Finding a path of length k in $O^*(2^k)$ time

Spyridon Maniatis-MPLA

June 28, 2013

1 Introduction

Let G be a simple graph and k a natural number. The k-path problem is to determine if G has a k-path of length at least k and to produce one (if the answer is yes). When k is given as part of the input, the problem is known to be NP-complete. The obvious solution of enumerating all possible k-paths in a graph with n nodes needs $\Theta(n^k)$ time therefore is polynomial only for $k = O(1)$ (a constant). Some work done on the problem:

- The first algorithm to reduce the dependency on k was given by Monien[1985]: Deterministic $O^*(k!)$ (O^* supresses poly(n,k) factors) which is polynomial for $k \leq (logn/loglogn)$.
- Alon, Yuster, Zwick [1995]: Randomized $O^*((2e)^k) \leq O^*(5.44^k)$ and deterministic $O^*(c^k)$, where c is a large constant. So they answered the important question of if there is a polynomial algorithm for the $O(logn)$-path problem.
- It is known for many years that when $k = n$ the problem is solvable in $O^*(2^k)$ time (Bellman[1962], Held and Karp[1962], Karp[1982]), so the natural question is if there is an algorithm that can match this runtime for all values of k.

Some faster algorithms have recently appeared in the literature:

- In 2006 two groups discovered independently $O^*(4^k)$ randomized and $O^*(c^k)$ deterministic algorithms (Kneis, Molle, Richter, Rossmanith with $c = 16$ and Chen, Lu, Sze, Zhang with $c = 12.5$)
- Koutis[2008]: Randomized $O^*(2^{3k/2}) \leq O^*(2.83^k)$ time (Some of his ideas will be used to give a randomized $O^*(2^k)$ time algorithm for the k-path problem).

The best known algorithms for finding a Hamilton path in a n-node graph run in $O^*(2^k)$ time, therefore any significant improvement in the runtime dependence on k given by the algorithm we will present would imply a faster Hamilton path algorithm and would be a breakthrough in algorithms for NP-hard problems (that's why it is a rather difficult, if not impossible, task).

2 Some preliminaries

Let F be a field and G a multiplicative group. We define the group algebra F[G]:

- elements : $\sum_{g\in G} a_g g$ where $a_g \in F$ for every $g \in G$.
- Addition in F[G]: $(\sum_{g\in G} a_g g) + (\sum_{g\in G} b_g g) = \sum_{g\in G} (a_g + b_g)g$.
- Multiplication in F[G]: $(\sum_{g\in G} a_g g) \cdot (\sum_{h\in G} b_h h) = \sum_{g,h\in G} a_g b_h g$.
- F[G] is a ring with zero the element $0 = \sum_{g\in G} a_g g$ where $a_g = O_F$ for every $g \in G$ and one the $1 \in G$.

As we will see later, we will work with the group algebra $GF(2^l)[{\mathbb{Z}_2}^k]$ where ${\mathbb{Z}_2}^k$ is the group of binary k-vectors with operation the addition modulo 2 and $GF(2^l)$ is the unique field on 2^l elements. We use W_0 to denote the all-zeros vector of ${\mathbb{Z}_2}^k$. Note that every $v \in {\mathbb{Z}_2}^k$ is its own inverse as $v^2 = W_0$.

3 The algorithm

Fix a simple graph G with vertex set $\{1, \dots, n\}$. Let F be a field, A the adjency matrix of G, $x_1, \dots x_n$ variables, $B[i,j] = A[i,j]x_i$, $\vec{1}$ the row n-vector af all 1's and $\vec{x}$ the column vector defined by $\vec{x}[i] = x_i$. Define the k-walk polynomial to be $P_k(x_1, \dots, x_n) = \vec{1} \cdot B^{k-1} \cdot \vec{x}$.

Proposition 3.1. $P_k(x_1, \cdots, x_n) = \sum_{i_1,\cdots,i_k \textit{ is a walk in } G} x_{i_1}, \cdots x_{i_k}$

There is a k-path in G iff $P_k(x_1, \cdots, x_k)$ contains a multilinear term. We give a randomized algorithm R with the following property:

- If P_k has a multilinear term, then $Pr[R \text{ outputs yes}] \geq 1/5$.
- If P_k does not have a multilinear term then R outputs no.

Theorem 3.1. *Let $P(x_1, \dots, x_n)$ be a polynomial of degree at most k, represented by an arithmetic circuit of size $s(n)$ with + gates (of unbounded fan-in), × gates (of fan-in two) and no scalar multiplication. There is a randomized algorithm that on every P runs in $O^*(2^k s(n))$ and answers yes with high probability if there is a multilinear term in the sum-product expansion of P and no if there is not one.*

Observation 3.1. *P_k can be implemented with a crcuit of size $O(k(m + n))$ where $m = |E(G)|$ and this way we can obtain our k-path algorithm.*

Here is our **basic idea**: Substitute random group elements for the variables such that all non-multilinear terms in P evaluate zero and some multilinear terms survive. We augment the scalar free multiplication circuit with random scalar multiplications over a field large enough that the remaining multilinear polynomial evaluates to nonzero with decent probability. We set $F = GF(2^{8+logk})$ (the unique field with $k + 8$ elements).

We are now ready to describe the **algorithm**: Pick n uniform random vectors $v_1, \ldots, v_n$ from ${\mathbb{Z}_2}^k$. For each multiplication gate g_i in the circuit for P, pick a uniform random $w_i \in F - \{0\}$. Insert a new gate that multiplies the output of g_i with w_i and provides the output to those gates that read the output og g_i. Let $P^{'}$ be the new polynomial represented by the arithmetic circuit. Output yes iff $P^{'}(W_0 + v_1, \ldots, W_0 + v_n) \neq 0$.

Runtime: The only non-trivial step is the evaluation of the polynomial that we get at the end. By definition the evaluation of $P^{'}(W_0 + v_1, \ldots, W_0 + v_n)$ takes $O(s(n))$ arithmetic operations but we have to account the cost of arithmetic in the group algebra $F[{\mathbb{Z}_2}^k]$. The elements of $F[{\mathbb{Z}_2}^k]$ can be naturally interpreted as vectors in F^{2^k}. Addition can be done in $O(2^k log|F|)$ time (with a component-wise sum) and ultiplication of vectors u and v over the group algebra in $O(k2^k log^2|F|)$ tine by a Fast Fourier Transformation style algorithm.

Correctness: We first look at a crucial observation of Koutis.

Observation 3.2. *For any $v_i \in Z_2^k$, $(W_0 + v_i)^2 = {W_0}^2 + 2v_i + {v_i}^2 = W_0 + 0 + W_0 \ mod\ 2$. Therefore all squares in P vanish in $P^{'}(W_0 + v_1, \ldots, W_0 + v_n)$ since F has characteristic 2. So if $P(x_1, \ldots x_n)$ does not have a multilinear term, then $P^{'}(W_0 + v_1, \ldots, W_0 + v_n) = 0$ over $F[{\mathbb{Z}_2}^k]$ regardless of the choices of v_i.*

We prove that the if sum-product expansion of $P(x_1, \ldots x_n)$ has a multilinear term, then $P^{'}(W_0 + v_1, \ldots, W_0 + v_n) \neq 0$ with probability at least 1/5, over the random choices of w_i's and v_i's. We may assume that every multilinear term in the sum-product expansion of P has the form $c \cdot x_{i_1}, \ldots x_{i_{k'}}$, where $k\prime \geq k$ and $c \in \mathbb{Z}$. For each one of them thereis a collection of corresponding multilinear terms in $P^{'}$ of the form: $w_1 \ldots w_{k'-1} \cdot \prod_{j=1}^{k'} (W_0 + v_i)$, where $w_1, \ldots w_{k'-1}$ distinct for every term, as the sequence of multiplication gates $g_1, \ldots, g_{k'-1}$ are distinct.

Proposition 3.2 (Koutis)**.** *If $v_1, \ldots, v_i \in {\mathbb{Z}_2}^k$ are linearly depended over GF(2), then $\prod_{j=1}^{k'} (W_0 + v_i) = 0$ in $F[{\mathbb{Z}_2}^k]$.*

When $v_1, \ldots, v_i$ are linearly independed, $\prod_{j=1}^{k'} (W_0 + v_i)$ is the sum over all vectors in the span of $v_1, \ldots, v_i$ since each vector in the span is of the form $\prod_{j \in S} v_j$ for some $S \subseteq [i]$ and there is a unique way to generate it. This observation, the last proposition state and the fact that any $k^{'} \geq k$ vectors are linearly depended gives us that $P^{'}(W_0 + v_1, \ldots, W_0 + v_n)$ evaluates to either 0 or $c \sum_{v \in {\mathbb{Z}_2}^k} v$ for some $c \in F$. We are ready for our final argument: If P has a multilinear term, then $c \neq 0$ with probability at least 1/5.

The vectors $v_{l_1}, \ldots, v_{l_k}$ chosen for the variables in a multilinear term of P are linearly independed with probability at least 1/4 , because (Blum, Kannan[1995]) a random $k \times k$ matrix over GF(2) has full rank with probability at leat $0.28 \geq 1/4$. Thus, in $P^{'}(W_0 + v_1, \ldots, W_0 + v_n)$ there is at least one multilinear term in P corresponding to a set of k linearly independed vectors, with probabolity a least 1/4.

Each coefficient c_i comes from a sum of products ok k-1 elements with $w_{i,1},, w_{i,k-1}$ corresponding to some multiplication gates $g_{i,1},, g_{i,k-1}$ in the circuit. If we see w_i's as variables, $Q(w_1, \ldots, w_{s(n)}) = \sum_i c_i$ is a degree-k polynomial over F. Then Q is not identically 0 and by Schwartz-Zippel Lemma we get that the algorithm's random assignment to the variables of Q results in an evaluation $0 \in F$ with probability at most $k/|F| = 1/2^3$. And so $Pr[\sum_i c_i = 0] \leq 1/2^3$. The overakk probability of success is at least $1/4 \cdot (1 - 1/2^3) \geq 1/5$.

Constructing a path: For an arbitrary node v_i, we remove v_i from the graph and run the k-apth detection algorithm for $O(logn)$ trials, using new random bits for each trial. If the algorithm outputs yes in some trial, we recursivelt call it on tge graoh with v_i removed, returning the k-path that it returns. Otherwise, we add v_i back to the graph and move to the next candidate node v_{i+1}, noting that such a move occurs at most k times (with high probability). We can bound the runtime with the reccurence: $T(n) \leq O^*(2^k \cdot klogn) + T(n-1)$ which is $O^*(2^k)$. the overall probability error can be bounded by a constant less than 1, since the probability that all $O(logn)$ trials result in error is inversely polynomial in n.

4 Conclusion

Two interesting open questions conjectured to have positive answers:

1. Let G be a graph with costs on its edges. The SHORT CHEAP TOUR problem is to find a path of length at least k where the total sum of costs on the edges is minimized. This problem is fixed-parameter tractable, in fact: SHORT CHEAP TOUR can be solved in $O^*(4^k)$ time by a randomized algorithm that succeeds with high probability.
   Can SHORT CHEAP TOUR be solved in $O^*(2^k)$ time?

2. Is there a deterministic algorithm for k-path with the same runtime complexity as our algorithm? A polytime derandomization of this algorithm (which relies on the fact that polynomial identity testing is in RP) would imply strong circuit lower bounds (Impagliazzo and Kabanets[2004]) that is why Koutis algorithm may be easier to derandomize.