INFORMATIQUE THÉORIQUE ET APPLICATIONS

S. BASAGNI
D. BRUSCHI
F. RAVASIO

On the difficulty of finding walks of length k

Informatique théorique et applications, tome 31, n° 5 (1997), p. 429-435

<http://www.numdam.org/item?id=ITA_1997__31_5_429_0>

© AFCET, 1997, tous droits réservés.

L'accès aux archives de la revue « Informatique théorique et applications » implique l'accord avec les conditions générales d'utilisation (http://www.numdam.org/conditions). Toute utilisation commerciale ou impression systématique est constitutive d'une infraction pénale. Toute copie ou impression de ce fichier doit contenir la présente mention de copyright.

NUMDAM

Article numérisé dans le cadre du programme
Numérisation de documents anciens mathématiques
http://www.numdam.org/

Informatique théorique et Applications/Theoretical Informatics and Applications
(vol. 31, n° 5, 1997, pp. 429-435)

ON THE DIFFICULTY OF FINDING WALKS OF LENGTH k (*)

by S. BASAGNI, D. BRUSCHI and F. RAVASIO ([1])

Abstract. – *We characterize the computational complexity of the following combinatorial problem: Given a directed graph $G = (V, E)$ endowed with a length function $w : E \to \mathbf{N}$, a pair of nodes s and t in V and an integer $k \geq 0$, does G contain a walk π from s to t of length exactly k? We show that the problem is NP-complete when G is a directed graph, an undirected graph, or a directed acyclic graph. The problem becomes NL-complete when w is a unary function.*

Keywords: *Combinatorial problems, computational complexity.*

Résumé. – *Nous caractérisons la complexité du problème combinatoire suivant : étant donné un graphe $G = (V, E)$ muni d'une fonction de longueur $w : E \to \mathbf{N}$, d'un couple de nœuds s et t dans V et d'un entier $k \geq 0$, le graphe G contient-il un chemin π de s vers t de longueur exactement k ? Nous montrons que le problème est NP-complet lorsque G est un graphe orienté, un graphe non orienté ou un graphe orienté acyclique. Le problème est NL-complet lorsque la fonction w est unitaire.*

1. INTRODUCTION

Path-finding problems consist of determining a walk in a graph satisfying certain conditions. They have been widely investigated in algorithmic graph theory for a long time, especially in the *shortest* path version and in the *longest* path version.

In the SHORTEST PATH problem (respectively, LONGEST PATH problem) given a directed graph $G = (V, E)$ endowed with a length function $w : E \to \mathbf{N}$, a pair of nodes s and t in V and an integer $k \geq 0$, one has to decide if G contains a path ([2]) from s to t of length $\leq k$ ($\geq k$). Shortest paths can be efficiently found by sequential or by NC algorithms [5, 7]. Instead, the LONGEST PATH problem is NP-complete [6]. Feasible solutions to the LONGEST

(*) Received November 1, 1996.

([1]) Dipartimento di Scienze dell'Informazione, Università degli Studi di Milano, Via Comelico 39/41, 20135 Milano, Italy.
{basagni,dbruschi,ravasio}@dsi.unimi.it

([2]) A path is a walk without cycles, also called *simple* paths or *elementary* paths in the literature.

Informatique théorique et Applications/Theoretical Informatics and Applications
0988-5004/97/05/© AFCET-Elsevier-Paris

PATH problem, however, can be found in graphs that have no positive cycles as for example in directed acyclic graphs (DAGs) [9].

Probably the most natural path-finding problem is the *exact* path version, asking for walks of length equal to an integer $k \geq 0$. The crucial element that distinguishes between the exact path version and both the longest and the shortest path versions is the presence of cycles in the solution. Thus, when looking for an exact path in a graph we must explicitly indicate if we are interested in either paths or walks.

The k-PATH problem, namely the problem of detecting a path of length k in a weighted directed graph, is NP-hard since it is a special case of the HAMILTONIAN PATH problem. To our knowledge, the best algorithm for finding a path of length k in an unweighed directed graph $G = (V, E)$ works in either $2^{\mathcal{O}(k)} \mid E \mid \log \mid V \mid$ or $\mathcal{O}(k! \mid E \mid)$ time [1].

In this paper we investigate the problem of deciding whether a weighted directed graph contains a walk of length k. Specifically, we characterize the computational complexity of the following combinatorial problem:

k-WALK

Input: A digraph $G = (V, E)$ endowed with a length function $w : E \to \mathbf{N}$, a pair of nodes s, t in V, an integer $k \geq 0$.

Question: Does there exist in G a walk π from s to t of length k?

The peculiarity of the k-WALK problem is that a solution might contain up to k occurrences of the same vertex, (i.e., a node might appear in a walk a number of times exponential in the input size.) This problem raises the question of whether a candidate solution to k-WALK can be checked in polynomial time and thus whether the problem is in NP. We also notice that the NP-hardness of k-WALK can not be inferred by the NP-hardness of the k-PATH problem.

Using a suitable walk-encoding scheme we prove that the k-WALK problem is NP-complete. We also show that it remains NP-complete when G is either an undirected graph or a DAG. The k-WALK problem can be solved in polynomial time when restricted to unweighted graphs (and $k = \mid V \mid^{\mathcal{O}(1)}$) or unweighted DAGs [8].

Our results show that, while in the general case the k-WALK problem and the LONGEST PATH problem are computationally equivalent, the former is harder than the latter when they are restricted to DAGs (unless P=NP). Furthermore, while the LONGEST PATH problem is NP-complete even for unweighed graphs, we point out that when k and the arc lengths of G are written in unary, the k-WALK problem is NL-complete.

2. PRELIMINARIES

The reader is expected to be familiar with basic concepts from graph theory and complexity theory (see, for example, [6, 11]). In the following, we briefly describe the conventions adopted throughout the paper.

A directed graph, or *digraph*, $G = (V, E)$, consists of a finite *vertex* or *node set* $V = \{1, 2, \ldots, n\}$ and an *arc set* $E = \{e_1, e_2, \ldots, e_m\}$ of ordered pairs of nodes (u, v), $u \neq v$, with u being the *tail* and v being the *head*. The *in-degree* $\delta^-(v)$ (respectively, *out-degree* $\delta^+(v)$) of a node v is the number of arcs having v as its head (tail).

An undirected *graph* is a digraph whose arc set is composed of unordered pairs. A *multigraph* is a digraph containing repeated arcs (*multiple arcs*) or arcs with both endpoints the same (*loops*). The *underlying graph* of a directed multigraph $G = (V, E)$ is the graph $H = (V', E')$, where $V' = V$ and $E' = \{(u, v), (v, u) \mid (u, v) \in E \vee (v, u) \in E\}$. A *weighted* digraph $G = (V, E)$ is a digraph associated with a *length function* $w : E \to \mathbf{N}$, where $\mathbf{N}$ is the set of the natural numbers $\{0, 1, 2, \ldots\}$. For each arc $e = (u, v) \in E$, both w_e and $w_{u,v}$ denote the *length* of e.

Given a digraph $G = (V, E)$ and a pair of nodes v_1, $v_{\ell+1}$ in V, a *walk* from the node v_1 to the node $v_{\ell+1}$ is usually defined as a sequence $\pi = (v_1, e_1, v_2, e_2, \ldots, e_{\ell-1}, v_\ell, e_\ell, v_{\ell+1})$ of nodes and arcs, such that $e_i = (v_i, v_{i+1}) \in E$, $1 \leq i \leq \ell$. Whenever it will be possible, we shall denote a walk as the sequence of either its arcs or its nodes. We do not impose any restriction on how many times a vertex is contained in a solution of the k-WALK problem. However, notice that such a number is at most exponential in the input size. Thus, in order to get a reasonable encoding scheme (in the meaning of [6]) for a solution of the k-WALK problem, we adopt the following encoding. Given a digraph $G = (V, E)$, let $\pi = (v_1, e_1, v_2, e_2, \ldots, e_{\ell-1}, v_\ell, e_\ell, v_{\ell+1})$ be a walk contained in G. Then, we represent π as the multiset $M = \{\langle e, 1 + \mu_e \rangle \mid e \in E, \mu_e \in \mathbf{N}\}$, where $1 + \mu_e$ is the number of times arc e is in the walk π. It can be easily verified that M has size polynomial in the length of the instance of the k-WALK problem.

Given a digraph $G = (V, E)$, a *path* is a walk containing each vertex in V at most once, a *trail* is a walk containing each arc in E at most once. A *Eulerian path* is a trail that traverses every arc of G. A *cycle* is a path in which v_1, $v_{\ell+1}$ coincide. A digraph without cycles is called a *directed acyclic graph (DAG)*. The length of a walk is the sum of the lengths of its

arcs. $G = (V, E)$ is *connected* if, for each pair of vertices s and t in V, G contains a walk from s to t or from t to s.

The complexity classes we refer to are NP, the class of decision problems solved by a nondeterministic polynomial time algorithm, and NL, the class of decision problems solved by a nondeterministic logarithmic space algorithm.

3. PROBLEM COMPLEXITY

In this section we study the computational complexity of the k-WALK problem. We prove that the k-WALK problem is NP-complete in the general case (on weighted digraphs) and also when it is restricted to weighted graphs and weighted DAGs. We first show that this problem belongs to NP. Next, we show that SUBSET SUM, a well known NP-complete problem [6], can be reduced via a polynomial time many-one reduction to the k-WALK problem. Thus the k-WALK problem is NP-complete. Finally, we point out that the NP-completeness result can be extended to undirected graphs and DAGs. We also prove that the k-WALK problem is NL-complete when k and the length function of G are polynomially bounded.

By representing walks as multisets we are able to prove the membership of the k-WALK problem in NP. In the following lemma we exhibit a certificate for such multisets, namely a deterministic polynomial time algorithm that given a weighted digraph $G = (V, E)$, a pair of vertices s and t in V, an integer $k \geq 0$ and a multiset $M = \{\langle e, 1 + \mu_e\rangle \mid e \in E, \mu_e \in \mathbf{N}\}$, answers "yes" if and only if M encodes a walk in G from s to t of length k. It follows that the k-WALK problem is in NP.

LEMMA 1: *The k-WALK problem is in NP.*

Proof: Given an instance $\mathcal{I} = \langle G, w, s, t, k\rangle$ of the k-WALK problem, we guess a feasible solution $M = \{\langle e, 1 + \mu_e\rangle \mid e \in E, \mu_e \in \mathbf{N}\}$ and verify in time polynomial in the size of $\mathcal{I}$ that M represents a walk from the node s to the node t of length k using the following method.

Consider the directed multigraph $G_M = (V_M, E_M)$, where E_M contains $1 + \mu_e$ copies of e, for each arc $e = (u, v)$ in M, and u, v are in V_M. Let G'_M be the underlying graph of G_M.

M represents a walk in G from s to t if and only if the following conditions are satisfied: (i) vertices s and t are in V_M, (ii) G'_M is connected and (iii) G_M contains a Eulerian path from s to t. Such conditions can be checked by a deterministic algorithm in time polynomial in the size of G. G'_M can

be built directly from M and Step (ii) can be implemented by a breadth-first search in G'_M. By a classical theorem (see, e.g., [11]), we perform Step (iii) comparing the degrees of the nodes of G_M. More specifically, G_M contains a Eulerian path from s to t if and only if G'_M is connected and for each $v \in V_M \backslash \{s, t\}$, $\delta^+(v) = \delta^-(v)$, $\delta^-(s) = \delta^+(s) + 1$, and $\delta^+(t) = \delta^-(t) + 1$. Such values can be easily determined from M.

Finally, the length of the walk encoded by M is k if and only if $\sum_{\langle e, \mu_e \rangle \in M} w_e \cdot (1 + \mu_e) = k$. ■

In the following theorem we show a deterministic polynomial time algorithm for reducing the well-known NP-complete SUBSET SUM problem to the k-WALK problem.

THEOREM 1: *The k-WALK problem is NP-complete.*

Proof: Given an instance $\mathcal{I}$ of SUBSET SUM represented by a set $A = \{a_1, a_2, \ldots, a_n\}$ of items, sizes $\langle x_1, x_2, \ldots, x_n \rangle \in \mathbf{N}^n$ and a target $B \in \mathbf{N}$, $B < \sum_{i=1}^{n} x_i$, we construct a weighted digraph $G_{\mathcal{I}} = (V, E)$ consisting of a sequence of n copies of the same gadget G_i, one for each x_i.

Each gadget $G_i = (V_i, E_i)$ is a weighted digraph so defined (see Figure 1): $V_i = \{s_i, t_i, u_i, v_i\}$, $E = \{(s_i, t_i), (s_i, u_i), (u_i, v_i), (v_i, u_i), (u_i, t_i)\}$, $w_{s_i, t_i} = 2^{b+2} + x_i$, $w_{s_i, u_i} = 2^{b+1}$, $w_{u_i, v_i} = 2^b$, $w_{v_i, u_i} = 0$, $w_{u_i, t_i} = 2^{b+1}$, where $b = \lfloor \log (\sum_{i=1}^{n} x_i) \rfloor + 1$ is the minimum number of bits required for representing B and thus $2^b > B$. In order to get the graph $G_{\mathcal{I}}$, we connect the node t_i of G_i to the node s_{i+1} of G_{i+1}, for $1 \leq i \leq n - 1$. It can be easily verified that given $\mathcal{I}$, $G_{\mathcal{I}}$ can be built by a deterministic algorithm that works in time polynomial in the length of $\mathcal{I}$. It remains to prove that there exists a feasible solution for $\mathcal{I}$ if and only if $G_{\mathcal{I}}$ contains a suitable walk. More formally, we prove that there exists a solution for $\mathcal{I}$ if and only if $G_{\mathcal{I}}$ contains a walk from $s = s_1$ to $t = t_n$ of length $k = (4n + 2) \cdot 2^b + B$.

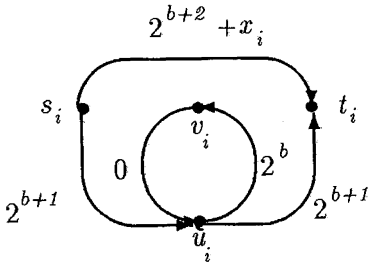

Figure 1. – The gadget G_i.

($\Leftarrow$) A walk π in G from s_1 to t_n of length $k = (4n + 2) \cdot 2^b + B$ must visit each gadget G_i, for $1 \leq i \leq n$, either through the arc (s_i, t_i) with a cost of $2^{b+2} + x_i$ or through the path (s_i, u_i, t_i) with a cost of 2^{b+2}. Furthermore, π can walk the cycle $\gamma_i = (u_i, v_i, u_i)$ a number c_i times, $c_i \geq 0$, with no extra cost if $c_i = 0$ or with an additional cost multiple of 2^b otherwise. Thus, π can be decomposed in a path π' from s_1 to t_n and in a certain number $c = \sum_i c_i$ of cycles $\Gamma = \cup_i \gamma_i$. Since no cycle in Γ contributes to the target value B encoded in k, the path π' must be of length $4n2^b + B$. After noticing that at least one path (s_i, u_i, t_i) has to be walked by π because $B < \sum_i x_i$, we conclude that π must contain exactly two cycles γ_j, γ_h, for some $1 \leq j, h \leq n$, in order to attain length k. In particular, π is a trail if $j \neq h$.

From the structure of π just illustrated, it follows that only the arcs of kind (s_i, t_i) contained in π' contribute to B. Hence, we can form a solution of SUBSET SUM picking the A's items that correspond to the arcs of π of type (s_i, t_i).

($\Rightarrow$) Given a solution A' of SUBSET SUM, we can find a walk π in G from node s_1 to node t_n of length $(4n + 2) \cdot 2^b + B$ as follows. For each $i = 1, 2, \ldots, n$, if a_i belongs to A' then go from s_i to t_i along with the arc (s_i, t_i) else if a_i is the first item that does not belong to A' then go from s_i to t_i along with the walk $(s_i, u_i, v_i, u_i, v_i, u_i, t_i)$, else along with the path (s_i, u_i, t_i). When the last element a_n has been considered, we have formed a walk from s to t of length $(4n + 2) \cdot 2^b + B$ as required. ■

It is not difficult to extend the proofs of Lemma 1 and Theorem 1 to k-WALK instances containing weighted graphs and weighted DAGs. In particular, in the latter case we have to delete the cycles γ_i's and set $k = 4n2^b + B$. Hence, in contrast with the longest path problem, the k-WALK problem remains NP-complete even when restricted to DAGs.

PROPOSITION 1: *The k-WALK problem remains NP-complete even when G is a graph.* ■

PROPOSITION 2: *The k-WALK problem remains NP-complete even when G is a DAG.* ■

It is known that the unary version of some intractable (unless P = NP) exact problems defined on graphs can be solved by polynomial time algorithms (and even NC algorithms) [2, 3, 4, 10], namely they are *pseudo-polynomial* problems [6]. In the present case we can be more precise since we can prove

that the unary k-WALK problem is logspace equivalent to the NL-complete directed GRAPH REACHABILITY problem.

PROPOSITION 3: *The unary version of the k-WALK problem is NL-complete.*

Proof: Given an instance $\mathcal{I} = \langle G, w, s, t, k \rangle$ of the unary k-WALK problem, where k and w are polynomially bounded in the size of G, we guess a sequence π of nodes and arcs in G. We can check by a deterministic algorithm that works in space bounded by a logarithmic function in the size of $\mathcal{I}$ if π is a walk in G from s to t of length k. Hence the unary version of the k-WALK problem belongs to NL.

Moreover, the instance $\langle G, s, t \rangle$ of the directed GRAPH REACHABILITY problem asking whether the node s is connected to the node t in the digraph G is reduced to the instance $\langle G, w, s, t, 0 \rangle$ of the k-WALK problem, where w is the constant length function $w : E \rightarrow \{0\}$. It easy to show that such a reduction can be computed by a deterministic logspace algorithm and to prove that G contains a walk of length 0 from s to t if and only if s is connected to t. ■

REFERENCES

1. N. ALON, R. YUSTER and U. ZWICK, Color-coding, *Journal of the ACM*, 1995, *42*, 4, pp. 844–856.
2. F. BARAHONA and W. R. PULLEYBLANK, Exact arborescences, matchings and cycles. *Discrete Applied Mathematics*, 1987, *16*, pp. 91–99.
3. D. BRUSCHI and F. RAVASIO, Random parallel algorithms for findings cycles, branchings and perfect matchings. *Algorithmica*, 1995, *13*, 4, pp. 346–356.
4. P. M. CAMERINI, G. GALBIATI and F. MAFFIOLI, Random pseudo-polynomial algorithms for exacts matroid problems. *Journal of Algorithms*, 1992, *13*, 2, pp. 258–273.
5. T. H. CORMEN, C. E. LEISERSON and R. L. RIVEST, *Introduction to Algorithms*, MIT Press and McGraw-Hill, 1990.
6. M. R. GAREY and D. S. JOHNSON, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, San Francisco, CA, 1979.
7. Y. HAN, V. PAN and J. REIF, Efficient parallel algorithms for computing all pair shortest paths in directed graphs. In *Proc. 4th ACM Symp. on Parallel Algorithms and Architectures*, 1992, pp. 353–362.
8. A. KAUFMANN, *Graphs, dynamic programming and finite games*. Academic Press, New York, 1967. Translation of v. 2 of Methodes et modèles de la recherche.
9. E. L. LAWLER, *Combinatorial Optimization: Networks and Matroids*. Holt Rinehart and Winston, New York, 1976.
10. C. H. PAPADIMITRIOU and M. YANNAKAKIS, The complexity of restricted spanning tree problems. *Journal of the ACM*, 1982, *29*, 2, pp. 285–309.
11. W. TUTTE. *Graph Theory*, Addison-Wesley, Reading, MA, 1984.