \) must be disjoint and \( |S_1 \cup S_2| = k \). From the property of an \((n,k)\)-universal set, there must exists a set \( T \in U \) such that \( T \cap (S_1 \cup S_2) = S_1 \). For this \( T \) \( M_{k,S \cap T} \) accepts \( w \). We have shown that \( L(M_{k,S}) = L_k(n, S) \). Now let us bound the size of \( M_{k,S} \). For \( k \leq n \), denote by \( T_k \) the maximum over \( S \subseteq [n] \) of the size of the NFA \( M_{k,S} \) constructed in this way. Using this notation we have for any disjoint subsets \( S_1, S_2 \subseteq [n] \) that
\[
|M_{k,S_1,S_2}| \leq T_{[k/2]} + T_{[k/2]} + 1 \leq 2 \cdot T_{[k/2]} + 1
\]
, where \( |M_{k,S_1,S_2}| \) denotes the size of \( M_{k,S_1,S_2} \) in the construction described above. Now note that \( M_{k,S} \) consists of \([U]\) copies of machines \( M_{k,S_1,S_2} \) (and the \( \epsilon \)-transitions to these copies). Using this we have
\[
T_k \leq 2^k \cdot k^{O(log k)} \cdot \log n \cdot 2 \cdot (T_{[k/2]} + 1) + 2^k \cdot k^{O(log k)} \cdot \log n + 1 = 2^k \cdot k^{O(log k)} \cdot \log n \cdot T_{[k/2]}.
\]
Using Claim \( 5.2 \) and \( T_1 \leq 3n \) we get
\[
T_k \leq 2^{2k+2 log k} \cdot k^{O(log^2 k)} \cdot \log n^{log k+1} \cdot 3n
\]
Using the fact that for any \( k \), either \( \log n^{log k} \leq k^{log^2 k} \) or \( \log n^{log k} \leq n \) we can write
\[
T_k = O^*(4^k \cdot k^{O(log^2 k)}).
\]

6 Non-deterministic XOR automata for \( L_k(n) \)

Informally, a non-deterministic xor automaton (NXA) is simply an NFA where the acceptance criteria for a word is that there is an odd number of accepting paths for \( w \), rather than just one. It will be convenient to formally define the XOR-language of an NFA rather than formally defining NXAs.

**Definition 6.1 (The language \( L_\oplus \)).** Let \( M \) be a non-deterministic xor automaton over an alphabet \( \Sigma \). We define the XOR-language of \( M \), denoted \( L_\oplus(M) \subseteq \Sigma^* \), to be the set of words \( w \) that have an odd number of paths to an accept state in \( M \).

The purpose of this section is to construct a small set of NFAs of size \( O(2^k \cdot k \cdot n) \) such that the union of their XOR-languages is \( L_k(n) \). This construction can be viewed as an ‘automata interpretation’ of (a simplified version) of the algorithm for SIMPLE k-PATH of Abasi and Bshouty \([2]\). This will be used to get an algorithm for SIMPLE k-PATH with running time \( O^*(8^k) \). We proceed with the construction.

In the rest of this section sums are always in \( \mathbb{F}_2 \), i.e., modulo 2. For each non-empty subset \( S \subseteq [k] \), define the function \( \phi_S : \{(0,1)^k\}^k \rightarrow \{0,1\} \) by
\[
\phi_S(v_1, \ldots, v_k) \triangleq \prod_{i=1}^{k} \sum_{j \in S} v_{i,j}
\]
and define \( \phi : \{(0,1)^k\}^k \rightarrow \{0,1\} \) by
\[
\phi(v_1, \ldots, v_k) \triangleq \sum_{\emptyset \neq S \subseteq [k]} \phi_S(v_1, \ldots, v_k).
\]

From Ryser’s formula for the permanent\([12]\) we know that
Lemma 6.2. $\phi(v_1, \ldots, v_k)$ is equal to the determinant of the $k \times k$ matrix over $\mathbb{F}_2$ whose columns are $v_1, \ldots, v_k$.

Fix a $k \times n$ matrix $A$ over $\mathbb{F}_2$ with columns $v_1, \ldots, v_n \in \{0,1\}^k$. For each non-empty subset $S \subseteq [k]$, we define a function $f_{A, S} : [n]^k \to \{0,1\}$ by $f_{A, S}(i_1, \ldots, i_k) \triangleq \phi_S(v_{i_1}, \ldots, v_{i_k})$. We define $f_A : [n]^k \to \{0,1\}$ by

$$f_A(i_1, \ldots, i_k) \triangleq \phi(v_{i_1}, \ldots, v_{i_k}) = \sum_{\emptyset \neq S \subseteq [k]} \phi_S(v_{i_1}, \ldots, v_{i_k}) = \sum_{\emptyset \neq S \subseteq [k]} f_{A, S}(i_1, \ldots, i_k).$$

Lemma 6.3. Fix any $k \times n$ matrix $A$ over $\mathbb{F}_2$ and non-empty $S \subseteq [k]$. There is a deterministic automaton $M_{A, S}$ for $f_{A, S}^{-1}(1)$ with $k+1$ states and at most $k \cdot n$ edges.

Proof. Let $v_1, \ldots, v_n$ be the columns of $A$. Let $T \subseteq [n]$ be the set of elements $i \in [n]$ such that

$$\sum_{j \in S} v_{i,j} = 1.$$

Observe that $f_{A, S}(i_1, \ldots, i_k) = 1$ if and only if $i_1, \ldots, i_k$ are all contained in $T$. This motivates the following construction: $M_{A, S}$ will contain the start state $q_0$, and the states $q_1, \ldots, q_k$. $q_k$ will be the only accept state. For each $0 \leq j \leq k-1$, and for every $i \in S$. There will be an edge from $q_j$ to $q_{j+1}$ labeled $i$.

Theorem 6.4. Fix any positive integers $k \leq n$ and any $k \times n$ matrix $A$ over $\mathbb{F}_2$. There is an NFA $M_A$ over $[n]$ of size $O(2^k \cdot k \cdot n)$ such that $L_\Xi(M_A) = f^{-1}_A(A)$.

Proof. For every non-empty $S \subseteq [k]$, $M_A$ will contain a copy of the automaton $M_{A, S}$ as described in Lemma 6.3. We unite the start state $q_0$ and accept state $q_k$ of all the automata $M_{A, S}$ to one start state $q_0$ and accept state $q_k$ of $M_A$. $L_\Xi(M_A)$ contains exactly the words $(i_1, \ldots, i_k)$ that are accepted by an odd number of the automata $M_{A, S}$. Since $L(M_{A, S}) = f_{A, S}^{-1}(1)$, this is exactly $f_A^{-1}(1)$.

6.1 Covering matrices

We wish to show there is a small set of matrices $A$ such that the union of the XOR-languages of the corresponding automata $M_A$ is equal to $L_k(n)$. This motivates the following definition.

Definition 6.5. Let $A$ be a set of $k \times n$ matrices over $\mathbb{F}_2$. We say $A$ is $(n, k)$-covering, if for every subset of $k$ distinct columns $I = (i_1, \ldots, i_k) \subseteq [n]$, there is a matrix $A \in A$ such that the columns $(i_1, \ldots, i_k)$ in $A$ are linearly independent.

From now on for $I = (i_1, \ldots, i_k) \subseteq [n]$ and a $k \times n$ matrix $A$ over $\mathbb{F}_2$ we denote by $A_I$ the restriction of $A$ to the columns $(i_1, \ldots, i_k)$.

Lemma 6.6. Fix any positive integers $k \leq n$. There exists a set $\mathcal{A}$ of $k \times n$ matrices over $\mathbb{F}_2$ that is $(n, k)$-covering with $|\mathcal{A}| \leq 2k \cdot \log n$.

Proof. We use the probabilistic method. It is known that when choosing a random $k \times k$ matrix over $\mathbb{F}_2$ the probability that it is non-singular is at least half. Fix $I = (i_1, \ldots, i_k) \subseteq [n]$. It follows that when choosing a random $k \times n$ $A$ matrix over $\mathbb{F}_2$, the probability that $A_I$ is singular is at most half. Thus, when independently choosing $2k \cdot \log n$ random $k \times n$ matrices $A^1, \ldots, A^{2k \cdot \log n}$ the probability that the columns $I$ are dependent in all of them is at most $2^{-2k \cdot \log n} = n^{-2k}$. Taking a union bound over all $\binom{n}{k} \leq n^k$ choices of $I$ we see there must be a choice of $\mathcal{A} = \{A^1, \ldots, A^{2k \cdot \log n}\}$ that is $(n, k)$-covering.
Theorem 6.7. Fix any positive integers \( k \leq n \). Let \( \mathcal{A} \) be a family of \( k \times n \) matrices over \( \mathbb{F}_2 \) that is \((n,k)\)-covering. Then the union of languages \( \bigcup_{A \in \mathcal{A}} \mathcal{L}_\oplus(M_A) \) is equal to \( \mathcal{L}_k(n) \).

Proof. Fix a word \( w = (i_1, \ldots, i_k) \in [n]^k \) that is not in \( \mathcal{L}_k(n) \). Then for any \( k \times n \) matrix \( A \), \( f_A(i_1, \ldots, i_k) \) is equal to the determinant of a \( k \times k \) matrix that has at least two identical columns so \( f_A(i_1, \ldots, i_k) = 0 \). This exactly means that \( w \notin \mathcal{L}_\oplus(M_A) \). On the other hand, given \( w = (i_1, \ldots, i_k) \in \mathcal{L}_k(n) \), i.e. \( i_1 \neq \ldots \neq i_k \), we have some \( A \in \mathcal{A} \) such that the columns \((i_1, \ldots, i_k)\) in \( A \) are linearly independent. For this \( A \), \( f_A(i_1, \ldots, i_k) = 1 \) and therefore \( w \in \mathcal{L}_\oplus(M_A) \). \( \square \)

Corollary 6.8. Fix any positive integers \( k \leq n \). There is a \( \mathcal{M} \) of \( 2k \cdot \log n \) NFAs, each of size at most \( O(n \cdot 2^k) \) such that \( \bigcup_{M \in \mathcal{M}} \mathcal{L}_\oplus(M) = \mathcal{L}_k(n) \).

6.2 An algorithm for SIMPLE \( k \)-PATH via XOR automata

We now construct an NFA whose XOR-language is the set of simple \( k \)-paths in a graph.

Corollary 6.9. Fix any positive integers \( k \leq n \). Fix a directed graph \( G = \langle [n], E \rangle \). Fix vertices \( s, t \in V \). There is a set \( \mathcal{N} \) of \( 2k \cdot \log n \) NFAs, each of size at most \( O^*(2^k) \) such that \( \bigcup_{N \in \mathcal{N}} \mathcal{L}_\oplus(N) \) is exactly the set of simple \((s,t)\)-\( k \)-paths in \( G \).

Proof. We take the family \( \mathcal{M} \) of NFA’s from Corollary 6.8. For each \( M \in \mathcal{M} \) we compute the intersection NFA \( N = M \cap M(G, s, t) \).

Note that the number of accepting paths of a word \( w \) in \( N \) is the product of the number of accepting paths in \( M \) and \( M(G, s, t) \). As \( M(G, s, t) \) is deterministic, this means \( \mathcal{L}_\oplus(N) \) is exactly the set of words in \( \mathcal{L}_\oplus(M) \) that are also \((s,t)\)-paths in \( G \). We take \( \mathcal{N} \) to be the set of all these NFAs \( N \). Hence \( \bigcup_{N \in \mathcal{N}} \mathcal{L}_\oplus(N) \) is the intersection of \( \mathcal{L}_k(n) \) with the set of \((s,t)\)-\( k \)-paths in \( G \). \( \square \)

The work of Vuillemin and Gama [13] on minimizing NXA gives in particular a method to check if the XOR-language of an NFA is empty.

Theorem 6.10 ([13]). There is a deterministic algorithm, that given an NFA \( M \) with \( s \) states, checks in time \( O(s^3) \) whether \( \mathcal{L}_\oplus(M) = \emptyset \).

Given a set of \( \mathcal{A} \) of \((n,k)\)-covering matrices with we could now use Theorem 6.10 to solve SIMPLE \( k \)-PATH in deterministic time \( |\mathcal{A}| \cdot O^*(8^k) \). However, currently there are no explicit constructions of such sets with \( |\mathcal{A}| = n^{O(1)} \). The only explicit construction we are aware of is implicit in Lemma 51 of Bshouty [4] and gives \( |\mathcal{A}| = 2^{O(k)} \cdot \log n \). Choosing \( |\mathcal{A}| \) randomly would lead to a randomized algorithm for SIMPLE \( k \)-PATH with running time \( O^*(8^k) \). We state two open problems whose solution could lead to an \( O^*(2^k) \) deterministic algorithm for SIMPLE \( k \)-PATH.

Corollary 6.11. Suppose that

\bullet Given integers \( k \leq n \) we can construct a set \( \mathcal{A} \) of \((n,k)\)-covering matrices in time \( n^{O(1)} \) with \( |\mathcal{A}| = n^{O(1)} \).

\bullet Given an NFA \( N \) we can check in deterministic time \( O(\text{size}(N)) \) whether \( \mathcal{L}_\oplus(N) = \emptyset \).

Then we can solve SIMPLE \( k \)-PATH deterministically in time \( O^*(2^k) \).

Acknowledgements

We thank Hasan Abasi, Nader Bshouty, Michael Forbes and Amir Shpilka for helpful conversations.
References

[1] D. R. Karger T. Lane A. Meyerson A. Blum, S. Chawla and M. Minkoff. Approximation algorithms for orienteering and discounted-reward tsp. SIAM J. Comput., 37(2):653–670, 2007.

[2] H. Abasi and N. Bshouty. A simple algorithm for undirected hamiltonicity. Electronic Colloquium on Computational Complexity (ECCC), 20:12, 2013.

[3] M. Bateni and J. Chuzhoy. Approximation algorithms for the directed $k$-tour and $k$-stroll problems. Algorithmica, 65(3):545–561, 2013.

[4] N. Bshouty. Testers and their applications. Electronic Colloquium on Computational Complexity (ECCC), 19:11, 2012.

[5] D. Lokshtanov F. V. Fomin and S. Saurabh. Efficient computation of representative sets with applications in parameterized and exact algorithms. CoRR, abs/1304.4626, 2013.

[6] M. L. Fredman and R. E. Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. J. ACM, 34(3):596–615, 1987.

[7] I. Glaister and J. Shallit. A lower bound technique for the size of nondeterministic finite automata. Inf. Process. Lett., 59(2):75–77, 1996.

[8] S. Sze J. Chen, S. Lu and F. Zhang. Improved algorithms for path, matching, and packing problems. In SODA, pages 298–307, 2007.

[9] S. Richter J. Kneis, D. Mölle and P. Rossmanith. Divide-and-color. In WG, pages 58–67, 2006.

[10] L. J. Schulman M. Naor and A. Srinivasan. Splitters and near-optimal derandomization. In FOCS, pages 182–191, 1995.

[11] U. Zwick N. Alon, R. Yuster. Color coding. In Encyclopedia of Algorithms. 2008.

[12] H. J. Ryser. Combinatorial mathematics, the carus mathematical monographs, the mathematical association of america. 1963.

[13] J. Vuillemin and N. Gama. Compact normal form for regular languages as xor automata. In CIAA, pages 24–33, 2009.