Annals of Discrete Mathematics 25 (1985) 239-254

HOW TO FIND LONG PATHS EFFICIENTLY

B. MONIEN
Universität Paderborn

We study the complexity of finding long paths in directed or undirected graphs. Given a graph $G = (V, E)$ and a number k our algorithm decides within time $O(k! \cdot |V| \cdot |E|)$ for all $u,v \in V$ whether there exists some path of length k from u to v. The complexity of this algorithm has to be compared with $O(|V|^{k-1} \cdot |E|)$ which is the worst case behaviour of the algorithms described up to now in the literature. We get similar results for the problems of finding a longest path, a cycle of length k or a longest cycle, respectively.

Our approach is based on the idea of representing certain families of sets by subfamilies of small cardinality. We also discuss the border lines of this idea.

1. Introduction

In this paper we study the problem of determining a path of length k in a directed or undirected graph $G = (V, E)$. This problem is closely related to the longest path problem and to some other problems which we will describe later. By a 'path' we always mean a 'simple path' (see [5]), i.e. we do not allow that a vertex appears on a path more than once. Without this restriction (i.e. by allowing a vertex to appear more than once on a path) or for graphs without cycles the problems is wellknown (see [10]) to be solvable in polynomial time whereas the problem of determining a simple path of length k, k arbitrary, is NP-complete. One can consider this problem as a single-source and single-destination problem (i.e. as the problem to decide for fixed $u,v \in V$ whether there exists a path of length k from u to v) or as the more general problem to decide for all $u,v \in V$ whether there exists a path of length k from u to v. We will study here the second approach. That is we want to compute a matrix $D^{(k)} = (d_{ij}^{(k)})$, $1 \leq i,j \leq n$, where $d_{ij}^{(k)}$ is equal to some path from i to j of length k, if such a path exists, and $d_{ij}^{(k)}$ is equal to some special symbol λ, if there exists no path from i to j of length k.

The straightforward algorithm which enumerates all sequences of length k+1 solves the problem within time $0(|V|^{k+1})$. It has to consider for every pair of nodes (i, j) and for any sequence $u_1, \ldots, u_{k-1}$ of nodes

$$i, u_1, \ldots, u_{k-1}, j$$

whether $i \to u_1 \to u_2 \to \ldots \to u_{k-1} \to j$ holds. We get the slightly better time bound $0(|V|^{k-1} \cdot |E|)$ if we take into account that we have only to consider nodes u_1 with $(i, u_1) \in E$. In the case k=2 the problem can be solved also by squaring the adjacency matrix of G (which leads to an estimation $0(|V|^{\alpha})$ with $\alpha < 3$, see [1]). To our knowledge no algorithm solving this problem for arbitrary k in less than $0(|V|^{k-1} \cdot |E|)$ time has been published.

The algorithm of Latin Multiplication which is described above all in the literature (see [10])for solving this problem computes for $1 \leq p \leq k$ (or for $p = 1, 2, 4, \ldots, k$, respectively) the matrices containing all paths of length p. This algorithm also has a worst case complexity of the above order.

We have seen that for any fixed k we have a polynomial time algorithm to solve this problem but its computational behaviour is terrible if the numbers k and |V| are not very small. In this paper we will describe an algorithm whose behaviour is much better and which will solve the problem for small k rather efficiently.

Theorem 1: Let G = (V, E) be any graph and let $k \in \mathbb{N}$. The matrix $D^{(k)}(G)$ can be computed within time $0(C_k \cdot |V| \cdot |E|)$, where $C_k = k!$.

Note that we have replaced the time bound $0(|V|^{k-1} \cdot |E|)$ by $0(C_k \cdot |V| \cdot |E|)$, $C_k = k!$. Estimations of this kind can also be found for other problems. We want to mention here the vertex cover problem ([12], time bound $0(2^{\mu}/2 + |E|)$, where μ is the cardinality of the solution) and the feedback vertex set problem for undirected graphs (unpublished result of the author, time bound $0(2^{\mu} \cdot (\log\mu)^{\mu} \cdot |V| \cdot |E|)$ where again μ is the cardinality of the solution). Our new algorithm allows to compute the solution for instances, e.g. |V| = 2o and k = 7, which were outside the computational practicability before.

The estimation of our algorithm depends on the one side on k and on the other side on |V| and |E|. The dependence on k (i.e. $C_k = k!$)

is not optimal. The reader will notice that we do not estimate very sharp in our proof. We will discuss this topic again at the end of section 2. On the other hand the behaviour in $|V|$ and $|E|$ seems to be close to optimal, i.e. an improvement of this behaviour would lead to improved algorithms also for other wellknown problems. Note that already in order to compute the matrix of all paths of length 2 we need $0(|V| \cdot |E|)$ time (at least this is our present knowledge) if we are not willing to use one of the algorithms for fast matrix multiplication. A similiar observation can be made when we are faced with the problem of deciding whether there exists a cycle of length k in the given graph G. This problem can be solved by computing first the matrix $D^{(k-1)}(G)$ and then comparing $D^{(k-1)}(G)$ with E, i.e. for every $(i, j) \in E$ we look whether there exists a path of length k-1 form j to i. Therefore we can compute in time $0(C_{k-1} \cdot |V| \cdot |E|)$ whether a graph $G = (V, E)$ has a cycle of length k. The problem of determining whether a graph has a triangle (i.e. the case k=3) has been studied carefully, since for undirected graphs there exists a n^2-reduction from the problem of determining a shortest cycle to the problem of determining a triangle, [8]. Also for the problem of determining a triangle only algorithms of time complexity $0(|V| \cdot |E|)$ and the algorithms for fast matrix multiplication are known, [8]. Therefore it is likely that the time bound $0(|V| \cdot |E|)$, which holds for any fixed k, is rather sharp.

Note that as a simple corollary of Theorem 1 we have proved above the following theorem.

Theorem 2: Let $G = (V, E)$ be any graph and let $k \in \mathbf{N}$. We can decide within time $0(C_{k-1} \cdot |V| \cdot |E|)$, $C_k = k!$, whether G has a cycle of length k and compute such a cycle if it exists.

We will show in Section 4 that we can use Theorem 1 also to find a longest path and a longest cycle efficiently (for finding a longest cycle we can apply our method only if the graph is undirected).

Theorem 3: Let $G = (V, E)$ be any graph. We can compute a longest path of G within time $0(c_{\mu+1} \cdot |V| \cdot |E|)$, $C_\mu = \mu!$, where μ is the length of the longest path of G.

Theorem 4: Let $G = (V, E)$ be an undirected graph. We can compute a longest cycle of G within time $0(C_{2\mu-1} \cdot |V| \cdot |E|)$, $C_\mu = \mu!$, where μ is the length of the longest cycle of G.

It was shown before, [7], that a longest cycle in an arbitrary graph $G = (V, E)$ can be found within time $0(|V|^{\mu} \cdot |E|)$ where μ is the length of the longest cycle of G. Both problems, determining a longest path as well as determining a longest cycle, are well known and well studied and are important in many applications (see [13]).

Before we start to prove our theorems we want to give an idea about the method we use. We feel that this method is quite general and should have further applications.

Let a graph $G = (V, E)$, $V = \{1,..., n\}$, and a number k be given. The first consideration is very simple. We start from the set of edges and then we compute successively for all p, $2 \leq p \leq k$, and for all $i, j \in V$ all the paths of length p from i to j. If we have already computed for all $i, j \in V$ all the paths from i to j of length p, then we can use this information to compute the paths of length p +1 by Latin Multiplication (see [10]). This approach is very time consuming since the number of paths of length p can be very large. We overcome this difficulty by considering instead of the family of all paths of some given length between two nodes some subfamily which we really need in order to compute finally paths of length k. We call this subfamily a representative for the family of all paths.

We will describe this idea in the next section and we will also give a close upper bound for the cardinality of the optimal representatives (Theorem 5). In section 3 we show how representatives can be computed efficiently and prove theorem 1. There is still a rather large gap between the cardinality of the representatives we get in section 3 and the optimal ones. In section 4 we prove theorem 3 and theorem 4.

2. The use of representatives

Our first step is to consider instead of paths (i.e. sequence of nodes) the sets of nodes lying on a path, i.e. we don't distinguish between paths running over the same set of nodes. From now on we will use in our proof only these sets. In an implementation of our algorithm

one should encode such a set as a sequence of nodes which form a path in G in order to have really paths available when the algorithm stops.

Let us set for $1 \leq i,j \leq n, 0 \leq p \leq n-1$

$$F_{ij}^{p-1} = \{U \in P_{p-1}(n) \mid U \text{ occurs as the set of inner nodes on a path of length p from i to j}\}.$$

Here $P_{p-1}(n)$ denotes the family of all subsets of $\{1,\ldots, n\}$ of cardinality p-1. Let us consider as an example the graph G given by figure 2.1.

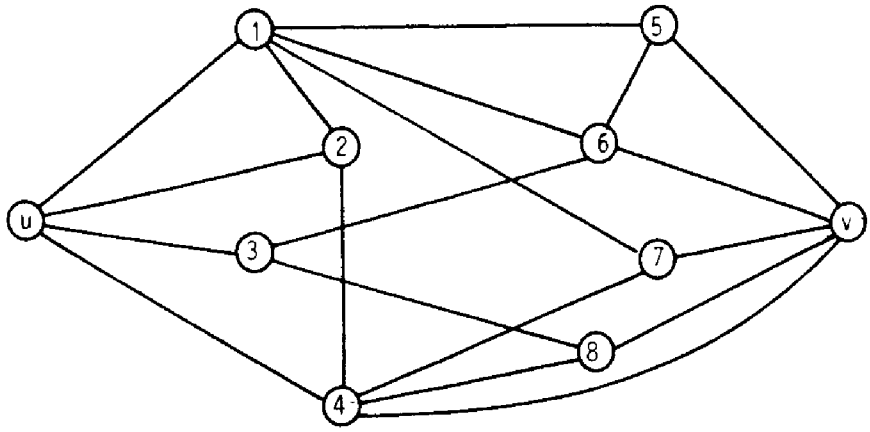

Figure 2.1: The graph G

Then $F_{uv}^{2} = \{\{2,4\}, \{1,5\}, \{1,6\}, \{1,7\}, \{3,6\}, \{3,8\}, \{4,8\}, \{4,7\}\}$.

Now let us define the notion of a representative. Let $q \in \mathbf{N}$ with $0 \leq q < n$ and let F be any family of sets over $\{1,\ldots, n\}$. A q-representative $\hat{F}$ for F is defined in such a way that if we consider any set $T \subset \{1,\ldots, n\}$ of cardinality at most q and ask whether F contains a set U with $T \cap U = \emptyset$ then we get the correct answer also by looking only through $\hat{F}$.

Definition: Let F be a family of sets over $\{1,\ldots, n\}$ and let $q \in \mathbf{N}$, $0 \leq q < n$. A subfamily $\hat{F} \subset F$ is called a q-representative of F if the following condition holds:

For every $T \in P_{\leq q}(n)$, if there exists some $U \in F$ with $T \cap U = \emptyset$ then there exists also some $\hat{U} \in \hat{F}$ with $T \cap \hat{U} = \emptyset$.

Let us consider again the above example. $\hat{F}: = \{\{2,4\},\{1,5\}\}$ is a 1-representative for F^2_{uv}. Since $\hat{F}$ contains two disjoint sets for any $T \subset \{1,\ldots,n\}$ with $|T| = 1$ the family $\hat{F}$ contains a set $\hat{U}$ with $T \cap \hat{U} = \emptyset$. Because of the analogous reason $\hat{F}: = \{\{2,4\},\{1.5\},\{3,6\}\}$ is a 2-representative for F^2_{uv} and it is not difficult to see that $\hat{F}: = \{\{2,4\}, \{1,5\},\{3,6\},\{1,7\},\{3,8\},\{4,8\}\}$ is a 3-representative for F^2_{uv}.

We have said that we will use the idea of the representative to compute the matrix $D^{(k)}$. Let $u,v \in V$ be two nodes. We have to decide whether there exists a path of length k from u to v. What do we have to know about the sets F^{k-2}_{ij}, $i,j \in V$?

A path from u to v of lenght k consists of an edge $\{u,i\} \in E$, $i \neq v$, and a path from i to v of length k-1 from i to v which does not contain u.

$$u \quad \underbrace{i \quad \ldots \quad v}_{\text{length k-1}}$$

Therefore it is sufficient to know for every $i \in V$ whether there exists a path from i to v of length k-1 not containing u. This information is given by a 1-representative for F^{k-2}_{iv}. We can formulate this simple observation in the following way:

Assume that we know 1-representatives for F^{k-2}_{ij}, $1 \leq i, j \leq n$. Then we can compute 0-representatives for F^{k-1}_{ij}, $1 \leq i,j \leq n$.

Note that for any family F a 0-representative $\hat{F}$ of F is empty iff F is empty and it has to contain only one arbitrary set from F if F is not empty.

We can easily generalize the above observation and get the following lemma which we will call the main lemma because of its importance for this paper.

Main lemma: Let p,q be numbers with $0 \leq p < n$ and $1 \leq q \leq n$. Assume that we know q-representatives for F^p_{ij}, $1 \leq i,j \leq n$, but not necessarily the sets F^p_{ij} itself. Then we can compute (q-1)-representatives for all the families F^{p+1}_{ij}, $1 \leq i,j \leq n$.

We can use the idea of the main lemma by computing first (k-2)-representatives for F^1_{ij}, $1 \leq i,j \leq n$, and then (k-3)-representatives for F^2_{ij}, $1 \leq i,j \leq n$,....., until we reach 0-representatives for F^{k-1}_{ij}, $1 \leq$

$i,j \leq n$. We will show in the next section that we can do this computation efficiently. Closely related with the complexity of this computation is the maximum number of sets which may belong to a representative. Therefore we define

$$\alpha(p, q, n) = \max_{F \subset P_p(n)} \min \{|\hat{F}| ; \hat{F} \text{ is a q-representative for } F\}$$

It is remarkable that we know this function explicitely. Results from [4, 9] imply that $\alpha(p, q, n) = \binom{p+q}{p}$ for $n \geq p+q$. Note that no proper subset of $F = P_p(p+q)$ is a q-representative of F and therefore $\alpha(p, q, n) \geq \binom{p+q}{p}$. In order to prove the other direction we have to introduce some new definitions.

Let $F \subset P_p(n)$. A set $T \subset \{1,..., n\}$ is called a hitting set of F if $U \cap T \neq \emptyset$ for all $U \in F$.

F is called q-minimal if for every $U \in F$ the family $F - \{U\}$ has a hitting set of cardinality q which is not a hitting set of F.

It is clear that $\hat{F} \subset F$ is a q-representative of F iff every hitting set of $\hat{F}$ of cardinality at most q is also a hitting set of F.
This implies that every family $F \subset P_p(n)$ has a q-representative which is q-minimal.

It was conjectured in [3] and shown in [4] and [9] (see also [2]), that every family $F \subset P_p(n)$ which is q-minimal contains at most $\binom{p+q}{p}$ sets. Therefore we get the following theorem:

Theorem 5: $\alpha(p, q, n) = \binom{p+q}{p}$ for $n \geq q + q$.

This theorem does not imply that we can compute a q-representative with cardinality $\leq \binom{p+q}{p}$ efficiently. The method which we will use in the next section leads only to q-representatives of cardinality $\sum_{i=1}^{q} p^i$. Therefore we do not think that the constant $C_k = k!$ in our Theorem 1 is close to be optimal. Note that a lower bound for C_k using the method of representatives is given by

$$\sum_{p=1}^{k-1} \alpha(p,k-p-1,n) = \sum_{p=1}^{k-1} \binom{k-1}{k-p-1} = \sum_{r=0}^{k-2} \binom{k-1}{r} = 2^{k-1} - 1.$$

It was already noticed in [3] that $\alpha(p, q, n) \leq \sum_{i=1}^{q} p^i$

The author realized the connections between the work of [3, 4, 9] and the work presented here only during the last stage of preparing this paper.

3. Proof of theorem 1:

We want to compute the matrix $D^{(k)} = (d_{ij}^{(k)})$, where $d_{ij}^{(k)}$ is some special path from i to j of length k, if it exists, and $d_{ij}^{(k)} = \lambda$ otherwise. As we described in the introduction we have to compute (k-p-1)-representatives for all the sets F_{ij}^{p}, $i,j \in V$, $1 \leq p \leq k-1$.

Actually we define trees whose nodes are labelled with the sets from F_{ij}^{p} such that the family of all the sets which occur as node labels in this tree form a (k-p-1)-representative of F_{ij}^{p}. The tree structure enables us to do the computations, described by the main lemma, efficiently. We will call such a tree a (k-p-1)-tree for F_{ij}^{p}.

Definition: Let $F \subset P_p(n)$ be a family of sets. Let q be some natural number. A q-tree for F is a p-nary node labelled and edge labelled tree of height at most q which satisfies the following conditions:

(i) Its nodes are labelled with sets from F or with the special symbol λ. Its edges are labelled with elements from $\{1,..., n\}$.

(ii) If a node is labelled with some set $U \in F$ and if its depth is less than q, then it has p sons and each of the p elements of U occurs as a label of one of the edges connecting this node with its sons.

(iii) If a node is labelled with the special symbol λ or if its depth is equal to q, then it has no sons.

(iv) Between the labels of the nodes and the edges the following relation holds: For any node ξ of this tree, if $E(\xi)$ is the set of elements from $\{1,..., n\}$ occurring as edge labels on the path from the root of this tree to ξ, then either label $(\xi) \in F$ and label $(\xi) \cap E(\xi) = \emptyset$ or label $(\xi) = \lambda$ and there exists no $U \in F$ with $U \cap E(\xi) = \emptyset$.

As an example (see figure 3.1) we want to describe a 3-tree for the set F_{uv}^{2} which we considered in the introduction, i.e. for $F = F_{uv}^{2} =$

$\{\{2,4\},\{1,5\},\{1,6\},\{1,7\},\{3,6\},\{3,8\},\{4,8\},\{4,7\}\}$. Note that for $0 \leq q \leq 2$, the first q levels of this tree form a q-tree for the family F.

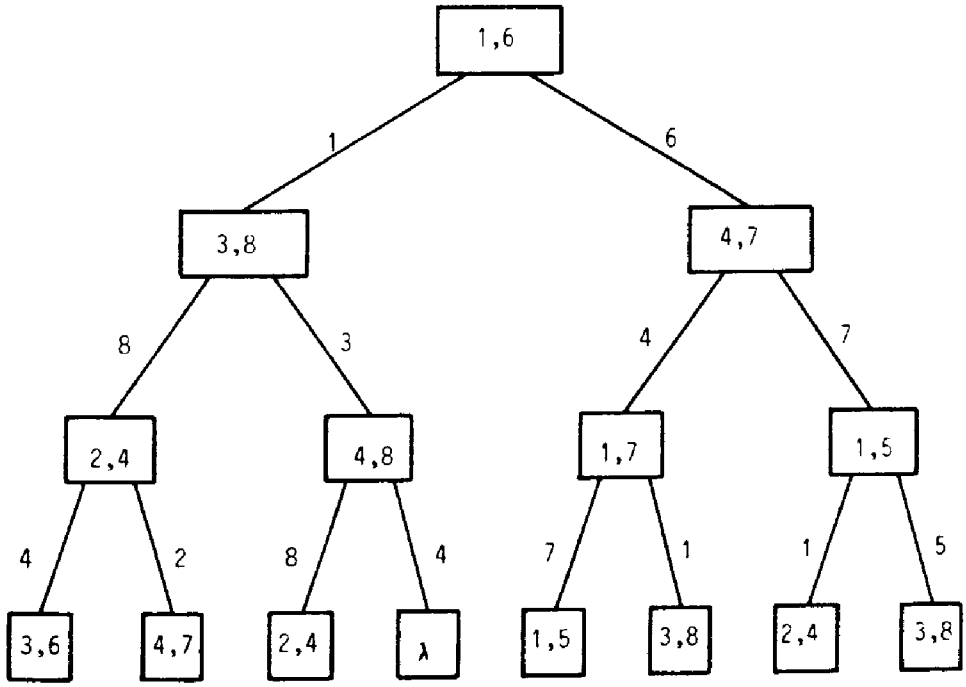

Figure 3.1: A 3-tree for the set F^2_{uv}

Lemma 1: Let $F \subset P_p(n)$ be a family of sets, let q be some natural number and let B be some q-tree for F. Then the family $\hat{F}$ consisting of all sets which occur as node labels in B form a q-representative of F. Furthermore we can decide for every $T \in P_q(n)$ in $0(p \cdot q)$ steps whether there exists some $U \in F$ with $T \cap U = \emptyset$ and compute such a U if it exists.

Proof: We will prove the second assumption first. Consider the following algorithm:

procedure Disjoint-Set (ξ: node of B; T: element of $P_{\leq q}(n)$)
begin
  if label (ξ) $\neq \lambda$ *and* label (ξ) $\cap$ T $\neq \emptyset$ *then*
  begin
    Let $\hat{\xi}$ be some son of ξ such that the edge
    from ξ to $\hat{\xi}$ is labelled with some element

```
    a ϵ label (ξ) ∩ T;
    call Disjoint-Set (ξ̂, T)
  end else
  If label (ξ) = λ then write (There exists no U ϵ F with U ∩ T = ∅)
  else if label (ξ) ∩ T = ∅ then write (U = label (ξ) fullfills U ∈ F
       and U ∩ T = ∅);
end;
```

Initially we call this procedure with Disjoint-set (root of B, T) and we have to show that it always produces the correct output. We observe three facts:

1.) If the algorithm finds a node ξ with Label $(\xi) \cap T = \emptyset$ then clearly U = label (ξ) has the property that $U \epsilon F$ and $U \cap T = \emptyset$ since every node label either is the special symbol λ or a set belonging to F.

2.) Now assume that the algorithm reaches a node ξ with label $(\xi) = \lambda$. Let $E(\xi)$ be defined as in the definition of the q-tree. This definition implies that there exists no $U \epsilon F$ with $U \cap E(\xi) = \emptyset$. But because of our algorithm $E(\xi) \subset T$ and therefore there exists no $U \epsilon F$ with $U \cap T = \emptyset$.

3.) There still is to show that always one of the write-statements is reached. If ξ is a node of depth $\hat{q}$, $\hat{q} < q$, then either we reach a write-statement or we call the procedure again with some node $\hat{\xi}$ of depth $\hat{q} + 1$. If ξ is a node of depth q, then $|E(\xi)| = q$ and since on the other hand $E(\xi) \subset T$ and $|T| = q$ we can conclude that in this case $E(\xi) = T$. Therefore if label $(\xi) \neq \lambda$ then label $(\xi) \cap T = \emptyset$ and we reach a write-statement since the condition of the while-statement is not fullfilled.

We have shown now that our algorithm computes a set $U \epsilon F$ with $T \cap U = \emptyset$ if such a set exists. The computation needs $0(q \cdot p)$ steps, since the number of calls of the procedure is bounded by the depth of the tree B (and this depth is bounded by q) and since during every call two sets of size p have to be compared (which needs 0(p) steps).

Thus our second assumption is proved. The first assumption follows directly from the above consideration since the above algorithm

computes for every $T \in P_{\leq q}(n)$ some set $U \in F$ with $T \cap U = \emptyset$ if such a set U exists. Furthermore this set U occurs as a node label of tree B and therefore it belongs to $\hat{F}$. Thus $\hat{F}$ is a q-representative of G. □

Being a p-nary tree of depth at most q, B has at most $(p^{q+1}-1)/(p-1)$ nodes and therefore the cardinality of the representative $\hat{F}$ is bounded by $(p^{q+1}-1)/(p-1)$.

Now we want to show that if q-trees for all the sets F_{ij}^p $1 \leq i, j \leq n$, are given, then we can compute efficiently (q-1)-trees for the sets F_{ij}^{p+1}.

Lemma 2: Let $0 \leq p \leq n$, $1 \leq q \leq n$. Assume that q-trees $B_{i,j}^p$ for F_{ij}^p, $1 \leq i,j \leq n$, have already been computed. For $u,v \in \{1,..., n\}$ we can compute a (q-1)-tree for F_{uv}^{p+1} in time $0(q \cdot (p+1)^q \cdot \text{degree}(u))$, where degree (u) is the degree of u in the graph G.

Proof: We compute the node labels and the edge labels of the (q-1)-tree B for F_{uv}^{p+1} level-wise, i.e. we compute first the label of the root of B and the labels of the edges leaving the root. After having computed the labels for all the nodes of depth i and all the edges connecting nodes of depth i with nodes of depth i+1, we determine the labels for the nodes of depth i+1.

Now let ξ be some node of depth i+1. Let $E(\xi) \subset \{1,..., n\}$ be the set of edge labels on the path from the root to ξ. Note that all these edge labels have already been computed. We have to find a set $U \in F_{uv}^{p+1}$ with $U \cap E(\xi) = \emptyset$ (if it exists).

Note that every path from u to v of length p+2 consists of one edge $(u,w) \in E$, $w \neq v$, and a path from w to v of length p+1 which does not contain the node u. Therefore there exists a set $U \in F_{u,v}^{p+1}$ with $E(\xi) \cap U = \emptyset$ if and only if there exists some $w \in \{1,... n\} - \{v\}$ with $(u,w) \in E$ and some $\hat{U} \in F_{wv}^p$ with $\hat{U} \cap (E(\xi) \cup \{u\}) = \emptyset$. But for every $w \in V$ with $(u, w) \in E$ we can decide because of lemma 1 in $0(p \cdot q)$ steps whether there exists a set $\hat{U} \in F_{wv}^p$ with $\hat{U} \cap (E(\xi) \cup \{u\}) = \emptyset$. Since we do this computation at most degree (u) times, we can compute one node label in time $0(p \cdot q \cdot \text{degree}(u))$. Computing the labels of the edges leaving this node takes no additional time. The lemma follows since B has at most $\frac{(p+1)^q}{q}$ nodes. □

Note that F_{ij}^{o} contains exactly the empty set if $(i,j) \in E$ and it is the empty family if $(i,j) \notin E$. Therefore a q-tree for F_{ij}^{o} has the form $\emptyset$, if $(i, j) \in E$ and the form λ, if $(i, j) \notin E$.

We have to compute the matrix $D^{(k)}$ which we get because of Lemma 1, if we know all the 0-trees for F_{ij}^{k-1}, $1 \leq i,j \leq n$. We start from the (k-1)-trees for F_{ij}^{o}, $1 \leq i,j \leq n$, (which we don't have to compute since these 'trees' are given by the set of edges E) and then we compute successively the (k-2)-trees for $F_{i,j}^{1}$, $1 \leq i,j \leq n$, the (k-3)-trees for F_{ij}^{2}, $1 \leq i,j \leq n$, and so on. All these computations can be performed because of Lemma 2 within the time.

$$c \cdot \sum_{p=1}^{k-1} p^{k-p} \cdot (k-p) \cdot |V| \cdot |E| \leq c \cdot (k-1) \cdot \sum_{p=1}^{k-1} p^{k-p} \cdot |V| \cdot |E| .$$

It can be shown easily by induction that $\sum_{p=1}^{k-1} p^{k-p} \leq (k-1)!$ for $k \geq 5$. Thus we have proved Theorem 1.

Theorem 1: Let $G = (V, E)$ be any graph and let $k \in \mathbb{N}$. The matrix $D^{(k)}(G)$ can be computed within time $0(C_k \cdot |V| \cdot |E|)$, where $C_k = k!$.

4. Proof of theorem 3 and theorem 4

It is clear that we find a longest path by computing successively the matrices $D^{(1)}, D^{(2)}, D^{(3)}, \ldots$ until we reach for the first time a matrix $D^{(\ell)}$ whose entries are all equal to λ. Then the matrix $D^{(\ell-1)}$ has some entry which is not equal to λ and this entry is a longest path. The computation of $D^{(1)}, D^{(2)}, \ldots, D^{(\ell)}$ needs no more time than the computation of only $D^{(\ell)}$. This is true since when we have computed some $D^{(k)}$ and have to compute $D^{(k+1)}$ then all the trees which have been constructed while computing $D^{(k)}$ can be used and have to be enlarged by one level.

Theorem 3: Let $G = (V, E)$ be any graph and let μ be the length of the longest path of G. We can compute a longest path of G within time $0(C_{\mu+1} \cdot |V| \cdot |E|), C_{\mu} = \mu!$.

The application of our method for computing a longest cycle is not so obvious. We are able to do so only for undirected graphs. In the case of undirected graphs there is some relationship between the length of the longest path and the length of the longest cycle. It was shown in [11] that in any k-connected graph with a longest path of length ℓ the length of the longest cycle is at least $\frac{2k-4}{3k-4} \cdot \ell$. We will not use this result here but use some simple lemma.

Lemma 3: Let G be an undirected graph and let Δ be the diameter of G. Suppose there exists a cycle C with $|C| \geq 2 \cdot \Delta + 2$. Then there exists also a cycle $\hat{C}$ with $\frac{1}{2}|C| < |\hat{C}| < |C|$.

As usually the diameter denotes the maximum distance in G.

Before we prove the lemma we want to show that it gives a sharp estimation. Consider the graph G given by figure 4.1.

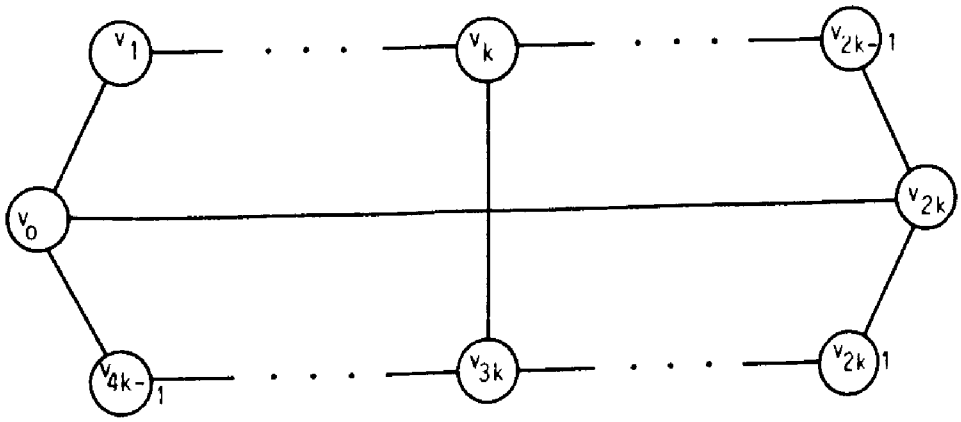

Figure 4.1: The graph G

This graph has a cycle of length 4k, its diameter is $\Delta = k + 1$ and besides its Hamiltonian cycle it has only cycles of length $2k + 1$ and $2k + 2$.

Proof of lemma 3:

Let C be a cycle with $|C| \geq 2\Delta + 2$. Set $k = \left\lfloor \frac{|C|}{2} \right\rfloor$. Then $k > \Delta$ holds. Let a,b be two nodes on C such that both paths from a to b on C have length at least k. Let P_1, P_2 be the two paths on C from a to b. Then $|P_1|$, $|P_2| \geq k$. Let P be a shortest path from a to b in G, $|P| \leq \Delta < k$.

We have to consider two cases.

(i) Except for the endpoints P and P_1 (or P and P_2, respectively) are vertex-disjoint. Then $\hat{C} = PP_1$ (or $\hat{C} = PP_2$, respectively) fulfills the conditions of the lemma.

(ii) Besides a,b the path P contains some further node from P_1 and some further node from P_2. Then we can assume that there exist d and e such that the path P has the from described by figure 4.2 and the following conditions hold:

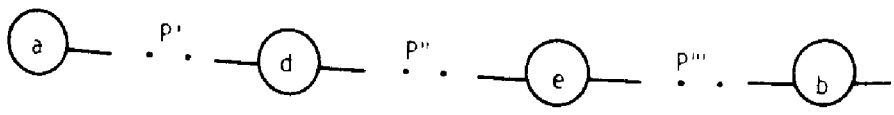

Figure 4.2: Partition of the path P

d belongs to $P_1 - \{a\}$, e belongs to $P_2 - \{b\}$, P' contains no inner point from P_2 and P'' contains no point from P_1 or from P_2.

Let P_{11}, P_{12}, P_{21}, P_{22} be the subpaths of P_1, P_2 defined by d and e (see figure 4.3).

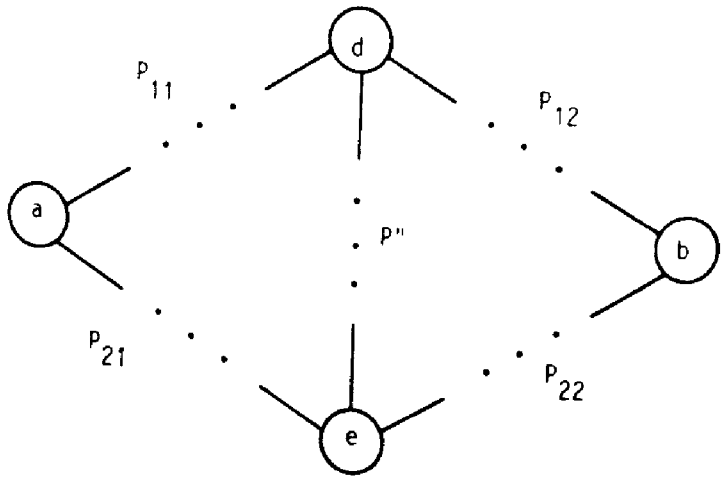

Figure 4.3: Cycle C, path P''

Since P is the shortest path from a to b we know that $|P''| \leq |P_{12}|$ and $|P''| \leq |P_{21}|$ holds. Let C_1 denote the cycle $P_{11}P''P_{21}$ and let C_2 denote the cycle $P_{12}P''P_{22}$. Then $|C_1| + |C_2| = |C| + |P''| > |C|$ and $|C_1| < |C|$ (because of $|P''| < |P_{12}|$) and $|C_2| < |C|$ (because of $|P'''| < |P_{21}|$). Therefore the longer one of the two cycles C_1, C_2 fulfills the conditions of the lemma. □

We use this lemma in order to compute a longest cycle. We can assume that the graph is biconnected (otherwise we compute the biconnected components, this needs time $0(|E|)$, see [1,5]). Then we compute the diameter Δ of G and for two nodes a,b with distance Δ we determine two vertex-disjoint paths from a to b, i.e. we compute a cycle of length k, $k \geq 2\Delta$ (this computation needs time $0(|V| \cdot |E|)$, see [1,5]. Then we compute successively D^{ℓ} for $\ell = k, k+1, \ldots$. By comparing D^{ℓ} with E we check whether there exists a cycle of length $\ell + 1$. We stop when we have reached for the first time some $D^{2\ell-1}$ such that there exists no cycle of length μ for $\ell < \mu \leq 2\ell$. Because of Lemma 3 we know that in this case the length of the longest cycle is equal to ℓ.

Theorem 4: Let G be an undirected graph and let μ be the length of the longest cycle of G. We can compute a longest cycle of G within time $0(C_{2\mu-1} \cdot |V| \cdot |E|)$, $C_{\mu} = \mu!$.

Acknowledgement: The author wants to thank R. Schulz, E. Speckenmeyer and 0. Vornberger for the many discussions we had during the preparation of this paper.

References

[1] Aho, A.V., J.E. Hopcroft and J.D. Ullman: The Design and Analysis of Computer Algorithms, Addison-Wesley, 1974

[2] Berge, C: Graphs and Hypergraphs, North Holland-American Elserier, 1973

[3] Erdös, P, and T. Gallai: On the Minimal Number of Vertices Representing the Edges of a Graph, Publ. Math. Inst. Hung. Ac. Sc. (Mag. Tud. Akad.) 6(1961), 181 - 203

[4] Erdös, P., A. Hajnal and J. Moon: A problem in Graph Theory, Math. Notes, Am. Math. Monthly 71(1964), 1107 - 1110

[5] Even, S.: Graph Algorithms, Pitman Publishing Limited, 1979

[6] Garey, M.R. and D.S. Johnson: Computers and Intractability, Freeman and Company, 1979

[7] Hsu, W., Y. Ikura and G.L. Nemhauser: A polynomial algorithm for maximum weighed vertex packings on graphs without long odd cycles, Math. Progr. 2o(1981, 225 - 232.

[8] Itai, A. and M. Rodeh: Finding a Minimum Circuit in a Graph, Proc. 1977 ACM Symp. Theory of Computing, 1 - 10

[9] Jaeger, F. and C. Payan: Détermination du nombre maximum d'âretes d'un hypergraphe T-critique, C.R. Acad. Sc. Paris 273(1971), 221 - 223

[10] Kaufmann, A. : Graphs, Dynamic Programming and Finite Games, Academic Press, 1967

[11] Locke, S.C.: Relative Lengths of Paths and Cycles in k-Connected Graphs, J. Comb. Th. B 32(1982), 206 - 222

[12] Monien, B. and 0. Vornberger: unpublished paper

[13] Roy, B. : Algebre moderne et théorie des graphes, tome 1,2, Dunod, 1969