ALGORITHMS FOR THE QUICKEST PATH PROBLEM AND THE ENUMERATION OF QUICKEST PATHS*

J. B. ROSEN†, S.-Z. SUN‡ and G.-L. XUE§

†Computer Science Department, University of Minnesota, Minneapolis, MN 55455, U.S.A. and
‡Institute of Operations Research, Qufu Normal University, Qufu, Shandong, People's Republic of China

(Received April 1990; in revised form February 1991)

Scope and Purpose—Shortest path problems are among the most commonly encountered problems at the interface of computer science and operations research. In a recent paper, Chen and Chin proposed a new variant of shortest path problem, termed quickest path problem. The quickest path problem is to find a path to send a given amount of data from the source to the sink with minimum transmission time, where the transmission time is dependent on both the capacities and the traversal times of the arcs in the network. This problem model has the conventional shortest path problem and the largest capacity path problem as two special cases and has important applications in the management of road transportation and in communication networks. In this paper, efficient algorithms are developed for the single pair quickest path problem, the constrained quickest path problem, and the enumeration of all paths with nondecreasing order of transmission time.

Abstract—Let \( N = (V, A, c, I) \) be an input network with node set \( V \), arc set \( A \), positive arc weight function \( c \) and nonnegative arc weight function \( I \). Let \( r \) be the amount of data to be transmitted. The quickest path problem is to find a routing path in \( N \) to transmit the given amount of data in minimum time. In a recent paper, Chen and Chin proposed this problem and developed algorithms for the single pair quickest path problem with time complexity \( O(re + rm \log n) \), where \( n = |V| \), \( e = |A| \), and \( r \) is the number of distinct capacity values of \( N \). In this paper, we first develop an alternative algorithm for the single pair quickest path problem with same time complexity and less space requirement. We then study the constrained quickest path problem and propose an \( O(re + m \log n) \) time algorithm. Finally, we develop an algorithm to enumerate the first \( m \) quickest paths to send a given amount of data from one node to another with time complexity \( O(rme + rmn^2 \log n) \).

1. INTRODUCTION

Shortest path problems are the most commonly encountered problems at the interface of computer science and operations research. Examples are the largest capacity path problem, the minimum cost-reliability ratio problem, the minimum cost-time ratio problem, the most reliable path problem, and various routing problems [1–6]. In a recent paper [3], Chen and Chin proposed a new variant of shortest path problem called the quickest path problem. Basically, the problem is to find a quickest path to send a given amount of data from one node to another. To make the present paper self-contained, we will briefly describe the problem below. For more details and its applications see [3].

We have a network \( N = (V, A, c, I) \), where \( G = (V, A) \) is a digraph without multiple arcs and self-loops, \( c(u, v) > 0 \) is the capacity for an arc \((u, v) \in A\), \( I(u, v) \geq 0 \) is the lead time for an arc \((u, v) \in A\). Throughout this paper, we will use \( n \) and \( e \) to represent the number of nodes \(|V|\) and number of arcs \(|A|\) respectively.

*This research was supported in part by the Air Force Office of Scientific Research grant AFOSR-87-0127, the National Science Foundation grant DCR-8420935 and a University of Minnesota Graduate School Doctoral Dissertation Fellowship. The authors of this paper are listed in alphabetical order.
†J. B. Rosen is a Professor of Computer Science at the University of Minnesota. His research interests include large-scale numerical optimization, global optimization, and parallel computing. He has published over 100 technical papers, is an editor of two journals and is the author or editor of 5 books.
‡S.-Z. Sun is a Ph.D. student in the Computer Science Department, University of Minnesota. His research interests include combinatorial optimization and parallel computing. He has previously published in the Journal of Shandong University.
§G.-L. Xue is a Ph.D. student in the Computer Science Department, University of Minnesota. He is also from the Institute of Operations Research, Qufu Normal University. His research interests lie in the design and implementation of sequential and parallel algorithms for optimization problems. He has published papers in Operations Research, ORSA Journal on Computing, Computers & Operations Research, JOTA and others.
*All correspondence should be addressed to: Dr G.-L. Xue, Computer Science Department, University of Minnesota, Minneapolis, MN 55455, U.S.A.
Suppose $p = (v_1, v_2, \ldots, v_k)$ is a $v_1 - v_k$ path, then the lead time along path $p$ is
\[
 l(p) = \sum_{i=1}^{k-1} l(v_i, v_{i+1}),
\]
and the capacity of path $p$ is
\[
 c(p) = \min_{1 \leq i \leq k-1} c(v_i, v_{i+1}).
\]
To send $\sigma > 0$ units of data from $v_1$ to $v_k$ through path $p$, the total transmission time required is
\[
 T(\sigma, p) = l(p) + \frac{\sigma}{c(p)}.
\]
For convenience, if path $p$ does not exist, we define the total transmission time along path $p$ to be infinity. Throughout this paper, we will assume that the $r$ distinct capacity values of the given network $N$ are $c_1 < c_2 < \cdots < c_r$. It is clear that $r$ can never exceed $e$, the number of arcs in $N$. We will assume that the graph $G$ is connected. Therefore $n \leq e + 1$ and $e = O(n^2)$.

Given a pair of nodes $u, v \in V$ and a positive number $\sigma$, the single pair quickest path problem is to find a quickest path to send $\sigma$ units of data from $u$ to $v$. Given a source node $u \in V$ and a positive number $\sigma$, the one-to-all quickest path problem is to find a quickest path to send $\sigma$ units of data from $u$ to all the other nodes. Given a positive number $\sigma$, the all-to-all quickest path problem is to find, for each pair of nodes $u$ and $v$, a quickest path to send $\sigma$ units of data from $u$ to $v$.

If $r = 1$, then the quickest path problems above reduce to the corresponding shortest path problems with respect to the weight function $l$. In the general case where $r \geq 2$, the shortest path from one node $s$ to another node $t$ with respect to the weight function $l$ is usually not the quickest path to send $\sigma$ amount of data from $s$ to $t$ [3]. Therefore the quickest path problem model is a nontrivial generalization of the shortest path problem model. In Fig. 1 we give a simple example of a network $N$ where the properties of each arc are given by the pair $(c, l)$. In this example the shortest $A-F$ path with respect to the weight function $l$ is $(A, B, E, F)$ with path length 40 (and transmission time 60 to send 100 units of data) while the quickest $A-F$ path to send 100 units of data is $(A, C, E, F)$ with total transmission time 59. The quickest $A-F$ path to send 100 units of data is $(A, D, E)$ with total transmission time 48. Although $(A, C, E)$ is a subpath of $(A, C, E, F)$, it is not a quickest $A-E$ path to send 100 units of data because its total transmission time is 49 which is greater than the transmission time of $(A, D, E)$. We will use this illustrative example throughout this paper.

In [3], Chen and Chin studied the single pair problem and proposed an algorithm with time complexity $O(re + rn \log n)$ and an additional space requirement of $O(re)$. They also gave an efficient algorithm to find the single pair quickest path as a function of $\sigma > 0$.

In this paper, we first develop an alternative algorithm for the single pair problem with time complexity $O(re + rn \log n)$ and additional space requirement $O(n)$ in Section 2. We then study the single pair problem with simple constraints and design an $O(re + rn \log n)$ time algorithm in Section 3. Finally, we discuss how to enumerate all the paths to send $\sigma$ units of data from a source to a sink with nondecreasing order of transmission time and develop an $O(rmne + rmn^2 \log n)$ time algorithm to enumerate the first $m$ quickest paths to send $\sigma$ units of data from the source to the sink.

Fig. 1. Simple example showing $(c, l)$ for each arc.
2. AN ALTERNATIVE ALGORITHM FOR THE SINGLE PAIR PROBLEM

In this section, we develop some basic theory and an alternative algorithm for the single pair quickest path problem with same time complexity and less space requirement.

Given the network \( N = (V, A, c, I) \), a positive number \( \sigma \), and a pair of nodes: the source \( s \), and the sink \( t \). Our goal is to find a quickest path to send \( \sigma \) units of data from \( s \) to \( t \).

For any positive number \( w \), we define \( N(w) = (V, A(w), c, I) \) to be a subnetwork of \( N \) where \((u, v) \in A(w) \) if and only \((u, v) \in A \) and \( c(u, v) \geq w \). For any pair of nodes \( u \) and \( v \) in \( N(w) \), we will refer to the shortest \( u-v \) path in \( N(w) \) with respect to the weight function \( I \) as the shortest \( u-v \) path in \( N(w) \). We will use this notation in the rest of this paper.

Observation 1. Let \( p \) be any \( s-t \) path in \( N \). Then \( c(p) \in \{c_1, c_2, \ldots, c_r\} \).

Proof. This follows directly from the definition (1).

In [3], Chen and Chin pointed out that the quickest path problem does not have the nice property that “any subpath of a shortest path must itself be a shortest path” possessed by the shortest path problem. The following theorem shows that the quickest path problem has some close relationship with the shortest path problem and therefore has a similar property which is good enough for us to develop efficient algorithms.

Theorem 1. Let \( p \) be a quickest \( s-t \) path in \( N \) to send \( \sigma \) units of data. Then

1. \( p \) is a shortest \( s-t \) path in \( N(c(p)) \);
2. any subpath of \( p \) must itself be a shortest path in \( N(c(p)) \).

Proof. Let \( q \) be any \( s-t \) path in \( N(c(p)) \). By the definition of \( N(w) \) and equation (2), \( q \) is an \( s-t \) path in \( N \) with capacity \( c(q) \geq c(p) \). Since \( p \) is a quickest path in \( N \), we have

\[
l(p) + \sigma/c(p) = T(\sigma, p) \leq T(\sigma, q) = l(q) + \sigma/c(q).
\]

Therefore \( l(q) \geq l(p) \) and (1) is proved.

(2) is a corollary of (1) and the shortest path property that any subpath of a shortest path must itself be a shortest path.

The next theorem suggests an efficient algorithm for the single pair quickest path problem. It is similar to Theorem 1 of [3] which forms the basis for their algorithm, but not the same.

Theorem 2. Let \( p_j \) be a shortest \( s-t \) path in \( N(c_j) \), \( j = 1, \ldots, r \). Let

\[
l(p_j) + \sigma/c(p_j) = \min_{i \leq j \leq r} \{l(p_i) + \sigma/c(p_i)\}.
\]

Then \( p_i \) is a quickest \( s-t \) path in \( N \) to send \( \sigma \) units of data.

Proof. Let \( p \) be a quickest \( s-t \) path in \( N \) to send \( \sigma \) units of data. By Observation 1, \( c(p) = c_{j_0} \) for some \( j_0 \in \{1, 2, \ldots, r\} \). By Theorem 1 and the assumption of this theorem, both \( p \) and \( p_{j_0} \) are shortest \( s-t \) paths in \( N(c(p)) \). Therefore \( l(p_{j_0}) = l(p) \) and \( c(p_{j_0}) \geq c(p) \). As a consequence, we have \( T(\sigma, p_{j_0}) \leq T(\sigma, p) \). Since \( T(\sigma, p_{j_0}) \leq T(\sigma, p_{j_0}) \) by equation (5), it follows that \( T(\sigma, p_{j_0}) \leq T(\sigma, p) \). Therefore \( q_i \) must be a quickest \( s-t \) path in \( N \) to send \( \sigma \) units of data.

Note that for \( j = 1, \ldots, r \), we may compute \( p_j \) and \( l(p_j) \) and also \( c(p_j) \) repeatedly using the network \( N \) and the shortest path algorithm of [4] with the modification in the sense that whenever \( c(u, v) < c_j \), we just think that the arc \((u, v)\) does not exist in carrying out the shortest path algorithm. With this in mind, we have the following algorithm.
Algorithm 1 (single pair quickest path)

Step 1. For \( j = 1, \ldots, r \), compute a shortest \( s-t \) path \( p_j \) in \( N(c_j) \); also compute the lead time \( l(p_j) \) and the capacity \( c(p_j) \).

Step 2. Find index \( k \) which minimizes \( \{l(p_j) + \sigma/c(p_j) \mid j = 1, \ldots, r \} \). \( p_k \) is the quickest \( s-t \) path in \( N \) to send \( \sigma \) units of data.

Since each of the \( p_j \) requires \( O(e + n \log n) \) time \([4]\), Step 1 takes \( O(re + rn \log n) \) time. It is clear that Step 2 takes \( O(r) \) time. Therefore the time complexity of Algorithm 1 is \( O(re + rn \log n) \). Besides the space required for storing the input network \( N \), Algorithm 1 also requires \( O(rn) \) additional space to store the (at most) \( r \) shortest paths. In the actual implementation of Algorithm 1, we may combine Step 1 and Step 2 and store only the current best \( s-t \) path. By doing so, we may reduce the additional space requirement to \( O(n) \) without increasing the time complexity. It should be noted that the algorithm of \([3]\) always requires \( O(re) \) space to store the transformed network \( N' \) while our algorithm requires only \( O(e) \) space to store the network \( N \) to carry out the algorithm.

It is always true that \( c(p_j) \geq c_j \). If \( c(p_j) = c(p_j') \) for some \( j' > j \), then \( p_j \) is also a shortest \( s-t \) path in \( N(c_i) \) for \( i = j, j+1, \ldots, j' \). This will also save some time in the actual implementation of Algorithm 1.

Consider the network in Fig. 1. Suppose that we want to send 100 units of data from \( A \) to \( F \). In this example, \( r = 4 \). The distinct capacity values are \( c_1 = 5, c_2 = 10, c_3 = 12 \), and \( c_4 = 20 \). Applying the shortest path algorithm of \([4]\) with the modification mentioned above, we get \( p_1 = (A, B, E, F), \quad l(p_1) = 40, \quad c(p_1) = 5, \quad p_2 = (A, C, E, F), \quad l(p_2) = 49, \quad c(p_2) = 10, \quad p_3 = (A, D, E, F), \quad l(p_3) = 53, \quad c(p_3) = 12. \) Since \( p_4 \) does not exist, by the convention made in Section 1, we assume \( l(p_4) + 100/c(p_4) = \infty. \) Since \( l(p_2) = \min_{1 \leq i \leq 4} \{l(p_i) + 100/c(p_i) \} \), \( p_2 = (A, C, E, F) \) is the quickest \( A-F \) path to send 100 units of data.

A straightforward extension of Algorithm 1 solves the one-to-all quickest paths problem in \( O(re + rn \log n) \) time. Applying the one-to-all problem \( n \) times solves the all-to-all problem in \( O(ne + n^2 \log n) \) time. Since \( n \leq e \leq n^2 \) and \( r \ll e \), we can simply say that the one-to-one and one-to-all quickest path problem can be solved in \( O(n^4) \) time and the all-to-all quickest path problem can be solved in \( O(n^3) \) time.

3. THE SINGLE PAIR PROBLEM WITH SIMPLE CONSTRAINTS

In this section, we study the single pair problem with simple constraints. We are given the network \( N = (V, A, c, l) \), a positive number \( \sigma \) and a pair of nodes \( s \) and \( t \). In addition, we are given a third node \( u \) and a \( u-t \) path \( p \) which does not contain \( s \). We want to find, among all \( s-t \) paths having \( p \) as a subpath, a path \( q \) which has the least transmission time to send \( \sigma \) units of data. We will call an \( s-t \) path a \( p \)-constrained \( s-t \) path if it has \( p \) as a subpath. In this term, we are looking for a \( p \)-constrained quickest \( s-t \) path. Besides its own interests, it is also a critical tool in developing algorithms for enumerating all paths in the next section.

The path \( q \) we are looking for must be the concatenation of an \( s-u \) path and \( p \). So we must investigate the \( s-u \) paths. From network \( N \) and a given capacity \( c_i \), we get a subnetwork \( N(c_i) \). Let \( N_{up}(c_i) \) be the network obtained by deleting from \( N(c_i) \) all the nodes in \( p \) except \( u \). We have the following result.

**Theorem 3.** Let \( c(p) = c_j \) and let \( p_i \) be a shortest \( s-u \) path in \( N_{up}(c_j) \) with respect to the lead time function \( l \), \( i = 1, 2, \ldots, r \). Let \( q_i \) be the concatenation of \( p_i \) and \( p \), \( i = 1, 2, \ldots, r \). Then

1. \( q_i \) is a \( p \)-constrained \( s-t \) path, \( i = 1, 2, \ldots, r \).
2. The transmission time to send \( \sigma \) units of data along \( q_i \) is
   
   \[
   T(\sigma, q_i) = \begin{cases} 
   (l(p) + l(p_i) + \sigma/\min(c(p_i), c_j)) & \text{if } i < j \\
   (l(p) + l(p_i) + \sigma/c_j) & \text{if } i \geq j 
   \end{cases}
   \]
3. Let \( q_i \) be such that \( T(\sigma, q_i) = \min_{1 \leq i \leq r} T(\sigma, q_i) \). Then \( q_i \) is a \( p \)-constrained quickest \( s-t \) path.
Proof. Since \( p \) is a \( u-t \) path and \( p_i \) is an \( s-u \) path which shares with \( p \) no node except \( u \), their concatenation \( q_i \) must be a \( p \)-constrained \( s-t \) path. This proves (1).

Since \( q_i \) is the concatenation of \( p_i \) and \( p \), we have \( c(q_i) = \min\{c(p_i), c(p)\} = \min\{c(p_i), c_j\} \) for all \( i \). For \( i \geq j \), we have \( c(p_i) \geq c_i \geq c_j \). Therefore we have

\[
\frac{\sigma}{c(q_i)} = \begin{cases} \sigma/\min\{c(p_i), c_j\} & i < j \\ \sigma/c_j & i \geq j \end{cases}
\]

It follows from the definition of \( q_i \) that \( l(q_i) = l(p_i) + l(p) \), for all \( i \). Therefore (2) is true.

Let \( q \) be a \( p \)-constrained quickest \( s-t \) path. Then \( q \) is the concatenation of an \( s-u \) path \( p' \) and \( p \), where \( p' \) and \( p \) share no common node except \( u \). Left \( c(p') = c_{i_0} \). Then \( p' \) is an \( s-u \) path in \( \mathcal{A}_{up}(c_{i_0}) \). Since \( p_{i_0} \) is a shortest \( s-u \) path in \( \mathcal{A}_{up}(c_{i_0}) \), we have \( l(p_{i_0}) \leq l(p') \) and \( c(p_{i_0}) \geq c_{i_0} = c(p') \). Therefore \( T(\sigma, q_i) \leq T(\sigma, q_{i_0}) \leq T(\sigma, q) \). Hence \( q_i \) must also be a \( p \)-constrained quickest \( s-t \) path.

With this theorem, we propose the following algorithm.

Algorithm 2 (constrained quickest path)

Step 1. Find \( j \) such that \( c(p) = c_j \).

Step 2. For \( i = 1, \ldots, r \), compute the shortest \( s-u \) path \( p_i \) in \( \mathcal{N}_{up}(c_i) \); also compute the lead time \( l(p_i) \) and the capacity \( c(p_i) \).

Step 3. Find index \( k \) which minimizes the \( r \) numbers \( \{l(p_i) + \sigma/c(p_i) \mid i = 1, \ldots, j\} \cup \{l(p_i) + \sigma/c(p) \mid i = j + 1, \ldots, r\} \). \( q = p_k \) is a \( p \)-constrained quickest \( s-t \) path.

Step 1 takes \( O(r) \) time. Since each of the \( p_i \) requires \( O(e + n \log n) \) time \([4]\), Step 2 takes \( O(re + mn \log n) \) time. It is clear that Step 3 takes \( O(r) \) time. Therefore the time complexity of Algorithm 2 is \( O(re + mn \log n) \) or simple \( O(n^4) \).

4. ENUMERATING ALL PATHS

Given the network \( N = (V, A, c, l) \), a positive number \( \sigma \), and a pair of nodes \( s \) and \( t \). The final problem we want to study is to enumerate all the \( s-t \) paths to send \( u \) units of data with nondecreasing order of transmission time. Without loss of generality, let \( s = v_1 \) and \( t = v_u \). The following analysis is an analogue of the analysis of \([5]\) where all the paths between a pair of nodes in a weighted digraph are enumerated with nondecreasing order of path length.

Let \( p_1 = (r[0], r[1], \ldots, r[k]) \) be a quickest \( s-t \) path. Let \( P \) be the set of all \( s-t \) paths in \( N \). Then every path in \( P-\{p_1\} \) differs from \( p_1 \) in exactly one of the following \( k \) ways:

1. It contains \( (r[1], r[2], \ldots, r[k-1], r[k]) \) as a subpath, but not \( (r[0], r[1]) \).
2. It contains the edges \( (r[2], r[3], \ldots, r[k-1], r[k]) \) as a subpath, but not \( (r[1], r[2]) \).
3. \( (r[i], r[i+1], \ldots, r[k-1], r[k]) \) as a subpath, but not \( (r[i-1], r[i]) \).
4. It does not contain the edge \( (r[k-1], r[k]) \) as a subpath.

The set of paths \( P-\{p_1\} \) may be partitioned into \( k \) disjoint sets \( P^{(1)}, \ldots, P^{(k)} \) with the set \( P^{(i)} \) containing all paths in \( P-\{p_1\} \) satisfying condition \( i \) above, \( 1 \leq i \leq k \).

For each \( i = 1, \ldots, k \), let \( p^{(i)} \) be a quickest path in \( P^{(i)} \). Notice that \( p^{(i)} \) can be computed using Algorithm 2. Let \( p^{(i)} \) have the minimum transmission among \( p^{(1)}, p^{(2)}, \ldots, p^{(k)} \). Then \( p^{(i)} \) also has the least transmission time among all paths in \( P-\{p_1\} \) and hence must be a second quickest path.

The set \( P^{(i)}-\{p^{(i)}\} \) may now be partitioned into disjoint subsets using a criterion identical to that used to partition \( P-\{p_1\} \). If \( p^{(i)} \) has \( k' \) edges, then this partitioning results in \( k' \) disjoints subsets.

We next determine the quickest paths in each of these \( k' \) subsets. Consider the set \( Q \) which is the
union of these \( k' \) quickest paths and the paths \( p^{(1)}, p^{(2)}, \ldots, p^{(t-1)}, p^{(t+1)}, \ldots, p^{(k)} \). The path with minimum transmission time in \( Q \) is a third quickest path. The corresponding set may be further partitioned. In this way we may successively generate the \( s-t \) paths in nondecreasing order of transmission time.

Since the number of different \( s-t \) paths ranges from 0 to \((n - 1)!\), it is impractical to enumerate all \( v_i-u_n \) paths. Instead, we propose an algorithm to enumerate the first \( m \) quickest paths for a given natural number \( m \).

\textit{Algorithm 3 (enumerating first \( m \) quickest paths)}

\textbf{Step 1.} \( Q = \{(a \text{ quickest } s-t \text{ path, } \emptyset)\} \); 
\textbf{Step 2.} For \( i = 1 \) to \( m \) do 
\begin{enumerate}
\item Let \( (p, C) \) be the tuple in \( Q \) such that \( p \) has the least transmission time; 
\item Output path \( p \); delete the corresponding tuple from \( Q \); 
\item Find the quickest \( s-t \) paths in \( N \) under the constraints \( C \) and the additional constraints imposed by the partitioning described above; 
\item Add these quickest \( s-t \) paths together with their constraints to \( Q \); 
\end{enumerate}
\textbf{end.}

Step 1 takes \( O(re + rn \log n) \) time. Each time line 3 is executed, it may require determining \( n - 1 \) constrained quickest paths. This takes \( O(rn^2 \log n) \) time and increases the size of \( Q \) by at most \( n - 1 \). Therefore the total time consumed by line 3 is \( O(rn^2 \log n) \). Now consider lines 1, 2 and 4. If we maintain \( Q \) as a heap, then the total amount of time required by these three lines is \( O(mn \log n + O(n[\log 2 + \log 3 + \cdots + \log m]) = O(mn \log n) + O(mn \log m) \). Since \( m \) is no greater than \( n \), we have \( \log m \leq n \log n \). Therefore the total time used by these three lines is \( O(mn \log n^2) \). To summarize, the time complexity of Algorithm 3 is \( O(rn^2 \log n) \) or simply \( O(mn^2) \).

5. CONCLUSIONS

We have studied various quickest path problems and developed efficient algorithms for solving them. In particular, we solved two out of three research problems posed at the end of a recent paper [3]. In all the cases studied, the time complexities of our algorithms are \( r \) times the time complexities of the corresponding shortest path algorithms, where \( r \) is the number of distinct capacity values in the input network. This is a very satisfactory result, because the various shortest path problems are special cases of the quickest path problems when \( r = 1 \).

In the discussions of this paper, the digraph \( G \) is assumed to have no multiple arcs and self loops. In practice however, there may be more arcs from one node to another with different lead times and capacities. Our algorithms readily extends to this case, although the analysis of time complexities should be altered accordingly.

\textit{Acknowledgement—} We are grateful to two anonymous referees for their helpful comments on the first version of this paper.

\textbf{REFERENCES}