Elektronische Informationsverarbeitung und Kybernetik
EIK 18 (1982) 3, 141–151
(Journal of Information Processing and Cybernetics)

Shortest Path Algorithms for Graphs of Restricted In-Degree and Out-Degree

By *Albrecht Hübler*, *Reinhard Klette*, and *Günter Werner* [1])

Abstract. The paper was stimulated by shortest path problems in binary image processing; a binary image may be viewed as a graph of in-degree and out-degree less or equal to 4. For graphs of restricted in-degree and out-degree solutions in linear worst case time are discussed for several path problems (existence, construction, minimal paths, etc.).

Introduction

Assume a binary-valued two-dimensional array which represents a maze. The value zero means a blocked path while the value one stands for an open position. We want to develop an algorithm which begins at a set of starting positions in this maze and tries to find a path to a set of final positions, i.e. to a certain exit of the maze. Any path that satisfies these constraints is called a *feasible solution.* A feasible solution that has minimal length is called an *optimal solution.* The problem to determine such feasible or optimal solutions may appear in binary image processing (cp. [7]), e.g. for image segmentation, where "binary image" and "maze" are synonyms.

Well-known methods of algorithm design may be applied to this problem, e.g. backtracking for feasible solutions, or the greedy method as described in [6] for optimal solutions "minimal paths".

The aim of this paper is the presentation of some algorithms, developed by the authors, which are connected with the maze problem as mentioned above. The general problem we are to deal with in this paper may be characterized as follows:

Let $G = (V, E)$ be a digraph of restricted in-degree and out-degree where V denotes the set of vertices and E the set of edges. For a *starting set* $S \subseteq V$ (the source) and a *final set* $F \subseteq V$ (the destination), paths from S to F are to be analyzed, i.e., the problem may be the decision about the existence of such paths or the determination of feasible or optimal solutions, and so on.

In Section 1 both a precise formalization of these tasks and a brief review of related results in literature will be given. It will be shown that a modification of Dijkstra's algorithm [5] may be used for efficient solutions of maze problems.

In Section 2 we shall discuss several path algorithms for binary images. In this case the in-degree and out-degree of the graphs are less or equal to 4. The presented algorithms can be applied in binary image processing; for digital image processing reference is made to [10], [13], and [14].

Finally, in Section 3 a few concluding remarks are given.

[1]) Sektion Mathematik der Friedrich-Schiller-Universität Jena (Direktor: Prof. Dr. habil. *G. Schlosser*).

The computation model we shall refer to is that of a *random access machine* (RAM) in the sense of [1]. The complexity of the algorithms is measured by the time complexity in the worst case assuming the uniform cost criterion for RAM instructions. For the representation of our algorithms we have chosen Pidgin ALGOL, see [1]. For fundamental notions of graph theory or asymptotic notation see [6].

1. Path problems for graphs

Detailed treatments of algorithms for path problems in graph theory can be found, e.g., in [1], [2], and [6]. Shortest path problems for graphs were dealt with in [3, 4, 5, 8, 9, 11, 15, 16] — just to cite a few. Applications of shortest path algorithms have a very widespread literature. We shall consider some special variants of shortest path problems which were partially originated by questions that arose from binary image processing.

The *in-degree* (*out-degree*) of a digraph $G = (V, E)$ is the maximum taken over all in-degrees (out-degrees) of vertices in V. For an undirected graph in-degree and out-degree are equal. In Fig. 3 the graph possesses degree 4. For $n =$ card V and $b \leqq n$, the graph G is called *b-adjacent* iff both the in-degree and the out-degree of G are less or equal to b.

Now, let $G = (V, E)$ be an unlabeled digraph. Assume that G is b-adjacent, for b independent of $n =$ card V, and both a starting set $S \subseteqq V$ and a final set $F \subseteqq V$ are given. For such a situation, we are interested in solutions of the following problems:

(P1) Decide if there exists a path from S to F;
(P2) compute a path from S to F if such a path does exist;
(P3) find out the minimal length of paths from S to F, i.e., compute the length of optimal solutions;
(P4) compute a minimal path from S to F;
(P5) calculate the number of all different paths from S to F which have minimal length;
(P6) compute all minimal paths from S to F;
(P7) determine the maximal cardinality of a set of pairwise disjoint paths[1]) from S to F;
(P8) calculate a set of pairwise disjoint paths from S to F which has maximal cardinality.

These computational problems cannot be satisfactorily solved by direct application of algorithms given in literature for solutions of such well-known shortest path problems as:

(P9) Assume a weighting function $c(e)$ for the edges $e \in E$. The *all pairs shortest path problem* is to determine an $n \times n$ matrix A such that $A(i, j)$ is the minimal cost of a path from vertex i to vertex j.
(P10) The *single source shortest path problem* is to determine paths with minimal costs from v_0 to all the remaining vertices of G where v_0 is a fixed source vertex.

Usually both for (P9) and for (P10) it is supposed that all weights $c(e)$ are nonnegative. For this case of nonnegative edges at present no known method for the all

[1]) Two paths $v_1, v_2, \ldots, v_s$ and $u_1, u_2, \ldots, u_t$ are disjoint iff $v_i \neq u_j$ for $i = 1, 2, \ldots, s$ and $j = 1, 2, \ldots, t$.

pairs shortest path problem uses less than $\Omega(n^3)$ worst case time, see [1, 2, 6, 8, 9, 11]. *Spira* [15] gives an $\mathcal{O}(n^2(\log n)^2)$ expected time algorithm for solving (P9), cp. [3]. For the single source problem Dijkstra's algorithm [5] computes the cost of minimal paths from v_0 to each vertex and requires $\mathcal{O}(n^2)$ worst case time. Dantzig's algorithm [4] does the same within $\mathcal{O}(n^3)$ worst case time, but in practice, for some digraphs with few arcs Dantzig's algorithm may be faster. Both Dijkstra's and Dantzig's algorithm can be extended to programs which compute not only the length of minimal paths but also one minimal path for each vertex $v \neq v_0$, see [2]. Finally *Johnson* [8] has proved that problem (P10) can be solved both in time $O(e \log n)$ and in time $\mathcal{O}(ke + kn^{1+1/k})$ for any fixed constant k, on a graph with e edges and n vertices.

For a b-adjacent graph G the number of edges is restricted by $b \cdot n$. Thus problem (P10) may be solved within time $\mathcal{O}(n \log n)$ respectively $\mathcal{O}(kn^{1+1/k})$ for any constant k, on b-adjacent graphs G for a fixed integer b.

Remark that for solving these shortest path problems efficiently, one important key is given by the selection of the proper data structure for the considered digraphs. For the above mentioned algorithms, adjacency lists are the common representations for the input graphs.

To simplify our discussions of the problems (P1), ... , (P8) we shall use evident relations between these problems as illustrated in Fig. 1. Here, a continuous line represents the case that a solution for the upper problem may be used as a solution for the lower problem, too. The pointed lines have nearly the same meaning: Adding a certain counter to the solution process or at the end of this process for the upper problem, the result of this counter may be used as a solution for the lower problem.

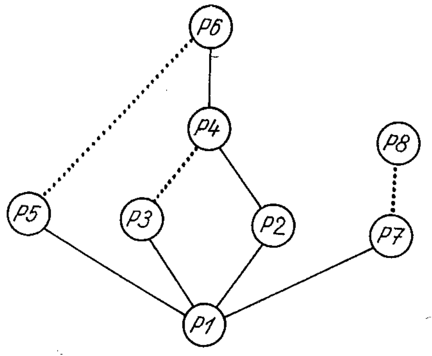

Fig. 1. Dependences between problems (P1), ... , (P8)

After these introductorily remarks, now we demonstrate that a modification of Dijkstra's algorithm may be used to solve problems (P3) and (P4) simultaneously. To this end, first of all we start with an algorithm for a simplified version of problem (P10) for undirected graphs. Then, in a short discussion we shall sketch the improvement of this algorithm to the case of processing of directed graphs. Finally it will be shown that problems (P3) and (P4) may be solved by a certain extension of the given graph followed by the application of the discussed algorithm.

Algorithm 1. Solution to (P10) for unlabeled b-adjacent undirected graphs.

Input. An unlabeled b-adjacent undirected graph $G = (V, E)$ represented by adjacency lists $L(v)$ for $v \in V$, a source $v_0 \in V$.

Output. For each $v \in V$, the length $l(v)$ of shortest paths from v_0 to v and a mapping $t\colon V \to V$ such that $v, t(v), t(t(v)), t(t(t(v))), \ldots, v_0$ is a shortest path from v to v_0, for $v \in V$.

Method. Breadth first graph traversal is used. The array $l(v)$ contains the length of the current shortest path from v_0 to v passing only through vertices which were already considered (i.e. in register u). When the algorithm stops, the array t permits the shortest paths to be explicitly specified in the manner described above. The algorithm is given in Fig. 2.

```
     begin
       $l(v_0) \leftarrow 0$;
       for $v \in V - \{v_0\}$ do $l(v) \leftarrow \infty$ od;
       for $v \in L(v_0)$ do $t(v) \leftarrow v_0$ od;
1      place vertices in $L(v_0)$ in QUEUE; $u \leftarrow v_0$;
       while QUEUE not empty do
2           for each $v$ in $L(u)$ do
               $l(v) \leftarrow \min\{l(v), l(u) + 1\}$ and, if $l(v)$ actually changes value, also
               set
               $t(v) \leftarrow u$ od;
3           for each $v$ in $L(u)$ do
               for each $w$ in $L(v)$ do
                   delete $v$ from $L(w)$ od od;
4           move first element of QUEUE to $u$;
5           add vertices in $L(u)$ to QUEUE od
     end
```

Fig. 2. Single source algorithm

Example 1. Consider the graph of Fig. 3. Let the vertex 6 be the source. Initially, $l(i) = \infty$, for $i = 1, 2, \ldots, 16$ and $i \neq 6$, $l(6) = 0$, $t(i) = 6$, for $i = 3, 5, 7, 10$ and QUEUE contains vertices 3, 5, 7, 10 (all lists $L(i)$ are assumed to be ordered). When the algorithm stops, we get $[l(1), l(2), \ldots, l(16)] = [4, 3, 1, 2, 1, 0, 1, 4, 3, 1, 4, 3, 2, 3, 6, 5]$ and $[t(1), t(2), \ldots, t(16)] = [2, 4, 6, 5, 6, 6, 6, 9, 4, 6, 9, 13, 10, 13, 16, 11]$.

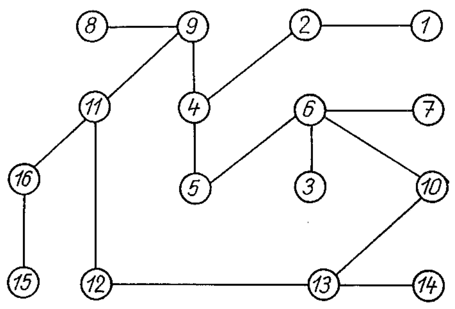

Fig. 3. A 4-adjacent graph

The proof that Algorithm 1 works correctly is by reduction to Dijkstra's algorithm. This is easily established once it is observed that the vertices in QUEUE are ordered with increasing minimal path length (to vertex v_0).

Lemma 1. *Algorithm* 1 *requires* $\mathcal{O}(b^2 \cdot n)$ *worst case time.*

Proof. Let $G = (V, E)$ be an undirected b-adjacent unlabeled graph with card $V = n$, for a fixed integer b. Lines 1 and 5 of Fig. 2 require $\mathcal{O}(b)$ time, line 4 may be done in constant time. The **for**-loop starting in line 2 requires $\mathcal{O}(b)$ time. The other **for**-loop at line 3 requires $\mathcal{O}(b^2)$ time where we assume an array LINK as described in [1, Section 2.3].

Each vertex may appear in QUEUE at most once (with the exception of v_0 that appears at most b times), thus the number of runs through the **while**-loop is restricted by n. Hence Algorithm 1 is of complexity $\mathcal{O}(b^2 \cdot n)$.

As a practical matter, we note that Algorithm 1 should be used for $b \ll n$ only. Furthermore, by passing through the program given in Fig. 2 a few improvements may be used as, e.g., to safeguard that vertex v_0 appears in QUEUE only once.

For directed graphs $G = (V, E)$ we suppose a representation by adjacency lists $L_1(v)$ and $L_2(v)$, for $v \in V$, where $L_1(v)$ contains all vertices adjacent — from v, and $L_2(v)$ all vertices adjacent — to v. Then, replacing the **for**-loop at line 3 in Algorithm 1 by

```
for each $v$ in $L_2(u)$ do
    for each $w$ in $L_1(v)$ do
        delete $v$ from $L_2(w)$ od od
```

and replacing L in lines 1, 2 and 5 by L_2, we get an algorithm that solves (P10) for unlabeled b-adjacent digraphs within $\mathcal{O}(n)$ time, for a fixed integer b. This algorithm is denoted by Algorithm 1*.

Now we explain how Algorithm 1* may be used to solve problems (P3) and (P4) simultaneously. At first assume that $S = \{v_0\}$ and $F = \{v_1\}$. This single source-single destination problem may be solved by a simplified version of Algorithm 1*. For card $S \geqq 2$ or card $F \geqq 2$ problems (P3) and (P4) are dealt with by the following process, assuming $b \geqq 2$:

(i) Two trees are added to the given graph such that the leaves of these trees coincide with the vertices in S and F given as lists, respectively, the originated graph is also b-adjacent, and in each tree, all leaves are of the same depth.

(ii) Apply a simplified version of Algorithm 1* to that single source-single destination problem defined by the roots of the both trees added in (i).

Both step (i) and step (ii) may be done within time $\mathcal{O}(n)$ where b is a fixed integer.

Theorem 1. *Problems* (P3) *and* (P4) *may be solved in time* $\mathcal{O}(n)$ *where* b *is a fixed integer.*

Example 2. Consider the graph of Fig. 3. Assume $S = \{1, 8, 15\}$ and $F = \{7, 14\}$. In step (i) we obtain a graph as shown in Fig. 4 where binary trees were used. In step (ii) a possible result may be the path $a, b, 1, 2, 4, 5, 6, 7, d$. In such a result, all nodes not in the original graph have to be deleted.

According to Fig. 1, problems (P1) and (P2) are also solvable in linear time. As stated above this paper was originated by shortest path problems in binary image

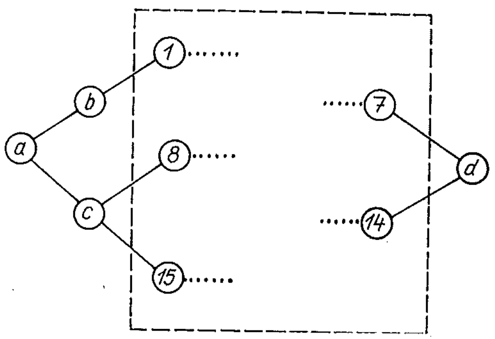

Fig. 4. Adding of two binary trees to the graph in Fig. 3

processing. Thus we shall renounce to deal with problems (P5), (P6), (P7), (P8) as general as represented in this section.

2. Path algorithms for binary images

In this paper, a binary image is a pattern of 1's on a background of 0's, filling out a scanning field of a certain size $N \times N$. In the binary image

$$A = (a_{ij})_{i,j=1,2,\ldots,N}$$

$a_{ij} = 0$ means that (i, j) is a background point, and $a_{ij} = 1$ that (i, j) is an object point. In Fig. 5, background and object points are represented by black and white squares, respectively. All object points of a binary image may be viewed as vertices of an undirected unlabeled 4-adjacent graph where two vertices are connected by an edge iff their representing object points are 4-neighbours. The 4-neighbourhood of a point (i, j) is defined to be the set $\{(i, j + 1), (i, j - 1), (i + 1, j), (i - 1, j)\}$.

For solving problems (P1), ... , (P8) for binary images

$$A = (a_{ij})_{i,j=1,2,\ldots,N}$$

considered as 4-adjacent graphs we shall use vector representations $B = (b_1, b_2, \ldots, b_{N^2})$ of the binary images instead of adjacency lists or adjacency matrices,

$$b_{(i-1)N+j} \leftarrow a_{ij}\,, \quad \text{for} \quad i, j = 1, 2, \ldots, N\,.$$

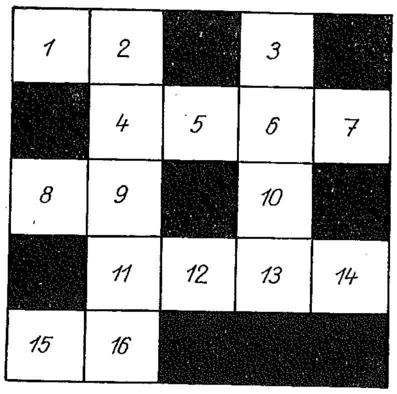

Fig. 5. Binary image to the graph in Fig. 3

Thus, the set of vertices V, the starting set S, and the final set F are subsets of $\{1, 2, \dots, N^2\}$, where $n = \operatorname{card} V = \mathcal{O}(N^2)$.

In the sequel we shall assume that a question "$v \in F$?" may be answered in constant time.

For example, a special problem may be that S represents the white squares in the leftmost and F the white squares in the rightmost column in the binary image. For this *left-to-right problem,* questions "$v \in F$?" may be answered in constant time without using any special data representation for F, and the number of points on a minimal path from S to F is greater or equal to N and less or equal to

$$\begin{array}{lll} (N^2+1)/2 & \text{if} \quad N \equiv 1 \bmod (2) & \\ N^2/2 & \text{if} \quad N \equiv 2 \bmod (4) & \\ (N^2+2)/2 & \text{otherwise} & \text{for } N \geqq 1 . \end{array}$$

Thus, it seems to be clear that all problems (P1), ..., (P8) have time complexity $\Omega(N^2)$ in the worst case for this left-to-right problem. In fact, this can be proved exactly. Furthermore, for this problem there exist binary images of size $N \times N$ with $2^{[N/2]} - 1$ different minimal paths from S to F at least. For example, consider a binary image A of size $N \times N$ where $a_{ij} = 0$ iff either $i \equiv 0 \bmod (2)$ and $j \equiv 3 \bmod (4)$, or $i \equiv 1 \bmod (2)$ and $j \equiv 1 \bmod (4)$, for $i, j = 1, 2, \dots, N$. Then from S to the point $(\lfloor N/2 \rfloor, N) \in F$, there are $2^{[N/2]} - 1$ different minimal paths exactly. Thus we have to think about a proper data structure for solutions of problem (P6). We have chosen that way that solutions of problem (P6) are represented by $N \times N$ matrices in which each object point of the input binary image is labeled by an integer $l \geqq 2$ iff a minimal path from S to this point has length $l - 2$, and $l - 2$ is at most equal to the length of a minimal path from S to F. Obviously, using this matrix any minimal path from S to F may be traced back. In Fig. 7 this matrix is given for the left-to-right problem for the binary image of Fig. 5 representing the solution to problem (P6) for this special input.

Theorem 2. *For binary images of size $N \times N$, problem* (P6) *may be solved in time $\mathcal{O}(n)$, where $n = \mathcal{O}(N^2)$ and membership in the final set F may be answered in constant time.*

The proof of this theorem is given by the following Algorithm.

Algorithm 2. Solution to (P6) for binary images.

Input. A binary image $A = (a_{ij})_{i,j=1,2,\dots,N}$ represented by a vector $B = (b_1, b_2, \dots, b_{N^2})$ with $b_{(i-1)N+j} \leftarrow a_{ij}$, for $i, j = 1, 2, \dots, N$, a starting set $S \subseteqq \{1, 2, \dots, N^2\}$ and a final set $F \subseteqq \{1, 2, \dots, N^2\}$ where membership in F may be answered in constant time.

Output. Solution to (P6) as a matrix as described above.

Method. As in Algorithm 1, breadth first graph traversal is used. Starting in S and ending in F, object points of A between S and F on minimal paths will be labeled by an integer $l \geqq 2$ so that a minimal path from S to this point has length $l - 2$. The algorithm is given in Fig. 6.

Algorithm 2 solves problem (P6) for binary images, in the manner as described, within $\mathcal{O}(N^2)$ steps because any object point of a given binary image may appear in QUEUE at most once. Applying Algorithm 2 to the left-to-right problem for the binary image in Fig. 5 yields the solution as shown in Fig. 7 where $k = 7$ at the end.

```
begin
   $i \leftarrow 0$; $l \leftarrow 2$; $k \leftarrow N^2$
   for all $j \in S$ do $b_j \leftarrow 2$ od;
   move all elements of $S$ into QUEUE;
   while QUEUE not empty do
         let $q$ be the first element in QUEUE;
         move $q$ from QUEUE to $i$;
         if $b_i < k$ then
           if $i \in F$ then $k \leftarrow l$
           else for all $j$ such that $b_j = 1$ and $j$ represents a point in the 4-
                    neighborhood of point $i$
                do concatenate $j$ to the end of QUEUE;
                   if $b_i > l$ then $l \leftarrow l + 1$ fi;
                   $b_j \leftarrow l + 1$
                od;
           fi;
         fi od;
end
```

Fig. 6. All minimal paths algorithm

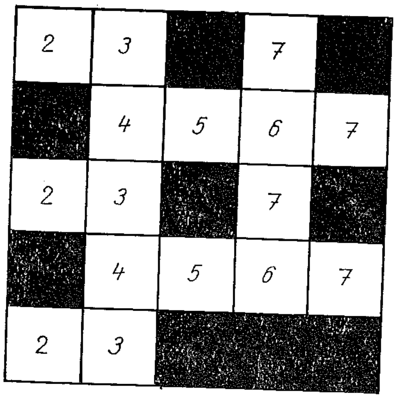

Fig. 7. All minimal paths representation of the left-to-right problem for the binary image of Fig. 5

Obviously, at the end of one run through Algorithm 2, $k - 2$ denotes the length of a minimal path from S to F. Thus, Algorithm 2 may be used for solving problem (P3), too. Furthermore, a small extension of Algorithm 2 leads to a program for solving (P5). To this end, at first Algorithm 2 is used. Then all minimal paths from S to F are traced back by the program given in Fig. 8. At the end of this process there is just one thing to do: to add all b_j's with $j \in S$. The result is exactly the solution to (P5), i.e., problem (P5) may be solved in time $\mathcal{O}(N^2)$ for binary images of size $N \times N$.

For the discussion of the problems (P7) and (P8) we restrict ourselves to the left-to-right problem for binary images. E.g., the bottom-up problem for binary images may be dealt with analogously.

The following statement is obvious.

Lemma 2. *For the left-to-right problem for binary images, two disjoint paths from S to F are lying one above the other.*

```
begin
  for $j \in F$ with $b_j = k$ do $b_j \leftarrow 1$; mark point $j$ (say with a star) od;
  $l \leftarrow k - 1$;
  while $l \geqq 2$ do
        place all unmarked points $j$ with $b_j = l$ at the end of QUEUE 1;
        while QUEUE 1 not empty do
              move first element of QUEUE 1 to $i$;
              $b_i \leftarrow \Sigma b_j$ for all marked points $j$ in the 4-neighborhood of point $i$;
              place point $i$ at the end of QUEUE 2 od;
  while QUEUE 2 not empty do
              move first element of QUEUE 2 to $i$;
              mark pont $i$ (with a star) od;
        $l \leftarrow l - 1$ od;
end
```

Fig. 8. Number of minimal paths algorithm

Thus, for solving (P8) for the left-to-right problem for binary images the following abstract program may be applied. Remember object points in binary images are labeled by 1.

```
Let $p_1, p_2, \ldots, p_m$ be the points in $S$ ordered from up to down; $i \leftarrow 1$ and $j \leftarrow 2$;
    while $i \leqq m$ do
      if there exists a path of object points from $p_i$ to $F$
        then label all points of the uppermost path of object points from $p_i$ to $F$
        by $j$;
        $j \leftarrow j + 1$ fi;
      $i \leftarrow i + 1$ od.
```

This program may be viewed as a depth first traversal of the binary images where it may be easily achieved that each object point of a given binary image will be considered at most once during the **while**-loop. The sorting of S at the beginning may be done by bucket sort in linear time. Finally, all different paths from S to F found by this program are labeled by $j = 2, 3, \ldots$, or $j_{\max}$, where $j_{\max} \leqq m = \text{card}\, S$. Altogether, we have

Theorem 3. *For binary images of size $N \times N$, problem* (P8) *may be solved in time $\mathcal{O}(n)$, where $n = \mathcal{O}(N^2)$ and S and F contain all object points in the leftmost and rightmost column, respectively.*

As a last remark, for the discussed input situation the explained program for solving (P8) may be used for solving (P7) immediately: the solution to (P7) is given by $j_{\max} - 1$.

3. Concluding remarks

The authors were interested in fast parallel algorithms for solving problems as considered in this paper for binary images, cp. [7]. But, the parallel computing systems so far considered did not offer asymptotic faster solution as stated above.

Furthermore, the authors were in search of convenient preprocessing for the $N \times N$ input binary images, e.g., a transformation in a quadtree data structure (see, e.g. [12]), such that the following shortest path algorithm may be done faster as in time $\mathcal{O}(N^2)$, for certain problems (P1), (P2), ... , or (P8). As in the parallel processing case this field could be a fruitful area for further study.

Acknowledgement

The continued discussions with our colleague Dr. *E. Creutzburg* are gratefully acknowledged, as is the help of Mrs. *Ziemak* in preparing this paper.
The authors are indebted to the referees for their remarks.

References

[1] *Aho, A. V., J. E. Hopcroft, J. D. Ullman,* The Design and Analysis of Computer Algorithms. Addison Wesley, 1974.
[2] *Berztiss, A. T.,* Data Structures. Second Edition. Academic Press, New York 1975.
[3] *Bloniarz, P. A., M. J. Fischer, A. R. Meyer,* A note on the average time to compute transitive closures. MIT/LCS/TM — 76 (1976), Laboratory for Computer Science, Massachusetts Institute of Technology.
[4] *Dantzig, G. B., W. O. Blattner, M. R. Rao,* All shortest routes from a fixed origin in a graph. In: Proceedings Theory of Graphs, Rome, July 1966; pp. 85—90.
[5] *Dijkstra, E. W.,* A note on two problems in connection with graphs. Numer. Math. **1** (1959), 269—271.
[6] *Horowitz, E., S. Sahni,* Fundamentals of Computer Algorithms. Computer Science Press, Inc., Potomac 1978.
[7] *Hübler, A., R. Klette, R. Lindner,* Elementare Operationen auf Binärbildern. Rostock. Math. Kolloq. **10** (1978), 53—62.
[8] *Johnson, D. B.,* Algorithms for shortest paths. Ph. D. Thesis, Dept. of Computer Science, Cornell University, Ithaca, New York, 1973.
[9] *Kleene, S. C.,* Representation of events in nerve nets and finite automata. In: Automata Studies (Eds.: *C. Shannon, J. McCarthy*); Princeton University Press, 1966; pp. 3—40.
[10] *Ковалевский В. А.,* Методы оптимальных решений в распознавании изображений. Наука, Москва 1976.
[11] *Robert, P., J. Ferland.* Généralisation de l'algorithme de Warshall. R.I.R.O. **2** (1968).
[12] *Rosenfeld, A.,* Quadtrees for pattern recognition and image processing. In: Proceedings 5th Internat. Conf. on Pattern Recognition, Miami Beach, Florida, December 1—4, 1980.
[13] *Rosenfeld, A., A. C. Kak,* Digital Picture Processing. Academic Press, New York 1976.
[14] Automatische Bildverarbeitung in Medizin und Biologie. Herausgeber: *H. Simon, K. D. Kunze, K. Voß, W. R. Herrmann.* Verlag Theodor Steinkopf, Dresden 1975.
[15] *Spira, P. M.,* A new algorithm for finding all shortest paths in a graph of positive arcs in average time $\mathcal{O}(n^2 \log n)$. SIAM J. Comput. **2** (1973), 28—32.
[16] *Spira, P. M., A. Pan,* On finding and updating shortest paths and spanning trees. In: IEEE 14th Annual Symposium on Switching and Automata Theory, 1973; pp. 82—84.

Kurzfassung

Der Artikel wurde durch Betrachtungen zu verschiedenen Aufgabenstellungen der Analyse kürzester Pfade in Binärbildern initiiert. Ein Binärbild kann als Graph mit durch 4 beschränktem Ein- und Ausgangsgrad betrachtet werden. In der Arbeit werden zu verschiedenen Aufgaben der Analyse kürzester Pfade in Graphen mit beschränktem Ein- und Ausgangsgrad Lösungen diskutiert, für welche höchstens lineare Zeit erforderlich ist.

Резюме

В статье рассмотрены различные постановки задач анализа кратчайших путей в бинарных изображениях. Бинарное изображение можно рассматривать как граф со степенью входа и выхода, не превышающей 4. В статье обсуждаются такие решения различных задач анализа кратчайших путей в графах с ограниченной степенью входа и выхода, для которых время решения линейно зависит от размеров изображения.

(Received: First version March 9, 1981,
final version August 17, 1981)

Authors' address:

Dr. A. Hübler,
Dr. R. Klette,
Dr. G. Werner

Sektion Mathematik der
Friedrich-Schiller-Universität Jena
6900 Jena
Universitätshochhaus
German Democratic Republic